Um Modelo Formal para a Quinta Disciplina

LOURIVAL PAULINO DA SILVA

TESE APRESENTADA AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA DA
UNIVERSIDADE DE SÃO PAULO
PARA OBTENÇÃO DO GRAU DE DOUTOR EM
CIÊNCIA DA COMPUTAÇÃO

Área de Concentração: **Ciência da Computação**

Orientador: **Prof. Dr. Flávio Soares Corrêa da Silva**

São Paulo - abril de 2004

Um Modelo Formal para a Quinta Disciplina

Este exemplar corresponde à redação final da tese de doutorado devidamente corrigida e defendida por Lourival Paulino da Silva e aprovada pela comissão julgadora.

São Paulo, abril de 2004

Banca examinadora:

Prof. Dr. Flávio Soares Corrêa da Silva (presidente)	IME-USP
Profa. Dra. Ana Cristina Vieira de Mello	IME-USP
Prof. Dr. Jaime Simão Sichman	EP-USP
Prof. Dr. José Reinaldo Silva	EP-USP
Prof. Dr. Ivan Rizzo Guilherme	IGCE-UNESP

Este trabalho é dedicado ao meu pai (in memoriam) e à minha mãe.

Agradecimentos

O desenvolvimento e a realização de atividades complexas raramente envolve uma única pessoa. No caso específico da pesquisa e redação de uma tese, esta é uma verdade patente: houve muitas contribuições, diretas ou indiretas, para a realização deste trabalho.

Os meus agradecimentos iniciais vão para o meu orientador, professor Flávio, por sua confiança, atenção, dedicação, paciência, conhecimento, e apoio durante todo o desenvolvimento deste projeto.

Agradeço também aos professores do laboratório de inteligência artificial do IME, em particular à professora Ana Cristina, pelas importantes discussões sobre a linguagem Z, *type checkers* e por me apresentar o CSP-OZ; e à professora Leliane, por suas sugestões e críticas que também contribuiram para este projeto. Estou em débito também com o Roberto Cássio, que muito me ajudou com suas críticas e observações nos seminários e na revisão desta tese. Valeu Roberto ! Também foram importantes os professores das diversas disciplinas que cursei no IME e na Poli, em particular, o professor Jaime que me apresentou o mundo dos Sistemas Multi-Agentes.

Sou muito grato aos professores Luck e D'Inverno, que se mostraram muito atenciosos sempre que tive dúvidas ou precisei de informações relativas ao SMART.

Devo também agradecimentos ao pessoal da biblioteca do IME e da secretaria da pós-graduação, em particular ao Pinho, por toda a sua ajuda e atenção. Sou também grato ao pessoal da biblioteca da FEA e ao pessoal da gráfica da Consist, em particular, ao Edson Gonçalves.

Estarei sempre em débito pela atenção, apoio e companheirismo dos meus amigos: Osvaldo, Orlando, Celso, Bruno, Isabel, Luciano, Jung, Bellocchi, Carlos Dantas, Sérgio e Carlos Cunha. Agradeço ao Luciano especialmente pela força com o Latex. Quero lembrar também o apoio fundamental do Cunha, que me cedeu o seu computador para que eu pudesse completar uma parte importante desta tese. O Orlando revisou cuidadosamente algumas partes desta tese. Tive também conversas importantes com o Osvaldo e a Isabel sobre como sobreviver a todo este processo. Além disso, a Isabel também contribuiu com algumas informações sobre Latex.

Devo, também, um reconhecimento especial à Silvana, por toda a sua ajuda no preparo dos vários relatórios e apresentações, e principalmente, por seu apoio e amizade.

Também sou muito agradecido ao pessoal da Consist, que me apoiou e permitiu que eu desenvolvesse este trabalho: Antônio Marmo, João Serrano, Jaime Teig, Luis Cláudio e todos os meus colegas das equipes de suporte técnico, de Business Intelligence/CRM. Em especial, sou grato ao Flávio Vitale por suas discussões e comentários argutos sobre a teoria de Senge e por haver contribuído com uma perspectiva gerencial para a minha análise dessa teoria. Destaco também as interessantíssimas conversas com o Sr. Raul Dicovsky que, entre outras coisas, me apresentou os trabalhos de Stafford Beer.

Minha família também foi fundamental durante todo este processo. Sou-lhes muito grato.

De um modo mais geral, agradeço a todos os meus companheiros de *squash* e de chope. Os momentos de descontração e alegria foram fundamentais para a conclusão deste trabalho.

Agradeço, em especial, minha querida Marilene, por todo seu apoio, paciência e carinho. Além disso, sua revisão atenta desta tese foi essencial. Muito obrigado.

Finalmente, agradeço a Deus.

Índice

Lista de Figuras

Lista de Tabelas

Resumo

Neste trabalho é apresentada uma aplicação de um método formal desenvolvido pela área de Engenharia de Software para produzir um modelo formal para uma teoria discursiva. A teoria estudada neste trabalho é denominada de "A Quinta Disciplina", de Peter Senge. Esta teoria se insere no contexto de um paradigma de teorias administrativas denominado de *Aprendizagem Organizacional*, e provocou grande interesse tanto da área acadêmica quanto de empresas em geral.

Como base para a construção desse modelo foi usado o arcabouço formal para Sistemas Multi-Agentes (SMA) denominado de SMART. Desta forma, o modelo formal para a teoria de Senge é construído num contexto de SMA, ou seja, definindo-se formalmente uma organização composta por agentes e que obedece aos requisitos apresentados pela teoria da Quinta Disciplina.

Análises baseadas nesse modelo evidenciam a importância que algumas características individuais dos agentes apresentam na teoria de Senge, tais como: honestidade e tenacidade.

Adicionalmente, é apresentado um estudo de caso baseado nesse modelo formal envolvendo operações simplificadas de uma lanchonete fictícia, mostrando que diferentes tipos de agentes, tanto humanos quanto artificiais, podem ser modelados.

Este uso de métodos formais para a modelagem de sistemas em geral apresenta novas perspectivas e revela novas potencialidades, ainda não exploradas, em relação ao uso destes métodos.

Abstract

In this thesis, an application of a formal method developed by the area of Software Engineering is presented, in order to build a formal model for a discursive theory, Senge's "Fifth Discipline". This theory is embedded in the context of the Organizational Learning paradigm and has captured great interest from academia and business organizations.

To serve as a foundation for this model, the Multi-Agent Systems (MAS) formal framework named SMART was selected. Therefore, the formal model for Senge's theory is built in a MAS context, ie, by formally defining an organization composed of agents and that follows the requirements presented by the Fifth Discipline theory.

Based on analysis performed on this model, it was possible to conclude that several individual features play an important role in Senge's theory. Thus, agents must be honest, cooperative, tenacious, and trust is fundamental in the agents' interactions.

In addition, a case study based on this model is presented. The study involves simplified operations of a fast food restaurant and shows that different agent types can be modelled: using different abstraction levels it is possible to represent either human or artificial agents.

The use of formal methods for modeling systems in general presents new perspectives and reveals new, not yet explored, potentialities concerning the use of these methods. For example, consistency checking of an organization with regard to a specific organizational theory can be investigated.

CAPÍTULO 1

Introdução

A área de Sistemas Multi-Agentes apresenta uma série de pontos de interesse em comum com a área de Teoria Organizacional e, portanto, pode se beneficiar com o desenvolvimento de conceitos e idéias advindas de Teoria Organizacional[CP98]. Entretanto, tais contribuições podem ser evidenciadas e mais efetivas a partir de um processo de sistematização, interpretação e abstração de conceitos intuitivos e experiências, e comunicação não ambígua e rigorosa do sistema de conhecimentos que compõe uma dada teoria organizacional. Nesta tese, os conceitos apresentados em linguagem textual na teoria da Quinta Disciplina[Sen90] são codificados como estruturas da linguagem Z, respeitando-se as regras impostas por esta linguagem para definir e relacionar estas estruturas.

Este capítulo apresenta as motivações que deram origem a este projeto. Além disso, aqui é apresentada uma perspectiva histórica sucinta da área de Teoria Organizacional e suas diferentes abordagens, dando maior destaque para a abordagem da Aprendizagem Organizacional. Neste capítulo também estão inseridos uma discussão sobre formalização, razões para o uso do arcabouço SMART como base para a construção do modelo formal para a Quinta Disciplina, a relação entre as áreas de Sistemas Multi-Agentes e de Teoria Organizacional, e, finalizando, as motivações para a construção deste modelo e os principais resultados deste projeto.

1.1 Por Que Formalizar

Inicialmente, é interessante considerar o que consta em dicionários da língua portuguesa sobre o vocábulo formalização.

> Formalização[1]: (s.f.) ato de formalizar.
>
> Formalizar[2]: (v.t.) 1. Dar forma a; formar - 2. Executar segundo fórmulas ou regras - 3. Reduzir um sistema de conhecimentos a estruturas formais.
>
> Formalizar[3]: [De formal + -izar.] V. t. d. 1. Dar forma a; formar. 2. Realizar segundo as fórmulas ou formalidades: Após o inquérito, o promotor formalizou a acusação. 3. Executar conforme as regras ou cláusulas. 4. Lóg. Substituir os conceitos e as relações dos sistemas por símbolos sujeitos a regras operatórias bem definidas. V. p. 5. Mostrar-se ofendido ou escandalizado; melindrar-se. 6. Bras. Vestir-se com muito apuro, como quem vai a uma solenidade. 7. Afetar formalidade ou gravidade; assumir uma aparência formal (4): "veste o paletó, ataca os quatro botões, formaliza-se" (Mauro Mota, O Pátio Vermelho, p. 62).

Portanto, conclui-se que a formalização é um processo que envolve regras, estruturas e símbolos.

Em sua coleção de ensaios sobre Lógica, Computação e Filosofia, Wang [Wan90] destaca alguns aspectos importantes sobre o processo de formalização em geral:

- Sistematização

  Utilizando-se métodos formais pode-se construir um sistema de axiomas. Ao permitir uma sistematização, a formalização tem a função de nos permitir discutir com precisão sobre toda uma área de estudos.

[1] Grande Dicionário Larousse Cultural da Língua Portuguesa - Editora Nova Cultural - 1999.
[2] idem.
[3] Dicionário Aurélio Eletrônico Século XXI - versão 3.0 - 1999.

- Comunicação

  O quanto a formalização facilita a comunicação, depende da experiência e formação de quem tem que utilizá-la. Além disso, existem diferentes tipos de demonstrações: as mais detalhadas e rigorosas e as menos rigorosas e completas. Também neste caso, o melhor conteúdo em termos de eficiência de comunicação depende mais da experiência de quem lê a formalização.

- Clareza e consolidação

  A clareza da definição formal de um termo depende do propósito que desejamos que o termo sirva, e também em parte da familiaridade que temos com as noções incluídas em cada definição. A maior vantagem de uma definição claramente expressa de uma noção é sua maior precisão.

  A formalização de um conceito envolve a análise deste conceito e também serve para esclarecer e explicar conceitos.

- Rigor

  Wang afirma que, em certo sentido, formalizar é tornar rigoroso. Entretanto, ele observa que o nível de rigor e detalhamento devem ser adequados aos propósitos do processo de formalização. Assim, o rigor e detalhamento excessivos podem prejudicar a inteligibilidade e leitura de uma formalização.

- Aproximação para a intuição

  A descrição de situações e pensamentos em palavras envolve formalização da intuição. Entretanto, de acordo com Wang, tem-se argumentado que não é possível descrever uma experiência em sua totalidade utilizando um número finito de proposições.

  Ao se fazer aproximações para intuições e experiências, estruturas abstratas e conceitos gerais resultam como produtos de processos de formalização e abstração. Estes produtos podem

servir como ferramentas que auxiliam o raciocínio e a investigação. Desta forma, propriedades que sejam provadas para algum tipo abstrato mais geral serão aplicáveis também a tipos mais específicos, derivados dos genéricos. Entretanto, a caracterização formal de conceitos intuitivos é, freqüentemente, uma tarefa muito difícil.

- Quanto formalizar

  Outra importante questão é o quanto de uma teoria, situação ou conceito pode ou deve ser formalizado. Wang compara este problema ao da apresentação de resultados de cálculos matemáticos com uma maior ou menor precisão: se para um dado experimento só os primeiros dois dígitos decimais são considerados significativos, é inútil apresentar um resultado com 8 dígitos decimais de precisão. Logo, o quanto deve ser formalizado irá depender dos propósitos do modelo formal resultante.

Resumindo, processos de formalização em geral podem permitir a sistematização, interpretação e abstração de conceitos intuitivos e experiências, e a comunicação não ambígua e rigorosa de um sistema de conhecimentos.

1.2 Por Que SMART

A área de Sistemas Multi-Agentes (SMA) tem recebido grande atenção durante a última década. Entretanto, conceitos envolvendo agentes, agência e desenvolvimento baseado em agentes ainda sofrem a falta de uma base conceitual consistente. Para suprir esta necessidade, Luck e D'Inverno [dL01] desenvolveram o arcabouço conceitual SMART.

Este arcabouço foi escolhido para o desenvolvimento desta pesquisa por várias de suas qualidades. O SMART é formal, sem ambigüidades, estruturado e extensível, tendo sido desenvolvido para servir como base para o estudo e construção de diferentes arquiteturas orientadas a agentes. Por exemplo, os trabalhos apresentados em [BCV98, LyLLd01, MLd03] usam este arcabouço como base para o desenvolvimento de seus respectivos modelos.

Um dos principais objetivos do trabalho apresentado nesta tese é produzir um modelo formal para a teoria da Quinta Disciplina num contexto de SMA, ou seja, definindo formalmente uma organização composta por agentes e que atende os requisitos apresentados pela teoria de Senge [Sen90].

Desta forma, o SMART tornou-se uma escolha natural para servir como base para o modelo produzido. Tirando proveito das características do SMART acima mencionadas, foram especificados estruturada e formalmente tipos de agentes que, baseados nos tipos introduzidos no SMART, incorporam capacidades mais sofisticadas de percepção e ação, culminando com a definição do agente que atua na organização aprendiz.

O arcabouço SMART é especificado na linguagem formal Z[Spi92]. Segundo [dL01], a linguagem Z foi escolhida pelos autores por razões que incluem: Z permite desenvolver formalmente especificações de sistemas, tem conexões próximas com a implementação de software e utiliza uma notação simples, que é expressiva, estruturada e bem aceita na comunidade de pesquisas em Inteligência Artificial. Além disso, já existe uma padrão definido pela organização internacional ISO [fSIEC02], ao qual a comunidade de pesquisadores vem se adaptando gradualmente. Outros métodos formais poderiam ter sido usados, como por exemplo VDM [Jon90], CSP [Hoa78], CCS [Mil89], ou lógica modal [Che80]; entretanto nenhum destes métodos oferece a mesma combinação de características mencionadas acima e requeridas pelos autores do SMART para o desenvolvimento deste arcabouço.

1.3 Teoria Organizacional

1.3.1 Introdução

Organizações são instituições que permeiam cada aspecto de nosso cotidiano, de um modo tão natural, que facilmente nos esquecemos que são elas as responsáveis pelos serviços que usamos e produtos que consumimos. Além disso, organizações são formadas por componentes, cujas

complexas interações influenciam seu comportamento, consumo de recursos, produção de bens ou serviços, e sua interação com o ambiente que as envolve.

Há várias definições para organização. Uma delas é: uma organização é uma entidade social, ou seja, composta por agentes; intencionalmente coordenada, portanto gerenciada; que interage entre si e com o ambiente externo, seja individualmente ou em grupos, segundo padrões que determinam a "cultura"da organização; com fronteiras relativamente identificáveis, ou seja, é possível diferenciar os participantes dos não participantes; que serve como um tipo de força de agrupamento, assumindo a forma de contratos implícitos ou explícitos entre os membros individuais e entre os membros e a organização; e que funciona em uma base relativamente contínua, de modo a realizar um conjunto de metas compartilhadas [Rob00].

De acordo com [PCG98] a área de Teoria Organizacional (TO) pode ser caracterizada por dois enfoques: primeiro, como o estudo de como os componentes e fatores acima mencionados interagem, de modo a influenciar o comportamento não somente das organizações, mas também das pessoas e do uso das tecnologias que fazem parte destas organizações; e segundo, como a investigação de princípios gerais organizacionais que são válidos para todas as organizações, independentemente da natureza dos membros das organizações, sejam eles naturais ou artificiais.

A espécie humana se organiza ou trabalha em grupos desde os princípios de seu desenvolvimento, inicialmente em situações de caça. Uma das primeiras conseqüências do trabalho em grupo foi a subdivisão de atividades. As pessoas agrupavam-se para enfrentar de modo mais eficiente os desafios, ambientes complexos e para superar limitações espaço-temporais, de conhecimento e de habilidades.

As principais organizações que estiveram presentes no começo de civilização humana eram basicamente governamentais ou de cunho militar [MSB99]. As raízes da TO remontam à antigüidade, sempre com enfoque em algum aspecto organizacional, por exemplo envolvendo a coordenação de atividades de um grupo, e objetivando algum tipo de ganho em eficiência.

O surgimento das organizações econômicas complexas foi fortemente influenciado pelo início da Era Industrial no Reino Unido. Durante esse período, a disponibilidade de novas fontes de energia

e tecnologias de transportes serviu de apoio para a mudança da produção regional de pequena escala para a distribuição e produção em larga escala. Tal mudança também gerou a necessidade de uma lógica sistemática para o funcionamento organizacional. Além disso, esse período foi influenciado pelos trabalhos de Adam Smith, que, em "A Riqueza das Nações", de 1776, afirmava sua crença na importância da divisão do trabalho e na especialização das tarefas [Chi00].

Desde Adam Smith, a análise do desenho e funcionamento organizacional tem se concentrado em alguma dimensão da organização, como por exemplo: sua cultura, métodos de produção, papéis funcionais e administração. Estas análises são influenciadas por seus correspondentes contextos culturais, o que inclui as tecnologias, o sistema de valores da sociedade, o sistema político, movimentos sociais, entre outros. Desta forma, certas características organizacionais podem receber maior enfoque ou simplesmente ser ignoradas de acordo com mudanças nesse contexto social mais amplo.

Cada enfoque corresponde a uma escola de TO, com seus pressupostos e princípios relativos à sociedade, relevância das pessoas, papel da tecnologia, dentre outros aspectos. Desta forma, cada escola incorpora um conceito de como uma organização deve ser e como gerentes e empregados devem se comportar.

Em resumo, a TO lida com questões associadas à estrutura, projeto e desempenho organizacional: o estudo de questões estruturais envolve aspectos descritivos e o estudo de questões de projeto envolve aspectos prescritivos. A TO descreve como organizações são estruturadas e fornece sugestões sobre como novas organizações podem ser criadas, ou como modificar organizações existentes; tendo sempre como meta o aumento da eficiência.

1.3.2 Visão Geral da TO

De acordo com [Chi00] a atividade da administração é interpretar as metas organizacionais e transformá-las em ação organizacional com eficácia e eficiência; usando para esse fim: planejamento, organização, direção e controle. Como mencionado na seção anterior, o conteúdo e significado

de cada escola da TO varia de acordo com seus diferentes enfoques. Por sua vez, esses enfoques dependem da ênfase associada às seguintes variáveis: pessoas, tarefas, estrutura, ambiente e tecnologia.

Neste trabalho é utilizada a subdivisão da TO em diversas escolas e teorias, conforme apresentado por Chiavenatto em [Chi00]. Uma visão geral desta subdivisão é apresentada na tabela 1.1.

Ênfase	*Escola de TO*	*Enfoques*
Tarefas	Administração Científica	Racionalização operacional do trabalho.
Estrutura	Teoria Clássica	Organização formal.
	Teoria Neoclássica	Princípios gerais da administração, funções do administrador.
	Teoria Estruturalista	Organização formal e informal, análise intra e interorganizacional.
Pessoas	Teoria das Relações Humanas	Organização informal, motivação, liderança, comunicações e dinâmica de grupo.
	Teoria Comportamental	Estilos de administração, teoria das decisões, integração de objetivos organizacionais e individuais.
Ambiente	Teoria Estruturalista	Análise intra-organizacional e ambiental, abordagem de sistema aberto.
	Teoria da Contingência	Análise ambiental, abordagem de sistema aberto.
Tecnologia	Teoria da Contingência	Administração da tecnologia.

Tabela 1.1: Escolas de TO e seus principais enfoques.
Fonte: adaptado de [Chi00].

Administração Clássica

Dentro do enfoque clássico de administração, duas escolas principais se destacaram: a Administração Científica e a Teoria Clássica da Administração.

- Administração Científica

O objetivo principal da Administração Científica é a máxima eficiência operacional, com ênfase na análise e divisão do trabalho. O trabalhador é visto como irresponsável, preguiçoso, negligente, podendo, entretanto, ser treinado. Esta escola teve seu início no começo do século XX com os trabalhos de Frederick Taylor[Tay03].

- Teoria Clássica

  No caso desta teoria, a ênfase está na estrutura organizacional, correspondendo portanto, a um enfoque sintético e global da organização. Esta escola foi criada aproximadamente em 1916, por Henry Fayol[Fay50], e, entre seus princípios pode-se destacar: a divisão do trabalho, autoridade e responsabilidade, disciplina, unidade de comando e direção, centralização e subordinação dos interesses individuais aos interesses organizacionais.

Teoria das Relações Humanas

A Teoria das Relações Humanas tem seu início durante a década de 1930, nos Estados Unidos, como decorrência do desenvolvimento das ciências sociais e da psicologia do trabalho. De acordo com esta teoria, o trabalho é tipicamente uma atividade coletiva na qual o nível de produção é mais influenciado pelas normas do grupo do que por incentivos materiais. O trabalhador é, portanto, membro de um grupo social. A colaboração é um fenômeno social baseado em códigos sociais, convenções e tradições. Normas sociais no grupo funcionam como mecanismos reguladores do comportamento de seus membros. A organização precisa ser capaz de satisfazer as necessidades psicológicas e sociais do trabalhador.

Teoria Neoclássica

Neste enfoque, princípios administrativos advindos da Administração Clássica são revistos com critérios mais flexíveis, com o objetivo de encontrar soluções para problemas administrativos. Entre seus princípios destacam-se: a administração é um processo operacional, envolvendo: organização, planejamento, direção e controle; a administração deve se basear em princípios básicos com valor

preditivo; tais princípios devem ser considerados verdadeiros. Os primeiros trabalhos publicados associados a esse enfoque, datam de 1935, com *"The End of the Economic Man"*, de Peter Drucker.

Teoria Estruturalista

O enfoque estruturalista lida com a relação entre os meios e recursos utilizados e as metas a serem realizadas. Além disso, surge também o interesse nas relações com outras organizações, numa perspectiva interorganizacional. Duas das principais representantes deste enfoque são a Teoria da Burocracia, de Max Weber, e a Teoria Estruturalista. A primeira se desenvolveu em meados de 1940, como conseqüência da parcialidade das teorias Clássica e das Relações Humanas, da necessidade de um modelo mais abrangente de uma organização racional e da crescente complexidade das organizações. A segunda surgiu durante a década de 1950, e envolve a análise das organizações sob uma perspectiva global, incluindo tanto a organização formal quanto a informal. São utilizadas também, tipologias organizacionais para analisar e comparar organizações. Nessa teoria, as organizações são vistas como sistemas abertos, interagindo todo o tempo com o ambiente.

Teoria Comportamental

O enfoque comportamental tem como influências os trabalhos desenvolvidos pela Teoria das Relações Humanas, tendo um momento de grande destaque com a publicação de *"Administrative Behavior"*, de Herbert Simon, em 1945. Além disso, este enfoque também foi influenciado pela psicologia organizacional.

Neste enfoque, estuda-se o comportamento organizacional como função dos processos e dinâmica organizacionais e do comportamento das pessoas na organização.

Uma importante contribuição para este enfoque foi apresentada por Herbert Simon. Segundo Simon, a organização deve ser vista como um sistema de decisões, onde cada pessoa participa de modo racional e consciente, tomando decisões em relação a alternativas comportamentais racionais. O processo de seleção de alternativas pode ser simplesmente uma ação de reflexo condicionado

ou resultado de raciocínio e planejamento. O curso de ação segue um princípio de racionalidade visando realizar uma meta. A decisão é racional se o tomador de decisões escolhe os meios adequados para realizar uma determinada meta. Entretanto, seres humanos têm um comportamento racional de acordo com aspectos das situações das quais estão cientes. Todos os demais aspectos não influenciam o processo de decisão e, portanto, o tomador de decisões tem uma racionalidade limitada.

Teoria de Sistemas

Este enfoque apresenta uma mudança importante nos princípios básicos sobre os quais escolas anteriores de TO se baseavam: reducionismo, pensamento analítico e mecanicismo. Sob a influência da Teoria Geral de Sistemas de Bertalanffy [Chi00], os seguintes princípios embasaram este enfoque: expansionismo, pensamento sintético e teleologia. O expansionismo se refere à noção de que o desempenho de um sistema é função de seu relacionamento com o ambiente onde o sistema se insere, mantendo o foco no todo e correspondendo a uma abordagem sistêmica. Segundo o pensamento sintético, cada fenômeno é parte de um sistema maior e é explicado em função do papel por ele desempenhado neste sistema. Finalmente, segundo a teleologia a relação entre causa e conseqüência é apenas probabilística, ao invés de determinística.

O enfoque sistêmico implica na influência da Cibernética, Matemática (incluindo a Pesquisa Operacional) e Teoria de Sistemas para a TO.

Dentre os vários modelos desenvolvidos nesse enfoque, destaca-se nesta tese o modelo sócio-técnico, que estrutura a organização em dois subsistemas: técnico e social. O primeiro corresponde à eficiência potencial da organização e envolve as tarefas, equipamentos e tecnologia. O segundo envolve as pessoas, suas características físicas e psicológicas, e seus relacionamentos sociais. O subsistema social transforma a eficiência potencial da organização em realizações.

Teoria da Contingência

O enfoque contingencial enfatiza o impacto do ambiente sobre a dinâmica organizacional. Assim características ambientais condicionam características organizacionais, e estas só podem ser analisadas em função daquelas.

Entre os princípios desta abordagem temos:

- Não há um modo único de organizar visando a realização das metas organizacionais em um ambiente complexo.
- A estrutura e funcionamento organizacionais dependem de sua interface com o ambiente.
- Ênfase na influência do ambiente sobre a dinâmica organizacional.
- A organização é um sistema, composto de subsistemas, delimitado por fronteiras que o identificam em relação ao ambiente, que é visto como um supersistema.
- As variáveis ambientais são independentes.
- As técnicas administrativas são as variáveis dependentes.

Novos Enfoques

O surgimento de novos enfoques para a TO é descrito em [Chi00]. Estas novas abordagens são uma conseqüência de mudanças profundas, que, em conjunto são denominadas de *Era da Informação*. Nessa nova era, as mudanças são mais freqüentes, intensas e descontínuas, dificultando interpretações das relações de causa e efeito entre os eventos. Ainda em 1968, Peter Drucker, em seu trabalho: *The Age of Discontinuity*, destacou a importância do conhecimento para a sociedade como um todo, e para o processo produtivo em particular, introduzindo, além disso, os termos: sociedade do conhecimento (*knowledge society*) e trabalhador do conhecimento (*knowledge worker*) [Fla99].

Desta forma, a TO requer novas abordagens para lidar com esses novos problemas e a crescente complexidade do ambiente econômico, ganhando importância nesse momento o capital intelectual.

De acordo com [Chi00] entre as novas abordagens surgidas, destacam-se:

- Melhoria Contínua

  É uma abordagem incremental e participativa, envolvendo mudanças contínuas que se processam de modo suave, partindo da base da organização para a cúpula. Seus princípios incluem: ênfase no cliente, discussões francas, criação de equipes de trabalho, desenvolvimento da autodisciplina e dos relacionamentos interpessoais, treinamento e acesso à informação para todos os empregados.

- Qualidade Total

  Esta é também uma abordagem incremental, onde o conceito de qualidade é aplicado para toda a organização. Neste caso, todos são responsáveis pela manutenção de padrões de qualidade, permitindo o emprego de um tipo de controle coletivo e descentralizado.

- Reengenharia

  Segundo esta abordagem, os processos organizacionais devem fundamentar o formato da organização. Assim, os três componentes da reengenharia são: os processos, as pessoas e a tecnologia da informação (TI). Em geral, a reengenharia privilegia equipes, ao invés de departamentos, visando reduzir a estrutura organizacional hierarquizada.

- Gestão do Conhecimento

  O conhecimento é, provavelmente, o fator produtivo mais importante nas empresas modernas. A utilização de todo o conhecimento disponível em uma organização torna-se essencial para que esta possa alcançar sucesso em suas atividades. Entretanto, este importante patrimônio é ainda freqüentemente negligenciado, permanecendo codificado e camuflado em

serviços, sistemas, documentos e processos organizacionais, ou ainda, armazenado nas mentes dos empregados.

A abordagem da Gestão do Conhecimento tenta enfrentar este problema através do uso de ferramentas e métodos visando a aquisição, conservação, proteção e o uso efetivo do conhecimento em uma organização.

- Aprendizagem Organizacional

  Esta abordagem enfatiza os processos de aprendizado; inovação; e criação, armazenamento, manipulação e transferência de conhecimento. Desta forma, espera-se aprimorar o comportamento organizacional, produzindo melhor desempenho e estabilidade para a organização. Neste enfoque, maior atenção é dada aos ativos intangíveis, que podem conferir vantagem competitiva para as organizações que os gerenciem com sucesso.

  Esta abordagem será apresentada com maiores detalhes na próxima seção, uma vez que este é o contexto onde se insere a teoria da Quinta Disciplina.

1.4 Aprendizagem Organizacional e a Quinta Disciplina

Na seção anterior a abordagem da Aprendizagem Organizacional foi brevemente apresentada como uma dentre as mais recentes abordagens da TO. Nesta seção, uma visão geral desta abordagem é apresentada com base nas perspectivas de diferentes autores da área.

Dentre as diferentes visões deste enfoque, deve-se mencionar o trabalho de [Arg77] que define que a "aprendizagem organizacional é um processo de detecção e correção de erros." [4]. Já [FL85] afirma que "aprendizagem organizacional significa o processo de melhoria das ações por meio de melhor conhecimento e compreensão"[5]. Por sua vez, [Gar93] descreve uma organização aprendiz

[4]Traduzido pelo autor.

[5]Traduzido pelo autor.

como "uma organização que tem competência para criar, adquirir e transferir conhecimento, e também de modificar seu comportamento para refletir os novos conhecimentos e insights[6]"[7].

Uma das primeiras referências à expressão "aprendizagem organizacional" é apresentada por [CD65], onde a aprendizagem organizacional é vista como o produto de interações entre adaptações nos níveis individual ou de subgrupo e organizacional. As adaptações ocorrem em conseqüência de três tipos de estresse: de desconforto, de desempenho e disjuntivo. O estresse de desconforto resulta da complexidade do ambiente e do esforço da equipe em compreendê-lo. O estresse de desempenho está associado às pressões organizacionais para que as ações dos indivíduos sejam bem sucedidas. Finalmente, o estresse disjuntivo resulta de crescentes graus de divergências e conflitos produzidos pelos diferentes comportamentos de indivíduos e grupos. Estresses de desconforto e de desempenho causam adaptações individuais e de subgrupo. A aprendizagem organizacional resulta tanto de influências do estresse de desempenho quanto do estresse disjuntivo.

De acordo com [Kim93] toda organização aprende, quer tenha deliberadamente planejado isto ou não. Para este autor, a aprendizagem organizacional ocorre por meio da aprendizagem individual. Como conseqüência, o autor propõe um modelo que conecta a aprendizagem individual à aprendizagem organizacional por meio de modelos mentais.

A aprendizagem individual envolve dois níveis: operacional e conceitual. O primeiro se refere ao aprendizado das etapas associadas à realização de uma determinada tarefa e se reflete em seqüências rotineiras de atividades. Tanto a aprendizagem operacional altera as rotinas quanto as rotinas influenciam a aprendizagem. Por outro lado, o nível conceitual envolve a reflexão sobre o modo como as atividades são realizadas, podendo levar à revisão ou eliminação destas. Portanto, em [Kim93] o modelo de aprendizagem individual envolve um ciclo de aprendizagem conceitual e operacional que recebe informações e é informado por modelos mentais.

Os ciclos de aprendizagem individual influenciam os modelos mentais compartilhados da organização e, como conseqüência, influem na aprendizagem organizacional.

[6]Este verbete consta do Dicionário Aurélio Eletrônico - Versão 3.0.

[7]Traduzido pelo autor.

O trabalho de [CLW99] destaca que a aprendizagem organizacional envolve uma tensão entre a assimilação de um novo aprendizado e o uso do que já foi aprendido, denominada de "tensão de renovação estratégica". Nesse artigo, é apresentado um arcabouço para a aprendizagem organizacional que parte do pressuposto que a renovação de toda a organização deva ser o fenômeno básico a ser estudado. Este arcabouço considera quatro processos sociais e psicológicos: intuição, interpretação, integração e institucionalização; denominados de *4I's*. Além disso, há três níveis de aprendizagem organizacional: individual, da equipe e organizacional. Neste arcabouço, os *4I's* estão relacionados em processos de alimentação e retroalimentação através dos níveis de aprendizagem.

Em [DTC97] define-se que uma organização aprendiz é aquela que desenvolve competências de modo a, primeiro, manter e melhorar o desempenho atual e, segundo, modificar a organização para garantir o desempenho futuro. Assim, há a necessidade de se desenvolver e manter a capacidade total de gerenciamento de uma organização, incluindo o desenvolvimento das competências corporativas e pessoais dos gerentes. Na verdade, este desenvolvimento deve se tornar uma competência da organização, de modo que seja sempre aplicado e incorporado aos sistemas, estruturas, valores e políticas organizacionais.

Um dos artigos mais influentes dessa área é o de Nonaka [Non91], que apresenta como organizações japonesas gerenciam seu conhecimento e criatividade. De acordo com Nonaka, as organizações bem sucedidas são aquelas que consistentemente criam novos conhecimentos, difundem estes conhecimentos amplamente por toda a organização e rapidamente os incorporam em novas tecnologias e produtos. Estas atividades definem uma organização cujo negócio é a inovação contínua.

Nonaka define dois tipos de conhecimento: explícito e tácito. O primeiro é formal e sistemático, podendo ser compartilhado e comunicado pois se encontra codificado em normas, especificações ou fórmulas. O segundo apresenta duas dimensões: uma envolve habilidades técnicas e experiência em atividades profissionais; a outra, cognitiva, consiste de crenças e modelos mentais que influenciam o modo como as pessoas percebem o ambiente. Ambas as dimensões do conhecimento

tácito são de difícil articulação e, portanto, oferecem maior dificuldade para a sua comunicação e compartilhamento. Surgem portanto, quatro modos de gerar conhecimento: de tácito para tácito, de tácito para explícito, de explícito para explícito e de explícito para tácito. O autor define, em seguida, uma interação dinâmica entre estes padrões, ou ainda, uma espiral de conhecimento que envolve a socialização, articulação, combinação e internalização que correspondem, respectivamente, a cada um dos modos acima citados. A organização criadora de conhecimento mantém este tipo de espiral em atividade constante. Adicionalmente, segundo esse autor, as etapas de articulação e internalização são críticas nesta espiral pois dependem do comprometimento de cada indivíduo para a sua realização.

Finalmente, uma das mais recentes e importantes contribuições para a escola da Aprendizagem Organizacional é apresentada por Senge em [Sen90], denominada de "A Quinta Disciplina". Nesta tese, essa teoria é também mencionada como teoria LO[8].

De acordo com Senge, a organização aprendiz é aquela que está "continuamente expandindo sua capacidade de criar seu futuro" [9].

Em sua teoria, Senge afirma que o ciclo de vida da maioria das organizações é relativamente curto em conseqüência da incapacidade destas organizações de perceber alguns tipos de ameaças às suas existências. Tal incapacidade está associada a deficiências de aprendizagem.

Para superar estas deficiências, Senge propõe a teoria da Quinta Disciplina, que se baseia em cinco disciplinas: pensamento sistêmico, domínio pessoal, modelos mentais, visão compartilhada e aprendizagem em equipe.

A organização que desenvolve esta teoria combina a aprendizagem adaptativa, dirigida à sobrevivência da organização, com a aprendizagem generativa, que aumenta as capacidades criativas da organização.

Assim, a Quinta Disciplina é uma teoria moderna, relevante para a área de TO, bem sucedida, estruturada em cinco disciplinas, e que focaliza tanto aspectos individuais quanto coletivos,

[8]LO: *Learning Organization.*

[9]Tradução do autor.

apresentando uma visão mais abrangente que as apresentadas nos outros enfoques citados acima. Portanto, a Quinta Disciplina se apresenta como uma teoria interessante e uma escolha adequada para o desenvolvimento da pesquisa reportada nesta tese.

Uma visão mais detalhada da teoria de Senge é apresentada no capítulo 2 desta tese.

1.5 Teoria Organizacional e Sistemas Multi-Agentes

A pesquisa em modelagem computacional de organizações apresenta crescente vigor e maturidade. Tal fato decorre do aumento na sofisticação das ferramentas utilizadas e, principalmente, da integração destas a conceitos derivados de estudos em Ciências Sociais aplicados às organizações humanas.

Nesse sentido, de acordo com [PCG98], um desenvolvimento importante é o crescente interesse que se observa da parte de pesquisadores da área de Inteligência Artificial Distribuída (IAD) por questões organizacionais. Esse fato é conseqüência da identificação da grande similaridade entre os problemas enfrentados pelas organizações humanas e os problemas enfrentados tanto em Resolução Distribuída de Problemas (RDP) quanto em Sistemas Multi-Agentes (SMA), duas sub-áreas de IAD. Estas similaridades incluem os seguintes aspectos:

- Ambas as áreas lidam com coleções de componentes inteligentes distribuídos, atuando em paralelo, visando a resolução de problemas.
- As propriedades de grupos humanos e sistemas de IAD não são representadas ou derivadas exclusivamente a partir das propriedades individuais dos seus membros ou agentes.
- Ambas as áreas enfrentam os mesmos problemas de alocação de recursos, tarefas e informações para conjuntos de componentes inteligentes.
- Considerando-se a tomada de decisão como um comportamento cognitivo e, portanto, redutível a um comportamento de resolução de problemas, a tomada de decisão em equipe pode

ser vista como uma resolução distribuída de problemas.

Desta forma, sistemas de IAD em geral, e de SMA em particular, podem servir como modelos de grupos humanos, e, nesse sentido, as áreas de IAD e SMA podem ser úteis como ramos experimentais de Ciências Organizacionais. Por outro lado, os desenvolvimentos mencionados acima formam um ciclo, no qual idéias advindas de organizações humanas servem como analogias que inspiram desenvolvimentos em SMA. Estes, por sua vez, transformam-se em ferramentas úteis à construção de modelos de organizações [PCG98].

O contexto de Sistemas Multi-Agentes é, portanto, adequado para o desenvolvimento de um modelo formal para a teoria da Quinta Disciplina.

1.6 Trabalhos Relacionados - Apresentação Inicial

Uma série de pesquisas e artigos reforçam esta perspectiva de que contribuições da área de Teorias Organizacionais podem ser úteis para a área de Sistemas Multi-Agentes e vice-versa. Nesta seção, tais trabalhos são apenas apresentados. Uma comparação destes trabalhos com a pesquisa reportada nesta tese é apresentada no capítulo 7.

Em [KP99] e [MH96], são apresentadas formalizações da teoria "Organization in Action"(OA) [Tho67]. Em ambos os trabalhos os autores apresentam uma revisão de parte de uma teoria discursiva usando métodos formais. No caso de [KP99], lógica de predicados de primeira ordem é usada para estudar a estrutura de argumentação que serve de embasamento para as proposições da teoria OA. Por outro lado, em [MH96] é usada uma lógica modal multi-agentes desenvolvida pelos próprios autores, com objetivos similares aos citados acima e com o objetivo adicional de investigar a utilidade e capacidade expressiva desse tipo de lógica para a formalização de uma teoria discursiva como a OA.

Adicionalmente, em [PCG98] e [CP94], diversos trabalhos relacionados à simulação computadorizada de organizações são apresentados. Todos esses trabalhos envolvem a construção de

modelos computacionais relacionados a organizações e teorias organizacionais. A seguir, alguns desses trabalhos são citados.

O problema da modelagem da tomada de decisão em equipe é o foco do trabalho apresentado em [KWW98]. Nesse trabalho uma equipe corresponde a um grupo que compartilha uma meta ou propósito. A equipe é composta por membros altamente diferenciados e interdependentes com liderança compartilhada, responsabilidades e atribuições individuais e coletivas. No processo de tomada de decisão em equipe são utilizadas as múltiplas inteligências de seus membros, freqüentemente envolvendo a divisão do problema em subproblemas menores que serão posteriormente integrados para a obtenção da solução da equipe.

O trabalho de [Lin98] se concentra na compreensão e no projeto de organizações que exigem alta confiabilidade. A perspectiva do autor é a de que modelos de simulação são ferramentas para examinar diversas possibilidades de estruturações de organizações. O autor examina os efeitos de condições externas na performance de tomada de decisão em situações onde é importante evitar enganos severos.

O dilema social de gerar cooperação voluntária entre indivíduos confrontados com opções conflitantes de tempo e esforço é o foco do artigo apresentado em [HG98]. O dilema, nesse caso, é que o indivíduo pode ajudar a criar um bem público, compartilhado por muitos, mas onde os custos individuais desestimulam tal esforço. Alternativamente, o indivíduo pode optar por tirar proveito dos esforços dos outros. Esta questão é fundamental para o estudo do comportamento cooperativo em organizações.

No trabalho de [CP98], é criada e examinada uma nova entidade: o *WebBot*. Um *WebBot* é um programa de computador que opera de modo autônomo para realizar uma tarefa, atuando como um conselheiro intelectual e assistente a um parceiro humano (ou a outros *WebBots*). Os *WebBots* são membros da organização que agem, se comunicam, têm memória e requerem coordenação. A questão específica investigada neste artigo envolve os efeitos da honestidade do *WebBot* em relação ao comportamento organizacional individual e coletivo. Nos testes foram construídas organizações onde os *WebBots* eram todos honestos ou todos desonestos. Os resultados mostraram que, ao longo

do tempo, *WebBots* honestos fazem mais perguntas enquanto aprendem mais, enquanto *WebBots* desonestos passam a perguntar menos.

Para finalizar as citações de artigos presentes em [PCG98] é apresentado o trabalho de [SD98]. A construção e a capacidade de reconstrução de organizações que incluem humanos ou agentes computacionais é o ponto central desse trabalho. Duas perspectivas advindas de teorias organizacionais são utilizadas para definir um arcabouço que descreve o problema da estruturação da organização: a teoria da contingência, que destaca a importância da correspondência organização-ambiente; e a teoria sócio-técnica, segundo a qual organizações são sistemas tanto sociais quanto técnicos.

O objetivo do trabalho apresentado em [YS99] é explorar um arcabouço conceitual orientado a agentes e baseado em papéis para modelagem de *workflow*[10]. Nesse trabalho processos de negócios são vistos como uma coleção de agentes autônomos, solucionadores de problemas, que interagem com outros quando interdependências ocorrem. Além disso, o *workflow* é modelado como um conjunto de papéis relacionados. Papéis são definidos em termos de metas, qualificações, obrigações, permissões, protocolos, etc.. São adotados protocolos para governar as interações entre papéis. Aos agentes são atribuídos papéis baseados na avaliação de qualificações e capacidades.

No trabalho apresentado em [K. 99] é relatado um estudo sobre a aprendizagem em Sistemas Multi-Agentes. Esse estudo se baseou no enfoque de [Arg77] para a abordagem da Aprendizagem Organizacional e considera quatro tipos de aprendizagem. O primeiro é denominado de laço simples individual[11] e incrementa o desempenho dentro do escopo de uma norma individual. O segundo é denominado de laço duplo individual[12] e incrementa o desempenho mediante mudança de uma norma individual. O terceiro é denominado de laço simples organizacional e incrementa o desempenho dentro do escopo de uma norma organizacional. Finalmente, o quarto é denominado

[10]Este termo não foi traduzido por ser de uso comum na área de computação. No contexto do trabalho citado, a tarefa de *workflow* consiste em descrever a coordenação e desempenho do trabalho realizado em uma organização.

[11]Tradução do autor para o termo: *single-loops*.

[12]Tradução do autor para o termo: *double-loops*.

de laço duplo organizacional e incrementa o desempenho mediante mudança de uma norma organizacional. Para permitir a implementação do sistema, normas individuais são implementadas pelo conhecimento dos indivíduos e normas organizacionais são implementadas pelo conhecimento organizacional.

Em [HSB02], é apresentado um modelo para especificação de organizações de Sistemas Multi-Agentes que se concentra em aspectos funcionais, estruturais e deônticos. A estrutura envolve os conceitos de papel, relações entre papéis e grupos. Quanto à dimensão funcional, esta inclui os conceitos de missões e planos globais, estruturados em um tipo de árvore de decomposição de metas [13]. No modelo apresentado, as dimensões funcionais e estruturais são independentes de modo a aumentar a flexibilidade do Sistema Multi-Agentes. Assim, a dimensão funcional pode ser alterada sem que haja a necessidade de se modificar a estrutura da organização. Apenas a dimensão deôntica, relativa a permissões e obrigações, tem que ser adaptada de acordo com as mudanças desejadas.

1.7 Este Trabalho: A Quinta Disciplina em SMART

Fica evidenciado pelo exposto na seção 1.4 que não há consenso na área de Aprendizagem Organizacional sobre o que efetivamente constitui uma "Organização que Aprende" [Gar93, CLW99]. Existem várias definições e diferentes enfoques. Dentre esses, a teoria da Quinta Disciplina tornou-se uma das mais populares e bem sucedidas nessa área.

Por outro lado, a área de Sistemas Multi-Agentes apresenta uma série de pontos de interesse comum com a área de Teoria Organizacional, conforme exposto na seção 1.5, e pode se beneficiar com o desenvolvimento de conceitos e idéias advindas de Teoria Organizacional.

Além disso, conforme discussão apresentada na seção 1.1, processos de formalização em geral podem permitir a sistematização, interpretação e abstração de conceitos intuitivos e experiências, e a comunicação não ambígua e rigorosa de um sistema de conhecimentos. Nesta pesquisa,

[13] Esta estrutura é denominada de *Social Scheme* pelos autores do artigo.

especificamente, este sistema de conhecimentos é a teoria da Quinta Disciplina.

Neste trabalho, os conceitos apresentados em linguagem textual por Senge são codificados como estruturas da linguagem Z, respeitando-se as regras impostas por esta linguagem para definir e relacionar estas estruturas.

O resultado deste trabalho, um modelo formal da teoria LO, atende a dois propósitos:

- Validação

  O modelo pode ser utilizado para se efetuar a validação da aderência de uma dada organização aos preceitos da teoria de Senge.

- Modelar Diferentes Níveis de Automação da Organização

  Usando o mesmo modelo, tanto agentes humanos quanto agentes artificiais, tais como sistemas computacionais, podem ser representados. De um ponto de vista gerencial, a visão da organização, por intermédio do modelo, será a mesma.

A pesquisa e projeto reportados nesta tese apresentam como suas principais contribuições:

- O modelo formal para a Quinta Disciplina, com as características citadas de precisão e rigor.
- A utilização deste modelo como ferramenta de apoio para a validação e modelagem citados acima.
- Discussão de propriedades observadas no modelo formal e que devem ser consideradas na implementação da teoria de Senge em organizações humanas, artificiais ou híbridas.
- Um estudo de caso baseado na técnica de animação da especificação formal.

1.8 Visão Geral deste Trabalho

Esta tese está estruturada da seguinte forma:

O capítulo 2 apresenta uma visão geral da teoria de Senge, com o detalhamento das disciplinas de pensamento sistêmico, domínio pessoal, modelos mentais, visão compartilhada e aprendizagem em equipe.

Os assuntos do capítulo 3 incluem: uma descrição sucinta da área de Sistemas Multi-Agentes, descrevendo seus objetivos, aplicações, problemas enfrentados, e sua relação com o arcabouço SMART. Adicionalmente, um breve tutorial da linguagem Z é apresentado na última seção deste capítulo, incluindo informações básicas sobre essa notação, que devem servir de apoio para a leitura do modelo formal apresentado nesta tese.

No capítulo 4 é apresentado, resumidamente, o arcabouço SMART, incluindo suas motivações e conceitos básicos. Além disso, a estrutura principal do arcabouço é detalhada, com a apresentação das especificações formais para entidades, objetos, agentes e agentes autônomos.

O modelo formal produzido para a Quinta Disciplina é descrito no capítulo 5. São apresentados vários refinamentos para os tipos de agentes. O processo é incremental e apresenta inicialmente tipos menos sofisticados de agentes, como por exemplo, agentes com autonomia e memória, agentes que podem lidar com planos, agentes que participam de organizações, e finalmente, agentes que são capazes de desenvolver as disciplinas de Senge e participar de uma organização aprendiz, de acordo com os requisitos definidos pela teoria da Quinta Disciplina.

Um estudo de caso é apresentado no capítulo 6. Esse estudo é baseado em um nível de abstração mais alto em relação ao modelo formal produzido no capítulo 5. Nesse estudo é utilizada uma técnica de animação do modelo formal. O modelo formal descreve um modelo simplificado para uma lanchonte fictícia e a animação envolve alguns processos básicos desse estabelecimento, tais como, o atendimento ao cliente, serviço de cozinha e gerenciamento da loja. Esse estudo de caso mostra que esse modelo pode ser útil para a validação da aderência de uma organização em relação a uma teoria organizacional, e também para modelar diferentes tipos de agentes: humanos ou artificiais.

Finalmente, no capítulo 7 são apresentadas as discussões e comentários finais, conclusões e considerações sobre trabalhos futuros.

CAPÍTULO 2

Visão Geral da Quinta Disciplina

Neste capítulo é apresentada de forma resumida a teoria da Quinta Disciplina [Sen90, Sen02], incluindo seus conceitos básicos e suas hipóteses subjacentes. Estes conceitos e hipóteses serão considerados posteriormente para o desenvolvimento do modelo formal descrito no capítulo 5.

2.1 Introdução

O livro básico adotado nesta pesquisa para o estudo da teoria de Senge é "The Fifth Discipline - The Art and Practice of the Learning Organization" [Sen90], embora também exista uma versão traduzida para o português [Sen02]. Outro livro utilizado como material de apoio para esta pesquisa é "The Fifth Discipline Fieldbook - Strategies and Tools for Building a Learning Organization" [SKR+94]. A seguir, é apresentada uma breve descrição do livro básico adotado neste trabalho.

O livro de Senge é dividido em cinco partes. Na primeira parte o autor discorre sobre como as ações do indivíduo criam a sua realidade e como esse mesmo indivíduo pode mudá-la, uma vez que ele reconheça o seu papel na sua construção. Esta parte, por sua vez, é subdividida em três capítulos. No primeiro capítulo Senge introduz as cinco disciplinas que formam o núcleo de sua teoria, e discute sobre a importância de se manter uma visão holística da realidade. No segundo capítulo são apresentadas deficiências de aprendizagem que estão presentes na maioria das organizações e que são, segundo Senge, a causa para o reduzido tempo de existência da maioria

das organizações. No capítulo 3 são apresentados, através de um exemplo, diversos problemas associados a visões fragmentadas da realidade. O objetivo é demonstrar a importância de uma percepção sistêmica em contraposição ao enfoque em eventos localizados.

A segunda parte do livro se concentra na apresentação da disciplina do pensamento sistêmico: a quinta disciplina. Esta parte compreende do capítulo 4 ao capítulo 8. No capítulo 4 Senge discorre sobre as "leis" da quinta disciplina. Estas leis correspondem a algumas regras gerais práticas envolvendo a aplicação desta disciplina. Tais regras incluem: "Os problemas de hoje vêm das 'soluções' de ontem", "Quanto mais você empurra, mais o sistema empurra de volta", entre outras. O capítulo 5 apresenta a visão sistêmica e a necessidade de se enxergar círculos de causalidade com a introdução dos conceitos de retroalimentação de reforço, retroalimentação de equilíbrio e retardos. O capítulo 6 introduz os arquétipos sistêmicos, que são estruturas genéricas que servem de apoio para o pensamento sistêmico. No capítulo 7 é introduzido o conceito de "princípio de alavancagem". O pensamento sistêmico deve permitir ao indivíduo identificar o ponto na estrutura sistêmica onde alterações e ações resultam em melhorias expressivas de longa duração. No capítulo 8 Senge destaca a importância de se enxergar as estruturas sistêmicas em atividade, mesmo em situações que envolvam grande complexidade.

Na terceira parte são apresentadas as demais disciplinas: o domínio pessoal é o tema do capítulo 9, os modelos mentais são apresentados no capítulo 10, no capítulo 11 o foco é na visão compartilhada, e, finalmente, no capítulo 12 a disciplina de aprendizagem em equipe é apresentada.

Na quarta parte do livro são apresentados protótipos de organizações, que devem ser desenvolvidos e testados. Estes protótipos apresentam questões e problemas tipicamente encontrados em organizações: falta de escrúpulos e dissimulação, coordenação e controle em uma organização descentralizada, alocação de tempo para aprendizado por parte dos gerentes, entre outros.

Na quinta parte Senge discorre sobre a perspectiva de uma nova disciplina, que talvez surja como conseqüência da aplicação em larga escala dos preceitos apresentados em sua teoria. Uma outra conseqüência dessa aplicação pode ser o desenvolvimento de novas formas de pensamento. Finalmente, Senge conclui o livro reiterando a importância de uma visão holística da realidade.

2.2 A Quinta Disciplina

Peter Senge afirma que o ciclo de vida da maioria das organizações é relativamente curto. Mencionando uma pesquisa realizada pela Royal Dutch/Shell em 1983, o autor relata que um terço das empresas mencionadas na publicação Fortune "500", edição de 1970, não existia mais quando da publicação da pesquisa. Adicionalmente, a mesma pesquisa indicava que o tempo médio de vida das maiores empresas industriais era de menos de 40 anos. Comenta ainda o autor que, embora na maioria das empresas que fecham suas portas haja evidências abundantes de sérios problemas, estas organizações não conseguem reconhecer as ameaças iminentes, perceber suas implicações ou propor alternativas. Na opinião de Senge estas dificuldades estão relacionadas a diversas deficiências de aprendizagem:

- "Eu sou minha posição".

  Quando os indivíduos concentram-se somente em suas posições, ou papéis, na organização, perdem o senso de responsabilidade pelos resultados obtidos por intermédio da interação das diversas posições.

- "O inimigo está lá fora".

  Em geral, esta é uma visão incompleta de uma situação, pois "aqui" (dentro da organização) e "lá fora" são partes de um sistema único.

- "A ilusão de estar no controle".

  Freqüentemente, posturas pró-ativas são, na verdade, disfarces para atitudes reativas, tentativas de enfrentar o "inimigo lá fora". Atitudes realmente pró-ativas são conseqüência de uma visão clara de como o indivíduo contribui para os seus próprios problemas.

- "Fixação em eventos".

  Há a necessidade de se visualizar padrões de longo prazo e toda a rede de conexões entre causas e efeitos, e não se fixar apenas em eventos.

- "Ameaças gradativas".

  As organizações, em geral, não estão preparadas para enfrentar ameaças gradativas que ponham em perigo suas existências. É importante que sejam observados tanto os processos gradativos de mudanças quanto os mais rápidos.

- "A ilusão de aprender a partir da experiência".

  Embora a experiência direta seja uma fonte poderosa para o aprendizado, ao se lidar com problemas organizacionais, normalmente não é possível associar as conseqüências de tomadas de decisões importantes com suas verdadeiras causas.

- "O mito da equipe de gerência".

  Esta equipe é composta por gerentes de diferentes funções e áreas de especialização da organização e, supostamente, deveria fazer frente a todas as deficiências mencionadas. Entretanto, normalmente tais equipes desperdiçam energia em disputas políticas enquanto tentam passar uma aparência de equipe coesa.

2.2.1 A Estrutura Influencia o Comportamento

No contexto da teoria de Senge, o termo estrutura se refere aos inter-relacionamentos entre variáveis chave, como por exemplo, recursos naturais, população e produção de alimentos. Ou seja, trata-se da estrutura sistêmica. Segundo Senge, pessoas diferentes atuando na mesma estrutura sistêmica, tendem a produzir resultados qualitativamente similares, pois o comportamento observado é resultante da influência da estrutura sistêmica.

Para que o comportamento individual possa apresentar um desempenho que supere (ou que pelo menos gerencie de modo eficiente) as limitações e problemas impostos pela estrutura sistêmica, é necessário que os indivíduos redefinam o escopo de sua influência sobre o sistema como um todo, passando a observar como suas decisões afetam os demais, ao longo dos relacionamentos entre as variáveis chave do sistema.

Normalmente, os comportamentos individuais resultam de uma abordagem focada em compreender eventos. Como conseqüência desta abordagem, a postura apresentada pelo indivíduo é reativa.

Em uma postura sistêmica, ao contrário, há múltiplos níveis de compreensão, conforme mostra a figura 2.1.

Figura 2.1: Níveis de compreensão - uma perspectiva sistêmica.

Fonte: adaptado de [Sen90, p. 52].

A compreensão de padrões de comportamento leva à visualização de tendências de longo prazo. Desta forma, este nível de compreensão pode sugerir ao indivíduo como responder a longo prazo às mudanças nas tendências.

O nível estrutural, por sua vez, é o mais importante e poderoso. Este nível se concentra nas causas subjacentes do comportamento. Com base neste foco, padrões de comportamento podem ser alterados.

Segundo Senge, as deficiências citadas acima permeiam a história e a cultura humana. Propõe, então, o autor uma teoria que deve funcionar como um antídoto contra essas deficiências: a Quinta Disciplina.

A teoria da Quinta Disciplina, como o próprio nome indica, se baseia em cinco disciplinas:

pensamento sistêmico, domínio pessoal, modelos mentais, visão compartilhada e aprendizagem em equipe; o que permite que a perspectiva sistêmica possa prevalecer às posturas meramente reativas.

Assim, a "Organização que Aprende" é aquela que segue os preceitos da Quinta Disciplina e, segundo Senge, está continuamente expandindo sua capacidade para criar o seu futuro. Uma tal organização não se satisfaz apenas com a "aprendizagem voltada a sobrevivência", denominada pelo autor de aprendizagem adaptativa. Mais do que isso, esta organização combina aprendizagem adaptativa com aprendizagem generativa, que é a aprendizagem que incrementa a capacidade criativa.

2.2.2 Pensamento Sistêmico

O conceito principal desta disciplina é que o indivíduo só pode compreender os padrões de eventos envolvidos em sistemas complexos a partir de uma visão global.

Os negócios são também sistemas, em que ações inter-relacionadas levam algum tempo para desenvolver totalmente seus efeitos. Um indivíduo imerso em todo esse emaranhado tem dificuldades para observar com nitidez todo o padrão de mudanças. Como conseqüência, em geral, ele tende a se fixar em aspectos isolados do problema para tentar encontrar uma solução que, na verdade, deveria ser de âmbito global.

O pensamento sistêmico é um arcabouço conceitual, um corpo de conhecimento e de ferramentas desenvolvidas ao longo dos últimos 50 anos, que engloba a Cibernética e a Teoria do Caos, entre outros. O pensamento sistêmico tem por objetivo tornar claramente visíveis os padrões complexos, auxiliando o indivíduo a mudar as situações efetivamente e com esforço mínimo. Para tanto, ele deve encontrar o que Senge denomina de "pontos de alavancagem do sistema".

De acordo com Senge a forma de pensamento sistêmico denominada de "dinâmica de sistemas", que vem sendo desenvolvida por Jay Forrester e colegas no *Massachusetts Institute of Technology* ao longo de 40 anos de trabalho, mostrou-se bastante útil como uma linguagem para descrever como

implementar mudanças bem sucedidas em organizações. Desta forma, as ferramentas de apoio ao pensamento sistêmico utilizadas por Senge, como por exemplo: "conexões e laços causais", "arquétipos" e modelos computacionais, têm suas raízes na compreensão de como processos complexos de retroalimentação podem gerar padrões problemáticos de comportamento em organizações.

A base do pensamento sistêmico são os retardos ou defasagens (*delays*) e dois tipos de processos de retroalimentação: reforço e equilíbrio.

O processo de reforço funciona como um motor de crescimento (ou declínio), com um efeito de "bola de neve". Na figura 2.2 vemos o processo de reforço para um bom produto: mais vendas levam a mais clientes satisfeitos, que leva a maior divulgação boca-a-boca, resultando em mais vendas, e assim sucessivamente.

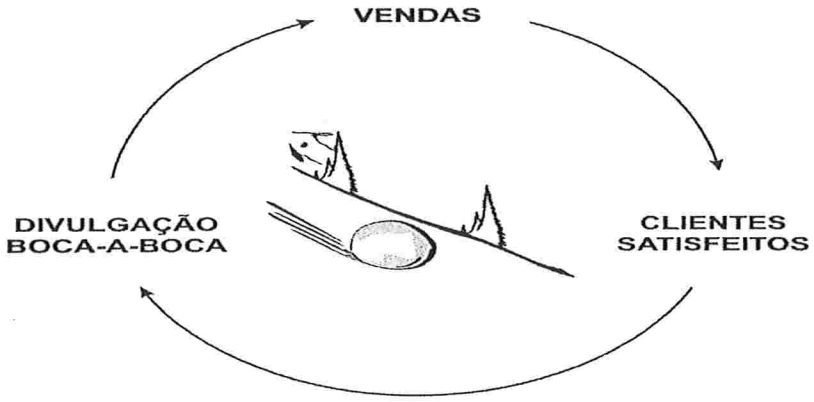


Figura 2.2: Processo de reforço.
Fonte: adaptado de [Sen90, p. 81].

Por outro lado, um processo de equilíbrio se encontra em ação sempre que existe um comportamento dirigido por um objetivo, explícito ou implícito. Na figura 2.3 pode-se observar um processo de equilíbrio: assim que o indivíduo percebe uma diferença entre a temperatura atual de seu corpo e a temperatura desejada, ajusta seu vestuário (usando roupas mais leves, ou mais pesadas). A temperatura observada do corpo estará mais próxima da desejada. Ele continua o processo tentando minimizar a diferença, mesmo que seu objetivo mude com o tempo.

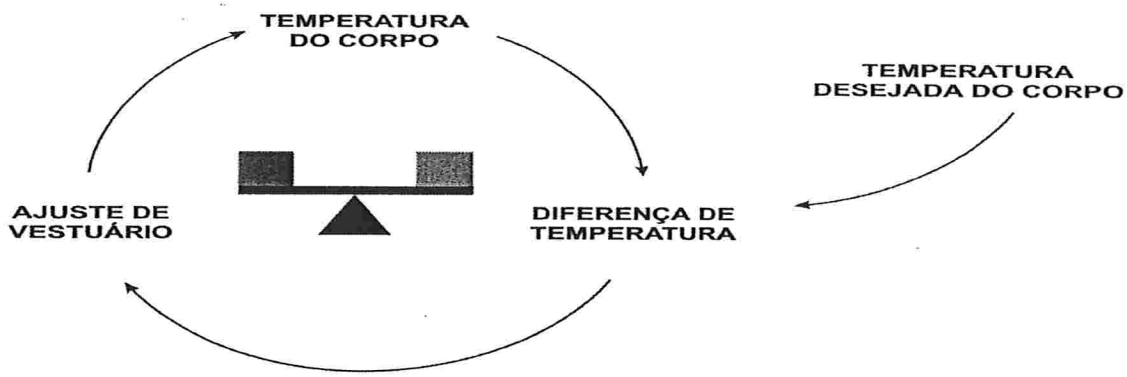


Figura 2.3: Processo de equilíbrio.

Fonte: adaptado de [Sen90, p. 84].

Além disso, diversos processos de retroalimentação apresentam, embutidos em seu funcionamento, defasagens. Tais defasagens são interrupções no fluxo de influências, fazendo que as conseqüências das ações ocorram de modo gradual. A figura 2.4 mostra um processo de equilíbrio com defasagem: visando reduzir a diferença entre as temperaturas observada e desejada da água, o indivíduo ajusta o volume de água utilizando a torneira. Porém, uma mudança de temperatura ocorre somente após um certo intervalo de tempo.

Os arquétipos são estruturas genéricas de padrões recorrentes, que visam facilitar o estudo das dinâmicas, ao permitir que sejam reconhecidas as diferentes estruturas em ação e os possíveis pontos de alavancagem. Os arquétipos são construídos a partir dos processos de reforço e equilíbrio e de retardos. Na figura 2.5 é apresentado o arquétipo "limites de crescimento": um processo de reforço é iniciado visando produzir um resultado desejado. Cria-se uma espiral de sucesso, entretanto é criado, como efeito secundário, um processo de equilíbrio que, finalmente, irá reduzir o ritmo de sucesso.

Finalmente, os laços causais são diagramas flexíveis, que exibem o caráter do relacionamento entre um par de conceitos, indicando, por exemplo, se a redução em uma variável causa um incremento ou redução em outra.

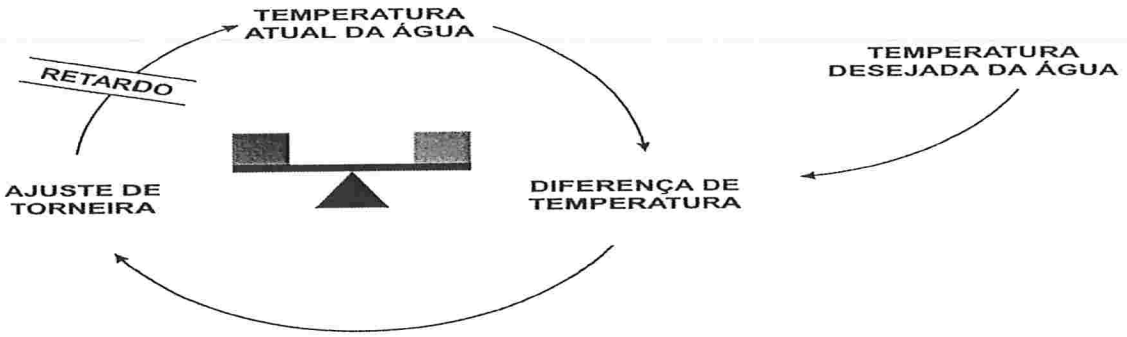


Figura 2.4: Processo de equilíbrio por retardo.

Fonte: adaptado de [Sen90, p. 90].

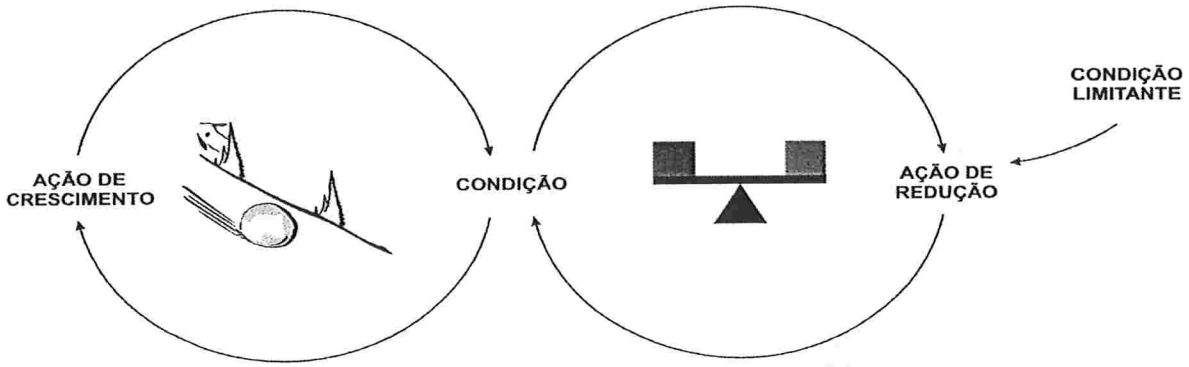


Figura 2.5: Limites de crescimento.

Fonte: adaptado de [Sen90, p. 90].

2.2.3 Domínio Pessoal

Domínio Pessoal é a disciplina que consiste em aprofundar e esclarecer continuamente a visão pessoal, de focar energias, e de desenvolver uma visão objetiva da realidade. Esta é uma disciplina essencial para o desenvolvimento da aprendizagem organizacional. Na verdade, Senge afirma que é seu alicerce espiritual. Aqui, o grande interesse reside na conexão entre as aprendizagens individual e organizacional.

Senge destaca, entretanto, que poucas organizações encorajam tal desenvolvimento pessoal,

o que resulta em um enorme desperdício de recursos. Por outro lado, o autor observa também que poucos indivíduos esforçam-se para desenvolver tal domínio; na maioria dos casos, estes concentram-se nos aspectos que os incomodam em suas vidas ao invés de dirigirem sua atenção e esforços para suas mais importantes aspirações.

Na verdade, aprendizado algum pode existir de modo sustentável sem que se baseie no interesse e curiosidade de cada indivíduo, afirma o autor. Sem isso, processos de capacitação de pessoal são aceitos com submissão pelos indivíduos, e não como um compromisso para o desenvolvimento. Além disso, os efeitos desses processos não são duradouros.

A prática principal desta disciplina envolve o desenvolvimento, por parte do indivíduo, da capacidade de manter tanto uma imagem clara da realidade quanto uma visão pessoal, gerando uma força interna denominada de "tensão criativa". Toda tensão requer resolução, o que, nesse caso, significa fazer com que a imagem da realidade se aproxime da visão pessoal (figura 2.6).

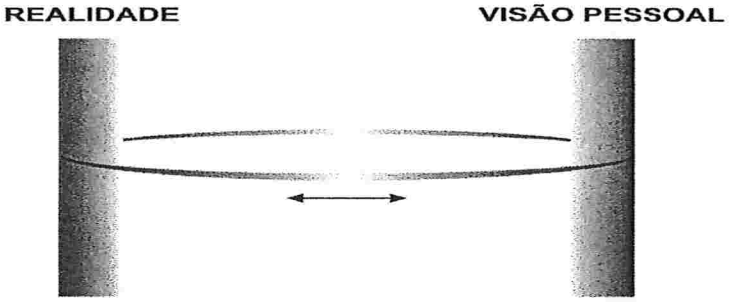


Figura 2.6: Tensão criativa.
Fonte: adaptado de [Sen90, p. 151].

2.2.4 Modelos Mentais

Os modelos mentais são hipóteses e generalizações que influenciam tanto a compreensão do mundo quanto o modo como o indivíduo interage com ele. Senge destaca que, freqüentemente, o indivíduo não está consciente dos seus modelos mentais e de suas influências sobre seu comportamento.

Diversas mudanças em organizações não são implementadas com sucesso em decorrência de conflitos que estas geram com poderosos modelos mentais tácitos pré-existentes.

Esta disciplina visa exteriorizar imagens, visões internas do mundo, para que estas possam ser criticadas e revisadas, se necessário. Tal processo envolve conversações (diálogos) que equilibram questionamento e defesa de pontos de vista, de modo que as pessoas possam expor seus pensamentos, tornando-se abertas a influências dos demais membros da organização.

Há dois tipos de habilidades centrais para que essa disciplina seja desenvolvida: reflexão e questionamento. O primeiro refere-se a uma redução no ritmo dos processos de pensamento do indivíduo para que este possa reconhecer como são formados seus modelos mentais. O segundo consiste em manter interações com outros indivíduos nas quais são compartilhadas abertamente suas visões e onde são desenvolvidos conhecimentos sobre as suposições de cada indivíduo a respeito de diversos assuntos.

2.2.5 Visão Compartilhada

Segundo Senge, a presença de metas, valores e missões que estejam profundamente compartilhados em toda a organização é essencial para a construção de uma organização de grande sucesso. Isto é fundamental para que a organização possa unir as pessoas em torno de uma identidade comum e de um senso de destino. Assim, esta visão é uma imagem compartilhada do futuro que a empresa deseja criar.

Frente a uma verdadeira visão, as pessoas desejam se superar e aprender. Não há a necessidade de uma norma definindo que os empregados devam agir com esse objetivo. Entretanto, nem sempre a visão de um líder se transforma em uma visão compartilhada. Esta disciplina pretende traduzir visão individual em visão compartilhada tendo como base a definição de princípios e diretrizes.

A prática desta disciplina envolve a exteriorização, por parte dos indivíduos, de imagens do futuro. Estas imagens devem promover o compromisso e alinhamento das pessoas com esta visão (decorrente das imagens). Desta forma, a disciplina da visão compartilhada se contrapõe a

processos que levem os indivíduos à simples submissão a uma visão imposta pela organização.

2.2.6 Aprendizagem em Equipe

Esta disciplina lida com a situação paradoxal de que, freqüentemente, o resultado total do trabalho em equipe é inferior à soma das capacidades de cada um de seus membros.

Quando uma equipe consegue desenvolver a capacidade de aprender, produz capacidades incomuns para a ação coordenada. Ao mesmo tempo, esta equipe gera resultados extraordinários e permite que seus membros possam se desenvolver mais rapidamente do que seria possível em outras circunstâncias.

Um dos princípios desta disciplina é o desenvolvimento da técnica de "diálogo". Neste contexto, "diálogo" é a capacidade dos membros da equipe de suprimirem suas hipóteses individuais e entrarem em um estado autêntico de "pensamento conjunto". Senge destaca que, em grego, *dialogos* tinha o sentido de fluxo livre de significados, pensamentos e intenções, através de um grupo, permitindo a descoberta por parte deste de percepções e insights que não poderiam ser obtidos individualmente. O autor alerta também para a diferença entre diálogo e discussão, consistindo esta última na troca de idéias e opiniões em forma de competição para determinar uma idéia "vencedora".

Outro aspecto interessante da disciplina de diálogo, é que esta também consiste em aprender a reconhecer padrões de interações em equipes que impedem ou dificultam o aprendizado conjunto. Um exemplo, bastante freqüente nas atividades em equipe, é o padrão defensivo de comportamento.

Esta disciplina é fundamental para a construção da organização aprendiz pois as equipes são as unidades básicas de aprendizado em organizações modernas. De fato, Senge afirma que a aprendizagem em equipe é um pré-requisito para a aprendizagem organizacional.

2.2.7 Modelagem Computacional

Modelos computacionais permitem uma visão mais concreta das conseqüências resultantes da adoção de determinadas hipóteses, ao contrário do mundo dos arquétipos, em que os elementos do sistema são vagamente definidos e onde os padrões de comportamento resultantes são meramente especulativos. Desta forma, pode-se produzir "laboratórios de aprendizagem" que funcionam como ambientes de transformação para equipes ou organizações [SKR+94].

Segundo [SKR+94], tais modelos têm sido usados para:

- Demonstrar como estruturas de sistemas produzem diretamente padrões de comportamento;
- Testar se uma determinada estrutura consegue reproduzir o desempenho observado no mundo real;
- Explorar como o comportamento irá se modificar quando diferentes aspectos da estrutura são alterados;
- Expor pontos de alavancagem, que poderiam não ser observados sem o uso deste ferramental; e
- Envolver equipes em uma aprendizagem sistêmica mais profunda ao permitir que as conseqüências de seus modos de pensar possam ser estudadas.

O grande problema na utilização de modelos computacionais reside na dificuldade em se aprender a representar o mundo real de modo confiável.

A simulação e modelagem computacional é essencial para o pensamento sistêmico, de acordo com [SKR+94]: modelos computacionais podem calcular com precisão as conseqüências das hipóteses representadas nos modelos, independentemente da complexidade dos sistemas. Adicionalmente, grandes quantidades de hipóteses podem ser simuladas e validadas utilizando-se tais modelos, o que, claramente, não pode ser realizado no sistema do mundo real. Assim, a modelagem computacional não é somente uma ferramenta de trabalho do pensamento sistêmico mas

uma poderosa ferramenta de aprendizagem deste pensamento, ao mostrar aos indivíduos de modo claro e prático as implicações de suas hipóteses.

2.2.8 A Importância do Todo

Peter Senge afirma que, no contexto de sua teoria, uma disciplina corresponde a um corpo de teoria e técnica que deve ser estudado e dominado visando a aplicação prática. Além disso, é muito importante ressaltar que praticar uma disciplina envolve o aprendizado contínuo.

Na visão desse autor, as cinco disciplinas devem ser desenvolvidas como um grupo harmonioso, utilizando-se a disciplina de pensamento sistêmico como o elemento que as funde em um corpo coerente de teoria e prática. Por essa razão, o pensamento sistêmico é denominado pelo autor de quinta disciplina. Na ausência de um pensamento sistêmico não há motivação para se observar como as demais disciplinas se inter-relacionam.

Entretanto, o pensamento sistêmico também depende das demais disciplinas para o desenvolvimento de todo seu potencial. A disciplina de modelos mentais resulta em uma visão aberta, necessária para que o indivíduo perceba deficiências, vícios e preconceitos em seu modo de enxergar o mundo. A visão compartilhada estimula o desenvolvimento de um compromisso de longo prazo. O domínio pessoal incentiva a motivação pessoal para o aprendizado contínuo de como as ações de cada indivíduo influenciam o mundo. Finalmente, o aprendizado em equipe permite que seja desenvolvida a habilidade de se enxergar o panorama geral existente, excedendo os limites das perspectivas individuais.

O objetivo de todo este trabalho é desenvolver uma visão global, que possa perceber relacionamentos e padrões de mudanças. Usando-se o pensamento sistêmico pode-se enxergar as estruturas subjacentes a situações complexas. Senge afirma que há dois tipos de complexidades a serem tratadas. A primeira é a complexidade de detalhes, que pode ser analisada com o uso de ferramentas sofisticadas de previsão e análise de negócios. A segunda é a complexidade dinâmica, onde causa e efeito possuem uma associação sutil. Para lidar com essa última, o pensamento sistêmico é a

ferramenta ideal.

Em resumo, o pensamento sistêmico torna compreensível o aspecto mais sutil da aprendizagem organizacional: uma nova percepção dos indivíduos em relação a si próprios e ao mundo. Os indivíduos devem perceber que estão conectados com o mundo, continuamente criando sua realidade e alterando-a.

2.3 Considerações Finais

Neste capítulo foi apresentada, resumidamente, a teoria da Quinta Disciplina de Senge, incluindo seus conceitos e hipóteses relativas aos indivíduos e à organização, para que seja possível criar uma organização aprendiz.

A teoria de Senge apresenta algumas características interessantes. Inicialmente, deve-se destacar a grande ênfase que o autor dá ao comportamento, iniciativa e capacidades de aprendizado individuais. Assim, o comportamento organizacional desejado, de "Organização que Aprende", deve emergir dessas características individuais e das interações entre os indivíduos. Além disso, essa teoria também se concentra em capacidades que podem ser desenvolvidas apenas coletivamente, como, por exemplo, as disciplinas de visão compartilhada e de aprendizagem em equipe.

Senge destaca ainda em seu livro uma série de casos de implementação dessa teoria, incluindo empresas importantes como: Shell, Harley-Davidson, Kyocera, e Federal Express.

Assim, a Quinta Disciplina é uma teoria moderna, relevante para a área de TO, bem sucedida, estruturada em cinco disciplinas, e que focaliza tanto aspectos individuais quanto coletivos. Portanto, a Quinta Disciplina se apresenta como uma teoria interessante e uma escolha adequada para o desenvolvimento da pesquisa reportada nesta tese.

No próximo capítulo, é apresentada uma visão geral da área de Sistemas Multi-Agentes, do arcabouço SMART e da linguagem Z com o objetivo de tornar claro o contexto no qual é desenvolvido o modelo formal para a teoria de Senge.

CAPÍTULO 3

Sistemas Multi-Agentes e Z

Neste capítulo é apresentada, de modo sucinto, uma visão da área de Sistemas Multi-Agentes (SMA). Além disso, na seção final é apresentado um breve tutorial sobre a linguagem formal Z.

3.1 Sistemas Multi-Agentes

Desde seu início, a área de Inteligência Artificial (IA) vem lidando com o problema de desenvolver teorias, técnicas e sistemas para estudar e compreender o comportamento e propriedades de raciocínio de uma entidade cognitiva individual. Com o desenvolvimento dessa área, o desafio de enfrentar problemas mais complexos, de larga escala e realísticos, provou estar além das capacidades de uma entidade individual. Suas capacidades são limitadas por seu conhecimento, recursos computacionais e sua perspectiva particular. Tais limitações correspondem ao que Herbert Simon, em *Models of Man*, de 1957, denomina de racionalidade limitada e que se torna exatamente uma das motivações subjacentes para a criação de organizações de resolução de problemas.

De acordo com [Syc98] a pesquisa em SMA concentra-se no estudo da construção e do comportamento de uma coleção de agentes autônomos, possivelmente preexistentes, que interagem entre si e com o ambiente. Portanto, além de aspectos relativos à inteligência individual, são abordados nesta área aspectos relativos a componentes sociais na resolução de problemas. Além disso, enfoques mais realísticos freqüentemente envolvem sistemas que se alteram dinamicamente, cujos

componentes são desconhecidos previamente e podem mudar no decorrer do tempo. Adicionalmente esses componentes podem ter implementações diferentes, efetuadas por diferentes equipes, em diferentes épocas, com o uso das mais variadas técnicas e recursos de computação. Sistemas com essas características são denominados de sistemas abertos.

Um SMA pode ser definido como uma rede de componentes (agentes) que buscam soluções para problemas. Tais componentes são autônomos, possivelmente heterogêneos, fracamente acoplados, e interagem para resolver problemas cujas complexidades excedem a capacidade individual de cada componente [Syc98].

Entre as características de um SMA, destacam-se: cada agente tem um ponto de vista limitado do problema, em função de informações ou recursos insuficientes; não há um controle global do sistema; os dados são descentralizados; e a computação é distribuída [Syc98].

Como conseqüência dessas características, esse paradigma mostra-se útil para enfrentar uma série de problemas, incluindo:

- Interconexão e interoperação de sistemas legados, via encapsulamento de software.
- Problemas que podem ser vistos convenientemente como uma sociedade de agentes autônomos interativos.
- Uso eficiente de fontes de informações distribuídas.
- Oferecer soluções onde o conhecimento especializado encontra-se distribuído. Por exemplo, engenharia concorrente [Dec98], saúde e manufatura [BW00], [PFL+99].
- Necessidade de aumento da eficiência computacional, confiabilidade, extensibilidade, disponibilidade, tolerância a falhas, flexibilidade, reutilização, e da facilidade para realizar manutenções.

Adicionalmente, as pesquisas em SMA também têm se beneficiado de conceitos e teorias desenvolvidos na área de teoria organizacional, como podem confirmar vários trabalhos publicados

recentemente [HSB02, CP98, CP94, DLPW95]. De acordo com [Syc98], uma organização fornece um arcabouço para interações entre agentes através da definição de papéis, expectativas de comportamentos e relações de autoridade.

Entretanto, apesar das diversas características positivas deste paradigma, existem importantes dificuldades e desafios a serem vencidos. Destaca-se entre eles o fato de que muitos conceitos envolvendo agência e desenvolvimento baseado em agentes ainda sofrem de uma falta de embasamento conceitual sólido. Um exemplo disso, é a própria conceituação do termo "agente". Há uma variedade de tipos de agentes: autônomos, inteligentes, virtuais, de interface, de informações, etc.. Além disso, não há consenso quanto às características básicas que compõem um agente [dL01].

Em [WJ95], um dos trabalhos mais importantes nessa área, são definidas duas noções para a definição de agentes: a noção fraca e a noção forte. A primeira inclui características como autonomia, sociabilidade, reatividade, pró-atividade. A autonomia envolve a capacidade do agente operar sem intervenção. A sociabilidade corresponde à capacidade do agente interagir com outros agentes. A reatividade permite ao agente perceber e reagir a mudanças em um ambiente dinâmico. Por último, a pró-atividade indica que o agente tem seu comportamento dirigido por metas. Quanto à noção forte, esta é associada à pesquisa em Inteligência Artificial e inclui noções mentais. As noções geralmente consideradas são: crença, desejo, intenção, conhecimento, entre outros.

Existem, entretanto, diversas outras caracterizações de agentes que incluem: veracidade, benevolência, racionalidade, adaptabilidade, entre outros. Além disso, para complicar ainda mais esse cenário, alguns autores apresentam definições conflitantes para as mesmas características. Por exemplo, em [EW95] a autonomia envolve um comportamento dirigido por metas e a manipulação de solicitações de alto nível por parte do agente.

Com o objetivo de resolver esse problema, [dL01] desenvolveu o arcabouço conceitual SMART.

O arcabouço SMART é formal, não ambíguo, estruturado e extensível, criado especificamente para servir como uma base para o estudo e construção de diversas arquiteturas orientadas a agentes.

3.2 A Linguagem Z

O arcabouço SMART é especificado na linguagem formal Z [Spi92]. Luck e D'Inverno [dL01] afirmam que essa linguagem foi escolhida por várias razões: ela permite o desenvolvimento formal de projetos de sistemas, possui uma conexão próxima com a implementação de sistemas computacionais; e utiliza uma notação simples, que é expressiva, estruturada e tem aceitação na comunidade de pesquisa em Inteligência Artificial. Apesar de existirem outros formalismos, tais como VDM [Jon90], CSP [Hoa78], CCS [Mil89], ou lógica modal [Che80]; nenhum desses ofereceu a mesma combinação de características requeridas pelo trabalho desenvolvido por esses autores.

A linguagem Z é formal, baseada em estados, e usa lógica de predicados, teoria de conjuntos e matemática discreta em sua notação. Tipos de dados baseados em matemática são usados para modelar os dados em um sistema. O comportamento do sistema especificado pode ser compreendido pelo uso de leis matemáticas que restringem estes tipos. Em Z, uma coleção de variáveis de estado pode ser usada para modelar o estado de um sistema. Adicionalmente, operações, representando mudanças de estados, podem ser especificadas. Especificações em Z freqüentemente começam com objetos atômicos, para os quais a estrutura interna não é relevante. Tais objetos são os membros dos conjuntos dados, ou tipos básicos, da especificação. Com base nesses tipos, tipos mais complexos podem ser construídos: conjuntos, tuplas, e esquemas. Além disso, definições por abreviação podem ser usadas para introduzir constantes globais. No apêndice A, são apresentadas informações básicas sobre Z. Informações mais detalhadas e abrangentes podem ser encontradas em [Spi92].

Para produzir a formalização apresentada nesta tese, foram utilizadas as seguintes referências: [Spi92, Spi89, Jac97]. A especificação produzida foi verificada sintaticamente[1] usando-se o produto ZTC versão 2.03 [ZTC03].

Especificações escritas em Z são compostas por dois tipos de parágrafos: parágrafos formais referentes à especificação propriamente dita, e parágrafos textuais que precedem os parágrafos

[1] Verificação de tipos.

formais e fornecem uma curta descrição do que está sendo especificado, facilitando assim a leitura da especificação.

A seguir são apresentados alguns conceitos introdutórios da linguagem Z que devem facilitar o entendimento do modelo formal apresentado nesta tese.

3.2.1 Conjuntos

O conceito de conjunto é central para a linguagem Z. Pode-se considerar que coleções de objetos similares formam conjuntos. Cada conjunto pode então ser tratado como um objeto da especificação. Os objetos que compõem um dado conjunto são denominados elementos.

Em Z só é possível construir conjuntos cujos elementos tenham o mesmo tipo, sendo que o tipo de um objeto corresponde ao conjunto maximal ao qual ele pertence.

As operações sobre conjuntos definidas em Z envolvem união, intersecção, diferença e potência. Além disso, são definidas as relações entre conjuntos e entre elementos e conjuntos. Dentre as relações entre conjuntos temos: igualdade, desigualdade, subconjunto e subconjunto próprio.

A representação em Z para essas relações e operações é apresentada abaixo.

$x \in A$	x é elemento de A
$\varnothing$	conjunto vazio
$A \subseteq B$	A é subconjunto de B
$A \subset B$	A é subconjunto próprio de B
$A \times B$	produto cartesiano
$A \cup B$	união dos conjuntos A e B
$A \cap B$	intersecção dos conjuntos A e B
$A \setminus B$	conjunto diferença entre A e B
$\bigcup A$	união generalizada dos elementos de A
$\#A$	tamanho de conjunto finito
$\mathbb{P} A$	conjunto potência de A

Considera-se, nesta tese, que estas operações e relações já sejam do conhecimento do leitor e, portanto, serão omitidos aqui os seus detalhamentos.

3.2.2 Definições

Na linguagem Z existem várias formas de se definir um objeto. Tais objetos podem ser simplesmente declarados, podem ser definidos por meio de abreviações, ou ainda as definições podem ser realizadas por meio de axiomas.

Declarações

A forma básica para se definir um objeto é simplesmente declará-lo. Tal declaração envolve a escrita do nome do tipo entre colchetes. Tipos assim declarados são denominados de tipos básicos. Desta forma, a declaração abaixo introduz um novo tipo básico, denominado T_1.

$[T_1]$

Uma única declaração pode introduzir a definição de mais de um tipo básico, conforme exemplo abaixo.

$[T_1, T_2, ..., T_n]$

Abreviações

Através de abreviações é possível introduzir novos nomes para objetos matemáticos definidos na especificação. No exemplo abaixo, o novo nome *id* é introduzido como uma constante global da especificação, possuindo o mesmo tipo e valor que a expressão *Expressão*.

$id == Expressão$

Definições Axiomáticas

Em Z, objetos podem ser definidos com restrições associadas a ele. As restrições devem ser respeitadas sempre que o símbolo for usado. O formato de uma definição axiomática é apresentado abaixo, onde o *predicado* especifica as restrições que se aplicam aos objetos introduzidos na *declaração*.

declaração
predicado

Definições genéricas

Em Z também é possível fazer-se a definição axiomática de uma forma genérica. Desta forma, pode-se definir uma família de constantes globais parametrizada por algum tipo X.

$[X]$
$a : X$
pred

Na definição apresentada acima, é definida uma constante a do tipo X que deve satisfazer o predicado *pred*. Neste tipo de definição, o conjunto X corresponde a um parâmetro formal cujo escopo é o corpo da definição.

Conjuntos e Predicados

Todos os objetos definidos em Z correspondem a conjuntos de algum tipo. Assim, em Z um predicado é definido em termos do conjunto de objetos que o satisfaz.

Assim, dado um predicado *pred* com uma variável livre a do tipo T; o conjunto C de valores de a que satisfazem *pred* é definido como segue.

$$C = \{a : T \mid pred\}$$

Pode-se, então, definir um símbolo como *Honesto*.

$$Honesto : \mathbb{P}\, T$$

...

Então, para se especificar que "a é honesto" é usada a representação:

$a \in Honesto$

3.2.3 Relações

Relações entre objetos são freqüentes no mundo real. Desta forma, em especificações formais há a necessidade de se descrever tais relações. Em Z, isto é possível usando-se objetos matemáticos denominados de *relações*.

Relações podem descrever associações entre um número finito qualquer de objetos. Entretanto, em Z usa-se relações entre pares de objetos, ou seja, relações binárias. Desta forma, uma relação é um subconjunto de um produto cartesiano.

Sejam A e B conjuntos. Então o conjunto de todas as relações entre A e B é representado por $A \leftrightarrow B$. Uma outra forma de se expressar uma relação é através de uma abreviação genérica.

$$A \leftrightarrow B == \mathbb{P}(A \times B)$$

Na relação acima, A é denominado conjunto fonte e B é denominado conjunto alvo. Qualquer elemento de tal relação é um conjunto de pares ordenados em que o primeiro elemento do par é tomado do conjunto fonte e o segundo do conjunto alvo.

Em uma relação R, subconjunto de $A \leftrightarrow B$, o domínio de R é o conjunto de elementos de A que se relacionam a algum objeto do conjunto alvo B. Por outro lado, o conjunto de elementos de B aos quais algum elemento de A está associado é denominado de imagem. Em Z, a notação para domínio e imagem de R é $\mathsf{dom}\, R$ e $\mathsf{ran}\, R$. Formalmente, domínio e imagem são definidos como segue.

$\mathsf{dom}\, R = \{a : A \mid \exists\, b : B \bullet (a, b) \in R\}$
$\mathsf{ran}\, R = \{b : B \mid \exists\, a : A \bullet (a, b) \in R\}$

Adicionalmente, pode-se considerar apenas parte dos conjuntos domínio e imagem através da imposição de restrições a esses conjuntos.

Assim, dada uma relação R, subconjunto de $A \leftrightarrow B$, e X, um subconjunto qualquer de A, então $X \lhd R$ denota a restrição de domínio de R para X. Ao se aplicar tal restrição, considera-se apenas o comportamento da relação R sobre o subconjunto X de A. Formalmente, temos:

$$X \lhd R = \{a : A;\ b : B \mid a \in X \land (a, b) \in R\}$$

De modo similar, pode-se considerar Y, um subconjunto qualquer de B, então $R \rhd B$ denota a restrição da imagem de R para Y. Ao se aplicar tal restrição, considera-se apenas o comportamento da relação R sobre o subconjunto Y de B. Formalmente, temos:

$$R \rhd B = \{a : A;\ b : B \mid b \in Y \land (a, b) \in R\}$$

Também pode ser interessante considerar a diferença entre os conjuntos fonte e alvo, e um outro conjunto dado. Desta forma, considerando-se a relação R acima e os conjunto X e Y, pode-se definir a diferença sobre o domínio e a diferença sobre a imagem desta relação.

$X \lhd\!\!\!\!- R = \{a : A;\ b : B \mid a \notin X \land (a, b) \in R\}$
$R \rhd\!\!\!\!- B = \{a : A;\ b : B \mid b \notin Y \land (a, b) \in R\}$

Relações são direcionais, envolvendo associações entre objetos de um conjunto fonte e objetos de um conjunto alvo. Contudo, deve-se observar que é possível reverter tal direção usando-se o operador $^\sim$, de tal forma que, considerando-se a relação R acima, o conjunto B passa a ser o conjunto fonte e o conjunto A passa a ser o conjunto alvo de $R^\sim$. Desta forma, temos:

$$\forall\, a : A;\ b : B \bullet b \mapsto a \in R^\sim \Rightarrow a \mapsto b \in R$$

Uma outra operação útil sobre relações é a composição. Considerando-se as relações R e S, é possível obter-se a relação composta de R e S se o conjunto alvo de R for idêntico ao conjunto fonte de S. Supondo-se então $R \subseteq A \leftrightarrow B$ e $S \subseteq B \leftrightarrow C$, a composta de R e S é dada por:

$$R \fatsemi S = \{a : A;\ c : C \mid \exists b : B \bullet (a, b) \in R \land (b, c) \in S\}$$

3.2.4 Funções

Funções são casos particulares de relações em que cada objeto de um conjunto está associado a no máximo um objeto do outro conjunto.

Uma função pode ser parcial caso possam existir elementos de seu conjunto fonte que não estejam associados a elementos de seu conjunto alvo. Por outro lado, uma função é dita total caso todo elemento de seu conjunto fonte esteja associado a um elemento de seu conjunto alvo.

Sejam A e B conjuntos. A notação em Z para definir uma função parcial tendo como fonte A e alvo B é $A \pfun B$. A notação para uma função total é $A \fun B$. Formalmente, temos:

$$A \pfun B = \{f : A \leftrightarrow B \mid \forall a : A;\ b_1, b_2 : B \bullet a \mapsto b_1 \in f \land a \mapsto b_2 \in f \Rightarrow b_1 = b_2\}$$
$$A \fun B = \{f : A \pfun B \mid \mathrm{dom}\, f = A\}$$

Adicionalmente, há outras importantes classes de funções: injetoras, sobrejetoras e bijetoras. A notação em Z para estas categorias segue abaixo.

$A \rightarrowtail B$	função injetora total
$A \pinj B$	função injetora parcial
$A \twoheadrightarrow B$	função sobrejetora total
$A \psurj B$	função sobrejetora parcial
$A \bij B$	função bijetora total

Seja f uma função, A o conjunto fonte e B o conjunto alvo de f.

A função f é injetora se:

$\forall x_1, x_2 : \mathsf{dom}\, f \bullet fx_1 = fx_2 \Rightarrow x_1 = x_2$

A função f é sobrejetora se:

$\mathsf{ran}\, f = B$

A função f é bijetora se for injetora e sobrejetora.

Uma operação útil sobre funções é a sobreposição. Considerando duas funções $f \subseteq A \times B$ e $g \subseteq A \times B$, a sobreposição de f e g é definida como a função $f \oplus g \subseteq A \times B$. Esta função assume os valores definidos no domínio de f com exceção da intersecção dos domínios de f e g, onde prevalecem os valores definidos por g.

$f \oplus g = ((\mathsf{dom}\, g) \rhd f) \cup g$

3.2.5 Seqüências

Uma seqüência é uma coleção ordenada de zero ou mais objetos. Uma seqüência vazia é denotada como $\langle\rangle$. Uma seqüência contendo os objetos x, y e z, nesta ordem, é denotada por $\langle x, y, z\rangle$.

Uma operação útil sobre seqüências é a concatenação, na qual duas seqüências são combinadas de modo que os elementos de uma delas são colocados após os elementos da outra respeitando-se a ordem original de cada uma, produzindo uma nova seqüência que contém os elementos das seqüências originais. Supondo-se que s e t sejam seqüências a concatenação de s e t é denotada por $s \frown t$.

Adicionalmente, pode-se fazer referência ao primeiro elemento de uma seqüência não vazia s, ou à parte de s que inclui do segundo até o último elemento de s. Para tanto, usa-se as operações *head* e *tail*. Assim, temos:

$s = \langle head\ s\rangle \frown tail\ s$

Pode ocorrer a situação em que haja interesse em apenas um conjunto de elementos de uma

seqüência. Neste caso, pode-se usar o operador de filtro $\upharpoonright$. Supondo-se que s seja uma seqüência e que A seja um conjunto, então $s \upharpoonright A$ corresponde à maior subseqüência de s contendo somente os objetos que sejam elementos de A.

Um outro modo de se extrair informações de uma seqüência é por meio do uso dos índices dos elementos desejados. Para tanto, usa-se o operador de extração $\upharpoonleft$. Supondo-se que s seja uma seqüência e que I seja um conjunto de números inteiros positivos maiores que zero; então, $I \upharpoonleft s$ corresponde à seqüência de elementos de s localizados nas posições de s cujos índices estejam presentes em I.

Uma outra propriedade importante de uma seqüência é o seu tamanho. A expressão $\#s$ denota o tamanho da seqüência s.

3.2.6 Tipos Construídos

Tipos construídos podem ser úteis à definição de uma série de estruturas como listas, coleções de tipos enumerados e estruturas definidas recursivamente.

Uma declaração de tipos enumerados tem a seguinte forma:

$$T ::= c_1 \mid c_2 \mid ... \mid c_n$$

Onde T é um novo tipo, e as constantes $c_1, c_2, ..., c_n$ são os únicos elementos do tipo T.

Podem ser incluídas cópias de outros conjuntos complementando um tipo construído usando funções construtoras. Assim, a notação abaixo introduz uma coleção de constantes advindas do conjunto *fonte*. A função *construtora* é injetora e tem como alvo o conjunto *TipoConstruido*.

$$TipoConstruido ::= construtora\langle\langle fonte\rangle\rangle$$

Além disso, constantes e funções construtoras podem ser usadas juntas na mesma definição.

$$\begin{aligned} T ::= & \; c_1 \mid c_2 \mid ... \mid c_n \\ & \mid construtora\langle\langle fonte\rangle\rangle \end{aligned}$$

Adicionalmente, o tipo fonte de uma função construtora pode se referir ao próprio tipo que esteja sendo definido, permitindo uma definição recursiva. Por exemplo, uma árvore binária pode ser definida como segue.

$$ArvoreBinaria ::= folha\langle\langle \mathbb{N} \rangle\rangle \mid ramo\langle\langle ArvoreBinaria \times ArvoreBinaria \rangle\rangle$$

3.2.7 Esquemas

Esquemas são utilizados em Z para estruturar a especificação formal em partes que possam ser mais facilmente compreendidas e que posteriormente possam ser combinadas e reutilizadas, facilitando a leitura e o processo de especificação.

Um esquema pode ser visto como um trecho de texto formal, que descreve algumas variáveis cujos valores são restritos de algum modo, e ao qual está associado um nome.

Há duas partes em um esquema: uma parte declarativa onde variáveis são introduzidas, e uma parte predicativa que apresenta as relações e restrições envolvendo algumas (ou todas) as variáveis do esquema. A parte predicativa pode ser vazia.

Esquemas podem ser apresentados em duas formas, horizontalmente ou verticalmente, conforme visto abaixo.

$$S_1 \mathrel{\widehat=} [D_1;\ ...;\ D_n \mid P_1;\ ...;\ P_m]$$

S_2

$D_1;\ ...;\ D_n$

$P_1;\ ...;\ P_m$

S_3

$D_1;\ ...;\ D_n$

Nos esquemas apresentados acima, S_1, S_2 e S_3 são os nomes dos esquemas, D_1; ...; D_n correspondem às declarações das variáveis e P_1; ...; P_m correspondem à parte predicativa. O esquema S_3 não possui parte predicativa. Os esquemas S_2 e S_3 são apresentados no formato vertical e o esquema S_1 é apresentado no formato horizontal.

Em esquemas apresentados verticalmente cada declaração de variável pode ser apresentada em uma nova linha, facilitando a leitura. O mesmo vale para a parte predicativa.

Esquemas podem também ser vistos como tipos da linguagem Z. Além dos tipos: básico, construído, conjunto potência e produto cartesiano, um esquema permite a introdução de tipos compostos, envolvendo uma variedade de componentes.

Por exemplo, o esquema *Sinteiro* corresponde a um tipo de dado composto por dois componentes: um conjunto de inteiros *sint* e um inteiro *int*. A introdução de uma variável desse tipo pode ser efetuada através de uma declaração var_1 : *Sinteiro*.

Sinteiro

$int : \mathbb{Z}$

$sint : \mathbb{P}\,\mathbb{Z}$

Para se descrever um objeto de um tipo esquema são listados os nomes dos componentes seguidos dos valores a eles atribuídos. Por exemplo, uma possível valoração para um objeto desse tipo pode ser descrita conforme o exemplo a seguir.

$Sinteiro\langle\!|\ int == 1, sint == \{1, 3, 5, 7\}\ |\!\rangle$

O tipo esquema *Sinteiro* é o conjunto de todas as valorações nas quais *sint* e *int* são associados, respectivamente, a um conjunto de inteiros e a um inteiro.

A ordem da declaração dos componentes não é importante em um tipo esquema. Para se fazer referência a um dado componente do esquema usa-se o operador de seleção "." . Por exemplo, considerando-se a declaração var_1 : *Sinteiro*, pode-se fazer referência ao componente inteiro de *Sinteiro* por meio da notação $var_1.int$.

Uma das operações mais freqüentes realizadas sobre esquemas é a inclusão. A parte declarativa

de um esquema pode conter declarações de variáveis de diversos tipos. Além disso, essa parte pode conter referências a nomes de esquemas previamente definidos. Tal referência é denominada de inclusão de esquema, pois implica na inclusão de todas as declarações e predicados do esquema "referenciado", no esquema que efetua a referência.

Por exemplo, no esquema S_a abaixo, ocorre a inclusão do esquema *Sinteiro* declarado anteriormente.

S_a

$$\begin{array}{|l} \textit{Sinteiro} \\ \textit{int2} : \mathbb{Z} \\ \hline \textit{int2} < \textit{int} \\ \textit{int2} \in \textit{sint} \\ \hline \end{array}$$

Esquemas são usados em especificações Z para descrever estados de um sistema e também operações sobre esses estados.

Existem algumas convenções em Z para descrever que o estado determinado por um dado esquema sofre alguma ação e que este estado pode ser modificado ou permanecer inalterado.

Inicialmente, há a necessidade de se informar que existem convenções em Z para a decoração de declarações de variáveis por meio do uso de caracteres especiais. Desta forma, pode-se indicar que uma variável é de entrada, de saída ou de estado através da justaposição dos caracteres "?", "!" e " ′ " ao nome da variável.

Pode-se agora caracterizar um esquema de descrição de estados como sendo aquele em que não ocorrem variáveis decoradas por quaisquer dos símbolos citados acima. Tal esquema apenas define as variáveis significativas, relações entre essas variáveis e restrições que se apliquem aos valores que essas variáveis podem assumir. Além disso, a justaposição do caracter " ′ " ao nome de um esquema de estado S designa um novo esquema contendo os mesmos predicados e variáveis declaradas em S, com a justaposição do caracter " ′ " a toda variável contida na parte declarativa e a cada ocorrência livre de variável na parte predicativa.

Por outro lado, em esquema de operações são usadas variáveis de entrada e de saída e, também, pode-se atribuir novos valores às variáveis declaradas no esquema.

Para simplificar o processo de definição de um esquema de operação sobre um esquema de estado usa-se a variante Δ. Supondo-se que S seja um esquema de estado, ΔS define um novo esquema definido como segue.

ΔS

S
S'

Cada nome de variável decorado com " $'$ " indica uma variável de estado que conterá o novo valor a ser assumido pela variável original (cujo nome não está decorado) após a realização da operação.

Por exemplo, uma operação sobre o esquema *Sinteiro* pode ser definida como segue.

MudaSinteiro

$\Delta Sinteiro$
$int? : \mathbb{Z}$

$int' = int?$
$sint' = sint$

Há também em Z uma forma simplificada de se definir que em uma dada operação as variáveis de estado permanecem inalteradas. Para tanto usa-se a variante Ξ. Supondo-se que S seja um esquema de estado, ΔS define um novo esquema definido como segue.

ΞS

ΔS

$v_1' = v_1$
$v_2' = v_2$
.
.
.
$v_n' = v_n$

No exemplo acima, $v_1', ..., v_n'$ correspondem às variáveis de estado presentes em S'.

Por exemplo, pode-se definir uma operação sobre o esquema S_a que altera o estado apenas da variável declarada nesse esquema, mantendo intacto o estado descrito por *Sinteiro*.

$MudaS_a$

ΔS_a
$\Xi Sinteiro$
$intNovo? : \mathbb{Z}$

$int2' = int2 + intNovo?$

3.2.8 Notações Lambda e θ

Em Z há uma forma específica para se definir funções, denominada de expressão lambda. Essa expressão apresenta uma estrutura contendo *declaração*, *predicado* e *expressão*.

$(\lambda\, declaração \mid predicado \bullet expressão)$

Na verdade, considerando funções como conjuntos de pares, o primeiro elemento de cada par é determinado pela *declaração* e pelo *predicado* e o segundo elemento é descrito pela *expressão*.

No exemplo abaixo é definida a função *dobro* que associa a cada inteiro o dobro de seu valor.

$dobro == (\lambda\, inteiro : \mathbb{Z} \bullet inteiro * 2)$

A expressão lambda pode ser usada em qualquer parte de uma especificação formal. No caso acima, essa expressão foi usada para introduzir a definição de uma função que pode ser usada diversas vezes no contexto da especificação, como apresentado a seguir.

$dobro\ 8$

Como discutido na seção anterior, uma referência a um esquema é, na verdade, uma referência a um conjunto de valorações. Podem haver, entretanto, situações em que surja a necessidade de se ter acesso às valorações de um dado esquema. Em Z o operador θ é usado com esta finalidade. Desta forma, supondo-se que NE seja um esquema, a expressão θNE se refere à valoração de NE que estiver em escopo no ponto da especificação em que for declarada essa expressão.

Além disso, o operador θ permite a definição de uma relação associada a um esquema de operação. Supondo que *OperaçãoSobreEsquema1* seja uma operação definida sobre o esquema *Esquema1*, a declaração abaixo define uma relação entre estados anteriores e posteriores de *Esquema1*.

OperaçãoSobreEsquema1

$\Delta Esquema1$

...

$relaçãoEntreEstados == \{OperaçãoSobreEsquema1 \bullet (\theta Esquema1, \theta Esquema1')\}$

Um outro uso interessante das notações θ e λ envolve operações de inclusão de esquemas. Considerando os esquemas abaixo:

Sinteiro

$int : \mathbb{Z}$

$sint : \mathbb{P}\,\mathbb{Z}$

S_a

$Sinteiro$
$int2 : \mathbb{Z}$

$int2 < int$
$int2 \in sint$

Em algumas situações, tendo em escopo o esquema S_a, pode haver a necessidade de se fazer referência ao esquema *Sinteiro*. Isto pode ser feito com o uso de uma função do tipo:

$(\lambda\, S_a \bullet \theta Sinteiro)$

3.3 Considerações Finais

Neste capítulo foi apresentada, de modo sucinto, uma visão geral da área de Sistemas Multi-Agentes (SMA). Adicionalmente, o arcabouço formal SMART foi objeto de uma apresentação introdutória. Concluindo, na seção final foi apresentado um breve tutorial sobre a linguagem formal Z, que visa servir como material de apoio para a leitura das especificações presentes nesta tese. O próximo capítulo contém uma visão mais detalhada do arcabouço SMART, com a especificação em Z dos tipos formais que servem de apoio para o modelo formal apresentado nesta tese.

CAPÍTULO 4

Visão Geral do Arcabouço SMART

Este capítulo contém uma descrição mais detalhada do arcabouço formal SMART, que serviu de base para a construção do modelo formal para a teoria de Senge. Nas seções a seguir, são apresentadas as especificações relativas a ambiente, entidades, objetos, agentes e agentes autônomos, bem como comentários sobre as interpretações particulares dos autores do SMART sobre esses conceitos. A apresentação a seguir se baseia em [dL01], primeira edição. Cumpre observar que uma segunda edição dessa mesma obra já está disponível, em que diversos ajustes e aperfeiçoamentos estão presentes. Dentre eles, consta inclusive algumas correções sugeridas pelo autor desta tese, as quais foram devidamente referenciadas pelos autores do livro [dL04, p. VII].

4.1 Introdução

No arcabouço SMART (*Structured and Modular Agents and Relationship Types*), o mundo é composto por entidades, que são especificadas como uma coleção de atributos. Além disso, entidades são classificadas de acordo com características adicionais que possuam: objetos são entidades com uma coleção de capacidades, agentes possuem um conjunto de metas e agentes autônomos têm motivações.

Para caracterizar tais elementos formalmente, alguns tipos básicos precisam ser definidos.

Os tipos *Attribute*, *Action*, *Goal* e *Motivation* são definidos em [dL01] do seguinte modo[1]:

- "Um atributo (*Attribute*) é uma característica perceptível".
- "Uma ação (*Action*) é um evento discreto que pode mudar o estado do ambiente quando executado".
- "Uma meta (*Goal*) é um estado do ambiente a ser realizado".
- "Uma motivação (*Motivation*) é qualquer desejo ou preferência que pode levar à geração e adoção de metas e que afeta o resultado do raciocínio ou tarefa comportamental, cuja intenção seja satisfazer tais metas".

Os tipos *Action* e *Motivation* são definidos como tipos básicos, enquanto os tipos *Goal* e *Actions* são definidos pelo uso de abreviação como um conjunto não vazio de atributos e como um conjunto de ações, respectivamente.

$[Motivation, Action]$

$Goal == \mathbb{P}_1\ Attribute \qquad Actions == \mathbb{P}\ Action$

Um conjunto *Attribute* corresponde a um conjunto de fórmulas em lógica de predicados de primeira ordem.

Assim, o esquema *Atom*, inclui um símbolo de predicado e uma seqüência de tipo *Term* como seu argumento.

Atom

$head : PredSym$
$terms : \text{seq}\ Term$

Termos (*Term*) podem ser constantes, variáveis, um símbolo de *functor* com uma seqüência de termos como parâmetros, e são denotados como tipos básicos. Além disso, símbolos de predicados são também definidos como tipos básicos.

[1]Traduzido pelo autor.

$[Const, Var, FunSym, PredSym]$

$$
\begin{aligned}
Term ::= & \ const\langle\langle Const\rangle\rangle \\
| & \ var\langle\langle Var\rangle\rangle \\
| & \ functor\langle\langle FunSym \times \text{seq } Term\rangle\rangle
\end{aligned}
$$

No SMART, atributos são definidos como predicados que não contêm variáveis. Assim, [dL01] usa as funções auxiliares *termvars* e *atomvars*[2]. Desta forma, predicados sem variáveis, e o tipo *Attribute* podem ser definidos.

$$
\begin{array}{|l}
termvars : Term \rightarrow \mathbb{P}\ Var \\
atomvars : Atom \rightarrow \mathbb{P}\ Var \\
\hline
\forall\, at : Atom;\ c : Const;\ v : Var;\ f : FunSym;\ ts : \\
\quad \text{seq } Term \bullet (termvars\ (const\ c) = \varnothing \wedge \\
\quad termvars\ (var\ v) = \{v\} \wedge \\
\quad termvars\ (functor(f, ts)) = \bigcup(\mathsf{ran}(mapseq \\
\qquad termvars\ ts)) \wedge \\
\quad atomvars\ at = \\
\qquad \bigcup(\mathsf{ran}(mapseq\ termvars\ at.terms)))
\end{array}
$$

$$
\begin{aligned}
BaseAtoms &== \{a : Atom \mid atomvars\ a = \varnothing\} \\
Attribute &== BaseAtoms
\end{aligned}
$$

Para que seja possível distinguir representações do ambiente real de representações internas aos agentes, o tipo *View* é definido.

$$View == \mathbb{P}_1\ Attribute$$

Em seguida, são apresentadas especificações de ambiente, entidade, objeto e agente, conforme especificado pelo arcabouço SMART.

[2] A função genérica *mapseq* é definida no apêndice A

4.2 Ambiente e Entidades

O nível mais alto de abstração no SMART corresponde ao tipo *Entity*, que é especificado como um conjunto não vazio de atributos, e também conjuntos de ações, metas e motivações. No modelo formal apresentado nesta tese, porém, foi necessário estender este tipo para incluir variáveis necessárias para a especificação de outros tipos de agentes, que serão apresentados nas próximas seções.

Entity

$$
\begin{array}{l}
attributes, store : \mathbb{P}\ Attribute \\
capableof : \mathbb{P}\ Action \\
goals, allgoals : \mathbb{P}\ Goal \\
motivations : \mathbb{P}\ Motivation \\
ownedresources : \mathbb{P}\ Entity \\
instsreq : Plan \pfun (\mathrm{seq}(Action \times \mathbb{P}\ Entity)) \\
resourcesofplan : Plan \pfun \mathbb{P}\ Entity \\
plans, allplans : \mathbb{P}\ Plan \\
planforgoal : Goal \pfun \mathbb{P}\ Plan \\
orgs : \mathbb{P}\ Organization \\
roles : \mathbb{P}\ Role \\
ownguidingideas : GuidingIdea \\
personalMasteryCapabilities, mentalModelsCapabilities : \mathbb{P}\ Action \\
systemsThinkingCapabilities, learningTeamCapabilities : \mathbb{P}\ Action \\
buildingSharedVisionCapabilities : \mathbb{P}\ Action \\
\hline
attributes \neq \varnothing
\end{array}
$$

Entidades precisam ser situadas em um ambiente. O tipo *Env* define o ambiente como um conjunto de atributos, representando todos os atributos válidos em um dado instante nesse ambiente. O estado do ambiente inclui em si todas as entidades.

$$Env == \mathbb{P}_1\ Attribute$$

EnvironmentState

$$env : Env$$
$$entities : \mathbb{P}\, Entity$$
$$\bigcup \{e : Entity \mid e \in entities \bullet e.attributes\} \subseteq env$$

4.3 Objetos

Um objeto é uma entidade capaz de realizar um conjunto de ações, enquanto um objeto neutro é um objeto que não pode atuar como um agente.

Object

$$Entity$$
$$capableof \neq \varnothing$$

NeutralObject

$$Object$$
$$goals = \varnothing \land motivations = \varnothing$$

As ações do objeto serão executadas em um ambiente e dependerão do estado de tal ambiente. Aqui a função *willdo* corresponde à função de seleção de ações do objeto. O esquema *ObjectState* formaliza o estado de um objeto situado em um ambiente.

ObjectAction

$$Object$$
$$objectact : Env \rightarrow Actions$$
$$\forall\, env : Env \bullet (objectact\ env) \subset capableof$$

ObjectState

$$EnvironmentState$$
$$ObjectAction$$
$$willdo : Actions$$

$$willdo = objectact\ env \land willdo \subset capableof$$

Quanto às operações de um objeto, variáveis associadas a estado podem mudar, enquanto outras variáveis como atributos, capacidades e a função de seleção, permanecem inalteradas. Se elas mudarem, a instanciação de um novo objeto ocorre [dL01].

Δ *ObjectState*

$$ObjectState$$
$$ObjectState'$$
$$\Xi ObjectAction$$

Interações mudam o estado do ambiente pela adição e remoção de atributos.

$$effectinteraction : Env \rightarrow \mathbb{P}\ Action \nrightarrow Env$$

O esquema *ObjectInteracts* mostra objetos individuais interagindo com seu ambiente.

ObjectInteracts

$$\Delta ObjectState$$

$$env' = effectinteraction\ env\ willdo$$
$$willdo' = objectact\ env'$$

4.4 Agentes

Um agente é um objeto que tem metas, adotadas ou geradas por ele mesmo. Um agente servidor é um agente que não possui motivações e, portanto, não pode criar suas próprias metas.

Agent

$$Object$$

$$goals \neq \varnothing$$

ServerAgent

$$Agent$$

$$motivations = \varnothing$$

O esquema *AgentPerception* define as capacidades de percepção de um agente. A função *canperceive* define os atributos que podem ser percebidos, enquanto a função *agperceives* define os atributos realmente percebidos pelo agente.

AgentPerception

$$Agent$$
$$peractions : Actions$$
$$canperceive : Env \rightarrow Actions \rightarrow Env$$
$$agperceives : \mathbb{P}\, Goal \rightarrow View \rightarrow View$$

$$peractions \subset capableof$$
$$\forall\, env : Env;\ as : Actions \bullet$$
$$\quad (as \in (\mathsf{dom}(canperceive\ env)) \Rightarrow$$
$$\quad as = peractions)$$
$$\mathsf{dom}\ agperceives = \{goals\}$$

O esquema *AgentAction* define que metas, percepções e ambiente influenciam o comportamento do agente.

AgentAction

$$Agent$$
$$ObjectAction$$
$$agentact : \mathbb{P}\, Goal \rightarrow View \rightarrow Env \rightarrow Actions$$

$$\forall\, g : \mathbb{P}\, Goal;\ v : View;\ env : Env \bullet$$
$$\quad (agentact\ g\ v\ env) \subset capableof$$
$$\mathsf{dom}\ agentact = \{goals\}$$

O esquema *AgentState* formaliza um agente situado em um ambiente. As variáveis *posspercepts* e *actualpercepts* correspondem a atributos do ambiente que são computados usando as funções *canperceive* e *agperceives*.

AgentState

$$
\begin{array}{l}
AgentAction \\
AgentPerception \\
ObjectState \\
posspercepts, actualpercepts : View \\
\hline
actualpercepts \subset posspercepts \\
posspercepts = canperceive\ env\ peractions \\
actualpercepts = agperceives\ goals\ posspercepts \\
peractions = \varnothing \Rightarrow posspercepts = \varnothing \\
willdo = agentact\ goals\ actualpercepts\ env
\end{array}
$$

Considerando as operações do agente, temos os esquemas $\Delta AgentState$ e *AgentInteracts*. Segundo este esquema, atributos, capacidades, metas, capacidades perceptivas, e funções de percepção e seleção de ações permanecem inalteradas.

Δ *AgentState*

$$
\begin{array}{l}
AgentState \\
AgentState' \\
\Delta ObjectState \\
\Xi AgentAction \\
\Xi AgentPerception
\end{array}
$$

O esquema *AgentInteracts* define como agentes individuais interagem com o ambiente.

AgentInteracts

$$
\begin{array}{l}
\Delta AgentState \\
ObjectInteracts \\
\hline
posspercepts' = canperceive\ env'\ peractions \\
actualpercepts' = agperceives\ goals\ posspercepts' \\
willdo' = agentact\ goals\ actualpercepts'\ env'
\end{array}
$$

4.5 Agentes Autônomos

Um agente autônomo possui motivações. Segundo [dL01], motivações são o princípio básico para a autonomia de agentes, uma vez que, a partir delas, o agente pode gerar suas próprias metas.

AutonomousAgent

$$Agent$$

$$motivations \neq \varnothing$$

O esquema *AutonomousAgentPerception* define as capacidades perceptivas de um agente autônomo. No caso desse tipo de agente, tanto metas como motivações são importantes para determinar o que é percebido em um ambiente. Desta forma, é introduzida aqui uma nova versão da função *agperceives*, denominada *autonomouswillperceive*. A capacidade de percepção, definida pela função *canperceive* não é alterada pela presença de motivações ou metas, logo permanece a mesma conforme a definição apresentada anteriormente no esquema *AgentPerception*.

AutonomousAgentPerception

$$AutonomousAgent$$
$$AgentPerception$$
$$autonomouswillperceive : \mathbb{P}\, Motivation \rightarrow \mathbb{P}\, Goal \rightarrow Environment \rightarrow View$$

$$\mathrm{dom}\ autonomouswillperceive = \{motivations\}$$

No caso de agentes autônomos, a função de seleção de ações deve considerar a cada instante as motivações dos agentes.

AutonomousAgentAction

$$AutonomousAgent$$
$$AgentAction$$
$$autonomousactions : \mathbb{P}\, Motivation \rightarrow \mathbb{P}\, Goal \rightarrow View \rightarrow Environment \rightarrow Actions$$

$$\mathrm{dom}\ autonomousactions = \{motivations\}$$

O estado do agente autônomo é definido como um refinamento da definição de estado de um agente. As ações executadas por tal agente são função de suas motivações, metas, percepções e ambiente.

AutonomousAgentState

$$
\begin{array}{|l}
\hline
AgentState \\
AutonomousAgentPerception \\
AutonomousAgentAction \\
\hline
willdo = autonomousactions\ motivations\ goals \\
\quad actualpercepts\ env \\
\hline
\end{array}
$$

Considerando operações de agentes autônomos, são definidos os esquemas: $\Delta AutonomousAgentState$ e $AutonomousAgentInteracts$.

Δ *AutonomousAgentState*

$$
\begin{array}{|l}
\hline
AutonomousAgentState \\
AutonomousAgentState' \\
\hline
DeltaAgentState \\
autonomouswillperceive' = autonomouswillperceive \\
autonomousactions' = autonomousactions \\
\hline
\end{array}
$$

O esquema *AutonomousAgentInteracts* define que a operação de um agente autônomo executando um conjunto de ações é especificada como um refinamento do esquema *AgentInteracts*.

AutonomousAgentInteracts

$$
\begin{array}{|l}
\hline
DeltaAutonomousAgentState \\
AgentInteracts \\
\hline
posspercepts' = canperceive\ env'\ peractions \\
actualpercepts' = autonomouswillperceive\ motivations' \\
\quad goals'\ posspercepts' \\
willdo' = autonomousactions\ motivations'\ goals' \\
\quad actualpercepts'\ env' \\
\hline
\end{array}
$$

4.6 Agentes com Memória

Em [dL01], um refinamento para o tipo *Agent* é apresentado: agentes com memória. Tais agentes são capazes de obter e armazenar informações como um estado interno. Estas informações são armazenadas em suas áreas de memória interna. No modelo apresentado nesta tese, este tipo de agente representa a primeira camada que é capaz de armazenar representações internas do ambiente e de raciocinar empregando tais representações.

StoreAgent

$$Agent$$

$$store \neq \varnothing$$

No caso desse agente, as ações envolvendo percepções podem ser tanto internas quanto externas, quando se referirem respectivamente à memória do agente, ou ao ambiente. Tais ações correspondem, na especificação, a *internalperactions* and *externalperactions*, respectivamente. De modo correspondente, as funções *storecanperceive* e *extcanperceive* determinam o conjunto de percepções que podem ser geradas a partir da memória interna do agente e as percepções possíveis a partir do ambiente externo, respectivamente.

StoreAgentPerception

$$StoreAgent$$
$$AgentPerception$$
$$internalperactions, externalperactions : Actions$$
$$storecanperceive, extcanperceive : Env \rightarrow Actions \nrightarrow View$$

$$internalperactions \neq \varnothing$$
$$internalperactions \cup externalperactions = peractions$$
$$internalperactions \cap externalperactions = \varnothing$$
$$\mathsf{dom}\ storecanperceive = \{store\}$$
$$\forall\ env : Env;\ as : Actions \mid env \in \mathsf{dom}\ storecanperceive \bullet$$
$$(as \in (\mathsf{dom}(storecanperceiveenv)) \Rightarrow as = internalperactions)$$
$$\forall\ env : Env;\ as : Actions \mid env \in \mathsf{dom}\ extcanperceive \bullet$$
$$(as \in (\mathsf{dom}(extcanperceiveenv)) \Rightarrow as = externalperactions)$$

O esquema *StoreAgentAction* define que metas, percepções e ambiente influenciam o comportamento do agente.

StoreAgentAction

$$StoreAgent$$
$$AgentAction$$

O esquema *StoreAgentState* descreve como será selecionado o conjunto das próximas ações do agente. Neste caso, as percepções possíveis resultam da união dos conjuntos *possinternalpercepts* e *possexternalpercepts*.

StoreAgentState

$$StoreAgentPerception$$
$$StoreAgentAction$$
$$AgentState$$
$$possinternalpercepts, possexternalpercepts : View$$
$$extenv : Environment$$

$$possinternalpercepts = storecanperceive\ store\ internalperactions$$
$$possexternalpercepts = extcanperceive\ env\ externalperactions$$
$$posspercepts = possinternalpercepts \cup possexternalpercepts$$
$$extenv \cup store = env$$

A função *externaleffectinteraction* descreve interações e é similar à função *effectinteraction* definida previamente.

$$externaleffectinteraction : Env \nrightarrow \mathbb{P}\ Action \nrightarrow Env$$

O esquema *UpdateStore* descreve como a memória interna é atualizada pelo *StoreAgent*.

UpdateStore

$$StoreAgent$$
$$updatestore : View \rightarrow Env \rightarrow \mathbb{P}\ Goal \rightarrow \mathbb{P}\ Action \rightarrow Env$$

Considerando operações de *StoreAgent*, são especificados os esquemas $\Delta StoreAgentState$ e

StoreAgentInteracts.

$\Delta StoreAgentState$

$StoreAgentState$
$StoreAgentState'$
$\Delta AgentState$

O esquema *StoreAgentInteracts* mostra que, como resultado de uma interação com o ambiente, tanto o ambiente quanto a memória interna do agente são alterados. Assim, é possível modelar a situação onde informações aprendidas são armazenadas na memória do agente.

StoreAgentInteracts

$\Delta StoreAgentState$
$UpdateStore$
$AgentInteracts$

$extenv' = externaleffectinteraction\ extenv\ willdo$
$store' = updatestore\ actualpercepts'\ store\ goals\ willdo$

4.7 Considerações Finais

Neste capítulo foi apresentada uma visão geral do arcabouço SMART. Este arcabouço apresenta especificações formais para o ambiente, entidades, objetos, agentes, agentes autônomos e agentes com memória, conforme pode ser observado no diagrama apresentado na figura 4.1. Tomando como base estas camadas básicas, foi construído um modelo formal para a teoria de Senge. Este modelo é apresentado no próximo capítulo.

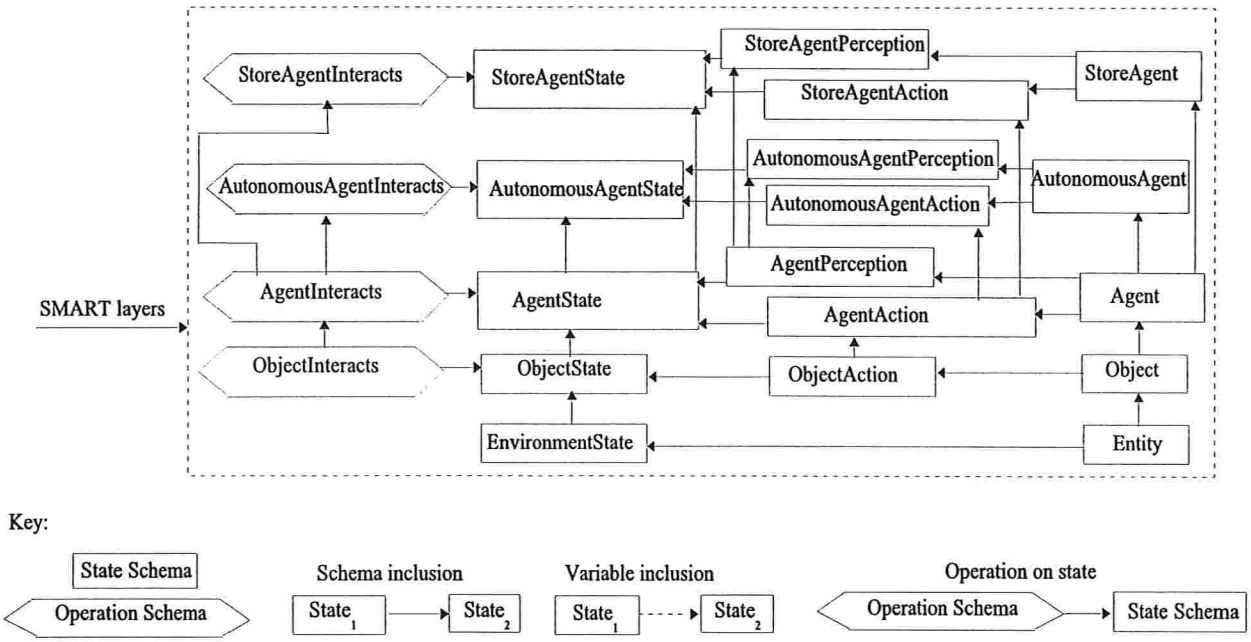


Figura 4.1: SMART - Diagrama de inclusão de esquemas.

CAPÍTULO 5

Um Arcabouço Formal para a Quinta Disciplina

Neste capítulo é apresentado um arcabouço formal[1] construído com base na teoria da Quinta Disciplina. Inicialmente, as características gerais do modelo são descritas. Em seguida, modelos de agentes mais sofisticados, que se baseiam nos modelos introduzidos no SMART, são apresentados. Na seção 5.3 é apresentado o modelo formal do agente que participa de uma organização que implementa a teoria da Quinta Disciplina, com a inclusão das cinco disciplinas propostas por Senge, e especificações relativas à adoção de metas e interações entre os agentes. Logo após, são apresentadas e discutidas propriedades e características do modelo formal. O material apresentado nesse capítulo, em forma resumida e simplificada, foi apresentado em [SS04b].

5.1 Introdução

No modelo construído para a teoria da Quinta Disciplina, é utilizada a mesma estratégia de modularização empregada no arcabouço SMART: com base em tipos mais básicos de agentes,

[1]Neste modelo toda denominação de tipos e variáveis se baseia em abreviações e concatenações de palavras da língua inglesa. Esta escolha não se deve a restrições da notação Z, apenas visa facilitar a publicação deste trabalho em revistas e jornais científicos internacionais.

novos agentes, mais complexos, são especificados.

Desta forma, o modelo pode ser visto como uma estrutura em camadas, conforme apresentado no diagrama de estrutura de esquemas apresentado na figura 5.1, onde a camada de mais alto nível corresponde ao modelo do agente que é membro de uma organização aprendiz, segundo a teoria de Senge.

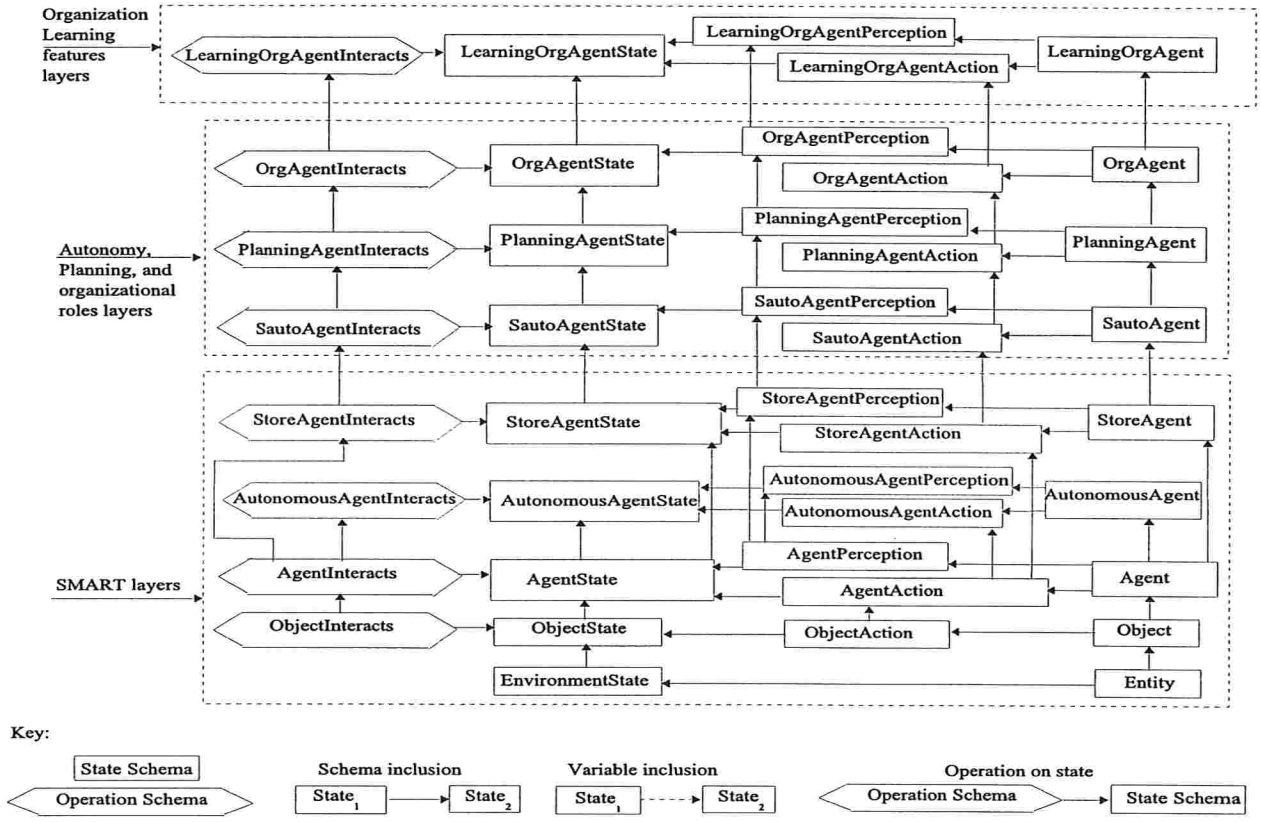


Figura 5.1: Modelo Formal da Quinta Disciplina - Diagrama de esquemas.

Para o desenvolvimento do modelo formal apresentado nesta tese foram utilizadas as seguintes referências sobre a notação Z: [Spi92, Spi89, Jac97, BSC94, Dil94, PST96, Mou01].

- O que foi formalizado

  Na construção deste modelo formal o autor desta tese tentou ser fiel ao máximo à teoria apresentada por Senge em [Sen90]. Entretanto, obviamente, nem todo o conteúdo da teoria de Senge pode ser traduzido por alguma expressão matemática. De um modo geral, o modelo apresentado se concentrou nas disciplinas e suas aplicações. Como conseqüência, esse modelo deve atender a uma série de requisitos, que são apresentados a seguir.

  Inicialmente são definidas abreviaturas para alguns conceitos associados à teoria de Senge.

  1. Componentes:

     $\texttt{Ambiente} == EV$
     $\texttt{Organização aprendiz} == LO$
     $\texttt{Agente membro da } LO == LAg$
     $\texttt{Equipe aprendiz} == LT$

  2. Disciplinas:

     $\texttt{Pensamento sistêmico} == STD$
     $\texttt{Domínio pessoal} == PMD$
     $\texttt{Modelos mentais} == MMD$
     $\texttt{Aprendizagem em equipe} == TLD$
     $\texttt{Visão compartilhada} == SVD$

  3. Elementos de apoio:

Motivações $==$ Mo
Idéias governantes (diretrizes) $==$ Gi
Visão $==$ Vi
Propósito $==$ Pu
Valores $==$ Va
Modelos mentais $==$ Mm
Processo de raciocínio $==$ Rpr
Crenças $==$ Bel
Modelos mentais compartilhados $==$ SMm
Idéias governantes (diretrizes) compartilhadas $==$ SGi
Visão compartilhada $==$ SVi
Propósito compartilhado $==$ SPu
Valores compartilhados $==$ SVa

(Tempo $\times$ Espaço $\times$ Entidade) Contexto $==$ $TSECont$
Eventos $==$ Ev
Percepção da realidade $==$ Rpe

Para a construção do modelo da Quinta Disciplina, adotou-se as seguintes hipóteses básicas:

1. O estado do ambiente inclui os seguintes elementos: LO's, entidades, LAg's e relacionamentos entre eles.
2. Toda LO é composta por um conjunto não vazio de LT.
3. Todo LT é composto por um conjunto não vazio de LAg.
4. Cada LAg tem um conjunto de Mo.
5. Cada LAg tem um conjunto de Mm.
6. Mm inclui Rpr e Bel.
7. Cada LAg interage com o ambiente.
   (a) Cada LAg percebe o ambiente.
   (b) Cada LAg é capaz de um conjunto de ações que afetam o ambiente.

(c) Cada *LAg* interage com outros *LAg*'s.

8. Cada *LAg* desenvolve *PMD*:

   (a) Cada *LAg* tem sua própria *Gi*.

   (b) Cada *Gi* inclui conjuntos de *Vi*, *Pu*, e *Va*.

   (c) Para cada *Gi*: *Vi*, *Pu*, e *Va* são consistentes: Objetivos em *Vi* são consistentes com objetivos em *Pu*. Nenhum objetivo em *Vi* contradiz qualquer *Va*. Nenhum objetivo em *Pu* contradiz qualquer *Va*.

   (d) As *Vi* de cada *LAg* são influenciadas pelas suas próprias *Mo*.

   (e) Cada *LAg* é capaz de conhecer sua tensão criativa, que corresponde à diferença entre a realidade percebida e a realidade desejada.

   (f) Cada *LAg* deseja reduzir sua tensão criativa, portanto, cada *LAg* constrói planos, incluindo séries de ações, visando realizar *Vi*.

9. cada LAg desenvolve *MMD*:

   (a) Cada *LAg* examina o papel de seus *Mm* em suas interações.

   (b) Cada *LAg* compreende que a realidade por ele percebida é influenciada por seu próprio *Mm*.

   (c) Cada *LAg* é capaz de expor seus *Mm* para outros *LAg*'s.

   (d) Cada *LAg* é capaz de consultar ou questionar o *Mm* de outros *LAg*'s.

   (e) A *SMm* emerge das interações entre os *LAg*'s, no contexto da *LO*.

10. Cada LAg desenvolve *TLD*:

    (a) Cada *LAg* é capaz de interagir com outros *LAg*'s no modo de diálogo.

    (b) Cada *LAg* é capaz de interagir com outros *LAg*'s no modo de discussão.

11. Cada *LAg* desenvolve *SVD*:

    (a) A *SGi* emerge das interações entre os *LAg*'s, no contexto da *LO*.

(b) A SGi inclui conjuntos de SVi, SPu, e SVa.

(c) A SVi emerge das Vi's dos LAg's, no contexto da LO.

(d) Todo LAg, no contexto da LO, tem uma atitude de compromisso em relação à SVi.

12. Cada LAg desenvolve STD:

(a) Cada LAg compreende as estruturas sistêmicas: laços de reforço, retardos, processos de equilíbrio, retroalimentação, e arquétipos.

Nas próximas seções são descritas as diversas camadas que compõem o modelo: agentes autônomos com memória, agentes que reconhecem planos, agentes organizacionais e, finalmente, agentes da Quinta Disciplina.

5.2 Refinando os Tipos de Agentes

5.2.1 Agentes Autônomos com Memória

Um agente autônomo é definido em [Ld01] como um refinamento do tipo *Agent*. Além disso, esses autores definem autonomia em função da existência de motivações próprias do agente. A partir de tais motivações, o agente é capaz de gerar suas próprias metas. No modelo formal para a Quinta Disciplina adotou-se o mesmo pressuposto. Entretanto, no trabalho reportado nesta tese, estamos interessados em modelar agentes autônomos que tenham uma memória interna. Assim, definimos aqui o tipo *SAutoAgent* que corresponde a um agente com motivações e memória.

SAutoAgent

$StoreAgent$

$motivations \neq \varnothing$

O esquema *SAutoAgentPerception* define que a percepção do agente é influenciada tanto por metas quanto por motivações.

SAutoAgentPerception

$$SAutoAgent$$
$$StoreAgentPerception$$
$$autoperceives : \mathbb{P}\, Motivation \rightarrow \mathbb{P}\, Goal \rightarrow View \rightarrow View$$

$$\mathrm{dom}\ autoperceives = \{motivations\}$$

De maneira similar, o comportamento do *SAutoAgent* é influenciado por suas motivações, metas, e pelo ambiente. A função que define o processo de seleção de ações, no caso desse agente, é definida por *autoact*.

SAutoAgentAction

$$SAutoAgent$$
$$StoreAgentAction$$
$$autoact : \mathbb{P}\, Motivation \rightarrow \mathbb{P}\, Goal \rightarrow View \rightarrow Env \rightarrow Actions$$

$$\mathrm{dom}\ autoact = \{motivations\}$$

O estado do *SAutoAgent* se baseia no estado do *StoreAgent*, conforme indicado pela inclusão do esquema *StoreAgentState*. A função de seleção de ações do agente é definida pela função *autoact*, que depende das motivações, metas, percepções e ambiente.

SAutoAgentState

$$SAutoAgentPerception$$
$$SAutoAgentAction$$
$$StoreAgentState$$

$$willdo = autoact\ motivations\ goals\ actualpercepts\ env$$

Considerando as operações deste tipo de agente, temos os esquemas $\Delta SAutoAgentState$ e *SAutoAgentInteracts*.

$\Delta SAutoAgentState$

$SAutoAgentState$
$SAutoAgentState'$

$\Delta StoreAgentState$
$autoperceives' = autoperceives$
$autoact' = autoact$

No esquema *SAutoAgentInteracts* é especificada a forma como novas percepções são obtidas pelo *SAutoAgent*: a função *storecanperceive* é aplicada em um novo estado interno, *store'*, com base nas capacidades perceptivas internas do agente. Da mesma forma, a função *extcanperceive* é aplicada em um novo estado do ambiente, *env'*, com base nas capacidades perceptivas externas do agente. As percepções que o agente tem possibilidade de captar correspondem, então, ao conjunto formado pela união das percepções internas e externas do agente. Portanto, aquilo que o agente efetivamente percebe é função de suas novas motivações e metas, e de novos potenciais perceptos e ambiente.

SAutoAgentInteracts

$\Delta SAutoAgentState$
$StoreAgentInteracts$

$possinternalpercepts' = storecanperceive\ store'\ internalperactions$
$possexternalpercepts' = extcanperceive\ env'\ externalperactions$
$posspercepts' = possinternalpercepts' \cup possexternalpercepts'$
$actualpercepts' = autoperceives\ motivations'\ goals'\ posspercepts'$
$willdo' = autoact\ motivations'\ goals'\ actualpercepts'\ env'$

5.2.2 Agentes e Planos

Um refinamento adicional de agentes, também apresentado em [dL96], define agentes que podem utilizar planos, tanto raciocinando sobre eles como produzindo-os. Tal agente tem um conjunto de planos associados a um conjunto de metas. Entretanto, no modelo apresentado nesta tese, o tipo *PlanningAgent* corresponde a um refinamento para o tipo *SAutoAgent*, ou seja, esse agente

também é autônomo e tem memória. Além disso, define-se também que tal agente tem recursos, e que as variáveis *instsreq* e *resourcesofplan* correspondem ao conjunto de recursos necessários para a realização de um plano. Essas variáveis são as mesmas apresentadas em [dL96], porém aqui elas são introduzidas diretamente no tipo *PlanningAgent* para representar que essas informações são internas a um dado agente. Adicionalmente, da mesma forma como é apresentado por esses autores, aqui também planos são considerados completos, e são definidos como uma seqüência de ações. Finalmente, a variável *plans* refere-se, no esquema *PlanningAgent*, aos planos atuais do agente.

$$Plan == \text{seq}\, Action$$
$$Resource == \mathbb{P}_1\, Entity$$

PlanningAgent

$$SAutoAgent$$

$$goals \subseteq \text{dom}\, planforgoal$$
$$\bigcup(\text{ran}\, planforgoal) = plans$$
$$plans = mapset\,(mapseq\, first)\,(\text{ran}\ instsreq)$$
$$\forall p : Plan \bullet resourcesofplan\, p = \bigcup(\text{ran}\,(mapseq\, second(instsreq\, p)))$$
$$\forall p : Plan \mid p \in plans \bullet \forall i : \mathbb{N} \mid i \leq \#plans \bullet \exists a : Action \mid a = head\,(i \upharpoonright p) \bullet a \in capableof$$

A percepção de um *PlanningAgent* corresponde à percepção de um agente do tipo *SAutoAgent*.

PlanningAgentPerception

$$PlanningAgent$$
$$SAutoAgentPerception$$

A ação desse tipo de agente é influenciada pelo ambiente, por suas motivações, metas, e por seus planos.

PlanningAgentAction

$$
\begin{array}{l}
PlanningAgent \\
SAutoAgentAction \\
planact : \mathbb{P}\, Motivation \rightarrow \mathbb{P}\, Goal \rightarrow \\
\quad \mathbb{P}\, Plan \rightarrow View \rightarrow Env \rightarrow Actions \\
\hline
\mathrm{dom}\, planact = \{motivations\}
\end{array}
$$

O esquema representando o estado do *PlanningAgent* é definido por intermédio da inclusão do esquema de estado do *SAutoAgent* e também depende de sua percepção e ação. A função de seleção de ações do agente é definida pela função *planact*, que depende das motivações, metas, planos, percepções e ambiente.

PlanningAgentState

$$
\begin{array}{l}
PlanningAgentPerception \\
PlanningAgentAction \\
SAutoAgentState \\
\hline
willdo = planact\ motivations\ goals\ plans\ actualpercepts\ env
\end{array}
$$

Para permitir a descrição das operações do *PlanningAgent* são especificados os esquemas $\Delta PlanningAgentState$ e *PlanningAgentInteracts*.

ΔPlanningAgentState

$$
\begin{array}{l}
PlanningAgentState \\
PlanningAgentState' \\
\hline
\Delta SAutoAgentState \\
planact' = planact
\end{array}
$$

Considera-se, para esse tipo de agente, que os planos não afetam as percepções potenciais (*posspercepts*) e efetivas (*actualpercepts*). Logo, a percepção do *PlanningAgent* é similar à especificada para o *SAutoAgent*. O processo de seleção da próxima ação dependerá também dos novos planos do agente.

PlanningAgentInteracts

$\Delta PlanningAgentState$
$SAutoAgentInteracts$

$willdo' = planact\ motivations'\ goals'\ plans'\ actualpercepts'\ env'$

5.2.3 Agentes Organizacionais

Nesta seção é apresentada a especificação de agentes que participam de organizações. Para tanto, inicialmente são introduzidas especificações para os tipos referentes à coletividades de agentes, tais como: grupos, equipes e organizações. Tais definições servirão como base para a especificação posterior de modelos para equipes e organizações que implementam a teoria de Senge.

Neste modelo, define-se que um grupo é um conjunto de indivíduos que compartilham um conjunto de recursos. Assim, em tal grupo não há a necessidade de compromisso mútuo em relação a uma meta conjunta. Por exemplo, quando em visita a um museu, um grupo de turistas compartilha suas dependências e obras de arte; um grupo de secretárias em um escritório compartilha as dependências do escritório, linhas telefônicas e outros equipamentos de trabalho. Recursos correspondem, neste trabalho, a entidades no ambiente, assim como definido em[dL96]. No esquema *Group*, foram introduzidas também algumas variáveis adicionais, por exemplo, *commongoals* e *learningmembers*, que não são usadas nesse momento, mas que serão usadas posteriormente no refinamento do tipo *Group* para a produção dos tipos *Team* e *LearningTeam*. A variável *membersgroup* corresponde, nesse esquema, ao conjunto de membros do grupo que são do tipo *PlanningAgent*. Além disso, uma parcela dos membros do grupo pode ser do tipo *LearningOrgAgent*. As restrições aplicadas na parte predicativa indicam que um grupo tem necessariamente um conjunto não vazio de recursos, e que deve haver mais de um membro no grupo. Adicionalmente, deve haver algum recurso dentre os disponíveis ao grupo, que seja necessário para a realização de algum dos planos dos membros do grupo.

Group

$membersgroup : \mathbb{P}\ PlanningAgent$
$resources : \mathbb{P}\ Resource$
$commongoals : \mathbb{P}\ Goal$
$developCommonGoals : \mathbb{P}\ PlanningAgent \rightarrow \mathbb{P}\ Goal$
$teamguidingideas : GuidingIdea$
$learningmembers : \mathbb{P}\ LearningOrgAgent$
$commonplans : PlanLib$
$developcommonplans : \mathbb{P}\ LearningOrgAgent \rightarrow \mathbb{P}\ Plan$

$resources \neq \varnothing$
$\#membersgroup > 1$
$\exists\, r : Resource \mid r \in resources \bullet (\forall\, a : PlanningAgent \mid$
$\quad a \in membersgroup \bullet (\exists\, p : Plan \mid p \in a.plans \bullet$
$\quad\quad r \in \{a.resourcesofplan\ p\}))$
$\mathsf{dom}\ developCommonGoals = \{membersgroup\}$
$\mathsf{ran}\ developCommonGoals = \{commongoals\}$

Um refinamento inicial para coletividades de agentes é representado pelo tipo *Team*, que corresponde ao modelo de uma equipe. Neste modelo, considera-se que um *Team* estende o tipo *Group* ao incorporar um conjunto de metas que são compartilhadas entre os seus membros. Desta forma, em uma equipe existe um conjunto de recursos compartilhados e, adicionalmente, um conjunto de metas que também estão representadas no conjunto *goals* de cada agente membro da equipe.

Team

Group

$commongoals \neq \varnothing$
$\forall\, g : Goal \mid g \in commongoals \bullet (\forall\, a :$
$\quad PlanningAgent \mid a \in membersgroup \bullet g \in a.goals)$

Com o objetivo de especificar organizações de agentes, o tipo *Organization* será definido adiante. A estratégia adotada neste trabalho é apresentar tipos mais genéricos de organizações e depois, incrementalmente, introduzir a especificação do tipo formal correspondente à Organiza-

ção Aprendiz[2]. Entretanto, primeiramente, há a necessidade de se especificar alguns tipos que servirão de apoio para tal definição. Desta forma, assume-se como hipótese neste trabalho que toda organização é composta por uma estrutura, um conjunto de equipes, papéis, normas e metas organizacionais.

A estrutura da organização é especificada no esquema *OrgStructure* como um grafo conexo, no qual cada nó corresponde a uma equipe e cada aresta corresponde a um relacionamento entre as equipes por ela conectadas.

OrgStructure

$$nodes : \mathbb{P}\ Team$$
$$edges : Team \rightarrow Team$$

$$\mathsf{dom}\ edges \subseteq nodes$$
$$\mathsf{ran}\ edges \subseteq nodes$$

Papéis são especificados no esquema *Role* que inclui conjuntos de ações que precisam ser executadas, conjuntos de metas, recursos gerenciados pelo agente que desempenhar tal papel, e um nível de autonomia organizacional. Tal nível indica a liberdade associada ao papel para a definição de novas tarefas visando realizar as metas associadas ao papel, conforme estipulado nas normas da organização.

Role

$$ident : Literal$$
$$rolegoals : \mathbb{P}\ Goal$$
$$assignedActivities : \mathbb{P}\ Plan$$
$$skillsRequired : \mathbb{P}\ Action$$
$$managedresources : \mathbb{P}\ Resource$$
$$roleorganizationalautonomy : Plan \times Norm \rightarrow OrgAutonomy$$

$$mapset\ (first)(\mathsf{dom}\ roleorganizationalautonomy) = assignedActivities$$

Papéis têm identificações, aqui definidas como uma seqüência de caracteres. Aqui é definido

[2]O termo é usado aqui como referência à organização que implementa a teoria de Senge.

apenas o identificador *facilitator*, que será utilizado mais adiante na especificação de diálogo e discussão.

$Symbol ::= AsciiSymbol \mid UnicodeSymbol$
$String == \text{seq}\, Symbol$
$Literal ::= facilitator \mid string\langle\langle String\rangle\rangle \mid comb\langle\langle Literal \times Literal\rangle\rangle$

Neste modelo, o nível de autonomia organizacional é definido apenas como: nenhum (*none*), baixo (*low*), médio (*medium*) e alto (*high*). Desta forma, o agente que, por exemplo, desempenhe um papel para o qual o conjunto imagem da função *roleorganizationalautonomy* seja igual a *none*, não tem flexibilidade na escolha das ações ou tarefas que deverá executar para atingir as metas associadas a este papel, devendo restringir-se aos planos definidos em *assignedActivities*. Além disso, um mesmo agente, ao desempenhar um dado papel, pode ter maior ou menor autonomia organizacional dependendo dos planos específicos e das normas relativas aos mesmos na organização. A especificação detalhada de autonomia das categorias baixa, média e alta não é relevante para este trabalho, apenas indica graus crescentes de liberdade na seleção de tarefas. Portanto, será omitido tal detalhamento.

$OrgAutonomy ::= none \mid low \mid medium \mid high$

O tipo *Norm* modela as normas que são válidas em uma dada organização. No modelo aqui apresentado, utiliza-se apenas uma especificação simplificada onde se considera que uma norma apresenta pré-condições para sua aplicação e possue um corpo, composto por ações[3].

Norm

$precond : \mathbb{P}_1\, Attribute$
$body : \mathbb{P}_1\, Action$

Nesse ponto é possível especificar o esquema *Organization* da seguinte maneira: *orggoals* cor-

[3]Uma especificação mais detalhada de norma organizacional está fora do escopo deste trabalho.

responde às metas que são definidas pelo corpo diretivo da organização, quando da criação desta. Tais metas, é claro, podem ser revisadas periodicamente. A organização possui estrutura, conjuntos de equipes, e define-se que o conjunto de seus membros corresponde exatamente aos agentes membros das equipes que a compõem.

Organization

$$
\begin{array}{l}
norms : \mathbb{P}\, Norm \\
roles : \mathbb{P}\, Role \\
structure : OrgStructure \\
teams : \mathbb{P}\, Team \\
membersorg : \mathbb{P}\, PlanningAgent \\
orggoals : \mathbb{P}\, Goal \\
\hline
orggoals \neq \varnothing \\
teams \neq \varnothing \\
teams = structure.nodes \\
membersorg = \bigcup \{t : Team \mid t \in teams \bullet t.membersgroup\}
\end{array}
$$

Um refinamento adicional para o tipo *Organization* envolve a definição de uma organização formal[4]. Neste modelo de organização há um conjunto não vazio de normas e papéis. Além disso, todo membro dessa organização desempenha pelo menos um papel.

FormalOrg

$$
\begin{array}{l}
Organization \\
regset : RegistrySet \\
perform : PlanningAgent \times Role \times RegistrySet \rightarrow Boolean \\
\hline
norms \neq \varnothing \\
roles \neq \varnothing \\
regset.regs \neq \varnothing \\
\forall\, m : PlanningAgent \mid m \in membersorg \bullet \\
\quad (\exists\, r : Role \mid r \in roles \bullet perform(m, r, regset) = positive)
\end{array}
$$

Neste tipo de organização, o *RegistrySet* armazena informação relativa aos agentes e os res-

[4]Formal, aqui, no sentido de que a organização possui posições e normas definidas.

pectivos papéis por eles desempenhados. Assim, o *RegistrySet* corresponde a um tipo de registro geral no contexto da organização.

Registry

$$reg : PlanningAgent \times Role$$

RegistrySet

$$regs : \mathbb{P}\ Registry$$

$$Boolean ::= positive \mid negative$$

Finalmente, dando prosseguimento à definição incremental via refinamento de tipos de agentes, é introduzido nesse ponto o agente organizacional, conforme especificado no esquema *OrgAgent*. Esse tipo de agente estende o *PlanningAgent*, apresentado previamente, e tem a capacidade de participar de um conjunto de organizações. Ele pode desempenhar, em cada organização, um conjunto de papéis. Além disso, esse agente possui um conjunto de planos que inclui tanto seus planos pessoais quanto aqueles relativos aos papéis por ele desempenhados. De modo similar, o conjunto de metas desse agente inclui tanto suas metas individuais quanto as metas associadas a cada um dos papéis por ele desempenhados.

OrgAgent

$$PlanningAgent$$

$$orgs \neq \varnothing$$
$$roles = \varnothing \Rightarrow (allplans = plans \land allgoals = goals)$$
$$roles \neq \varnothing \Rightarrow ((allplans = \bigcup \{r : Role \mid r \in roles \bullet r.assignedActivities\} \cup plans) \land$$
$$(allgoals = \bigcup \{r : Role \mid r \in roles \bullet r.rolegoals\} \cup goals))$$
$$\forall p : allplans \bullet \forall i : \mathbb{N} \mid i \leq \#allplans \bullet \exists a : Action \mid a = head\,(i \upharpoonright p) \bullet a \in capableof$$

Como conseqüência, a percepção deste agente é função de suas motivações, metas, ambiente e, adicionalmente, de seus papéis. De fato, para cada papel existe um conjunto de recursos associados

(*managedresources*), que devem ser gerenciados pelo agente que o estiver desempenhando. Esses recursos fazem parte do ambiente percebido pelo agente e requerem sua atenção especial em relação a itens como: quantidade, qualidade, etc.. Portanto, a percepção do ambiente pode ser influenciada, dependendo do fato de existirem ou não recursos gerenciados pelo agente no ambiente. Além disso, as metas consideradas em *orgperceives* já incluem tanto metas individuais, quanto metas associadas aos papéis desempenhados pelo agente.

OrgAgentPerception

$$
\begin{array}{l}
OrgAgent \\
PlanningAgentPerception \\
orgperceives : \mathbb{P}\, Motivation \rightarrow \mathbb{P}\, Role \rightarrow \mathbb{P}\, Goal \rightarrow Env \rightarrow View \\
\hline
\mathsf{dom}\, orgperceives = \{motivations\} \\
\mathsf{dom}(orgperceives\ motivations) = \{roles\} \\
\mathsf{dom}(orgperceives\ motivations\ roles) = \{allgoals\}
\end{array}
$$

De modo similar ao exposto acima, a ação deste agente é influenciada por suas motivações, metas individuais e metas associadas aos papéis por ele desempenhados. Além disso, os planos considerados na função *orgact* já incluem tanto planos individuais, quanto planos associados aos papéis desempenhados pelo agente.

OrgAgentAction

$$
\begin{array}{l}
OrgAgent \\
PlanningAgentAction \\
orgact : \mathbb{P}\, Motivation \rightarrow \mathbb{P}\, Goal \rightarrow \mathbb{P}\, Plan \rightarrow View \rightarrow Env \rightarrow Actions \\
\hline
\mathsf{dom}\, orgact = \{motivations\} \\
\mathsf{dom}(orgact\ motivations) = \{allgoals\} \\
\mathsf{dom}(orgact\ motivations\ allgoals) = \{allplans\}
\end{array}
$$

O estado do *OrgAgent* depende de suas percepções, ações e é uma extensão do estado do *PlanningAgent*. Nesse esquema de estado, fica evidenciada a influência adicional das metas e planos associados aos papéis do agente nas percepções e tomada de decisão do agente.

OrgAgentState

$$
\begin{array}{l}
OrgAgentPerception \\
OrgAgentAction \\
PlanningAgentState \\
\hline
actualpercepts = orgperceives\ motivations\ roles\ allgoals\ posspercepts \\
willdo = orgact\ motivations\ allgoals\ allplans\ actualpercepts\ env
\end{array}
$$

Nos esquemas $\Delta OrgAgentState$ e *OrgAgentInteracts* são especificadas as operações deste agente. No esquema abaixo, as funções *orgperceives* e *orgact* permanecem inalteradas, ou seja, uma operação não altera a percepção e a ação do agente.

$\Delta OrgAgentState$

$$
\begin{array}{l}
OrgAgentState \\
OrgAgentState' \\
\hline
\Delta PlanningAgentState \\
orgperceives' = orgperceives \\
orgact' = orgact
\end{array}
$$

Interações do agente com o ambiente envolvem mudanças em seu estado. Neste caso, mudanças em quaisquer dos planos reconhecidos pelo agente afetam as percepções efetuadas. Além disso, alterações nas percepções e planos afetam os resultados da função de seleção de ações do agente.

OrgAgentInteracts

$$
\begin{array}{l}
\Delta OrgAgentState \\
PlanningAgentInteracts \\
\hline
actualpercepts' = orgperceives\ motivations'\ roles' \\
\quad allgoals'\ posspercepts' \\
willdo = orgact\ motivations'\ allgoals'\ allplans'\ actualpercepts'\ env'
\end{array}
$$

Neste ponto, a estrutura em camadas do modelo, incorpora os níveis de planejamento e organizacional conforme apresentado no diagrama de estrutura de esquemas apresentado na figura 5.2.

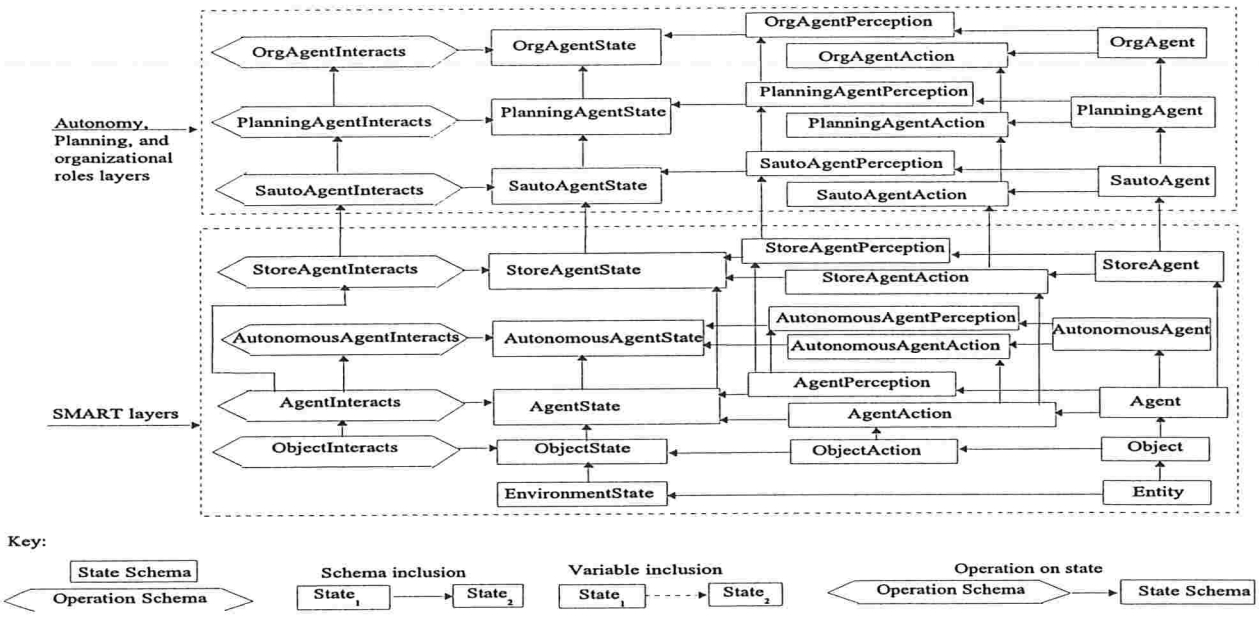


Figura 5.2: Níveis Planejamento e Organizacional - Diagrama de esquemas.

5.3 Agentes Formais da Quinta Disciplina

O objetivo desta definição incremental, via refinamento de tipos de agentes, é definir um tipo mais especializado de agente. Esse agente deve ser capaz de participar de uma organização aprendiz que obedeça aos requisitos descritos na teoria de Senge, especificados formalmente neste modelo.

Assim, nesta seção é apresentado o *LearningOrgAgent*, que é um agente organizacional que desenvolve o domínio pessoal, tem modelos mentais e desenvolve capacidades de pensamento sistêmico. Entre as motivações deste agente temos a redução da tensão criativa e um compromisso com uma visão clara da realidade. Na apresentação desta formalização, as disciplinas de Senge foram divididas em dois grupos: intrapessoais e interpessoais, de acordo com o tipo de capacidade que o agente pode desenvolver ao praticar tais disciplinas. O primeiro grupo lida principalmente com os estados internos (e suas representações) do agente, enquanto o segundo lida com interações

entre *LearningOrgAgents*.

A especificação deste modelo foi verificada sintaticamente[5] utilizando-se o produto ZTC versão 2.03 [ZTC03].

5.3.1 Disciplinas: Ações e Princípios

De acordo com Senge, todas as disciplinas têm ações e princípios a elas associados. Desta forma, um agente que desenvolve uma dada disciplina deve conhecer seus princípios e ser capaz de executar as ações associadas. Assim, são declaradas abaixo as variáveis globais *PMActions* e *PMPrinciples* que correspondem a ações e princípios de domínio pessoal, respectivamente. De modo similar, *STActions* e *STPrinciples*; *MMActions* e *MMPrinciples*; *SVActions* e *SVPrinciples*; *LTActions* e *LTPrinciples*; correspondem às ações e princípios, respectivamente, de pensamento sistêmico, modelos mentais, visão compartilhada e aprendizagem em equipe.

$$
\begin{array}{|l}
AllDisciplinesActions, PMActions : \mathbb{P}\, Action \\
STActions, MMActions, SVActions, LTActions : \mathbb{P}\, Action \\
\hline
PMActions \neq \varnothing \land STActions \neq \varnothing \\
MMActions \neq \varnothing \land SVActions \neq \varnothing \land LTActions \neq \varnothing \\
AllDisciplinesActions = PMActions \cup STActions \cup \\
\quad MMActions \cup SVActions \cup LTActions
\end{array}
$$

$$
\begin{array}{|l}
AllDisciplinesPrinciples, PMPrinciples : \mathbb{P}\, Attribute \\
STPrinciples, MMPrinciples, SVPrinciples, LTPrinciples : \mathbb{P}\, Attribute \\
\hline
PMPrinciples \neq \varnothing \land STPrinciples \neq \varnothing \land MMPrinciples \neq \varnothing \\
SVPrinciples \neq \varnothing \land LTPrinciples \neq \varnothing \\
AllDisciplinesPrinciples = PMPrinciples \cup STPrinciples \cup \\
\quad MMPrinciples \cup SVPrinciples \cup LTPrinciples
\end{array}
$$

Assim, o esquema *DisciplinesActionsAndPrinciples* descreve o conjunto de ações e princípios das disciplinas.

[5] Verificação de tipos.

DisciplinesActionsAndPrinciples

$$
\begin{array}{l}
actions : \mathbb{P}\ Action \\
principles : \mathbb{P}\ Attribute \\
\hline
actions \subseteq AllDisciplinesActions \\
principles \subseteq AllDisciplinesPrinciples \\
actions \neq \varnothing \land principles \neq \varnothing
\end{array}
$$

A seguir, é declarada a função *entityLearnDisciplinesActionsAndPrinciples*. Esta função define que o desenvolvimento das disciplinas por um agente envolve a instanciação de uma nova entidade que é capaz de desenvolver as cinco disciplinas. Esta função associa a um par, contendo uma entidade e um conjunto de ações e princípios, uma nova entidade. Para cada uma das variáveis da nova entidade são associados os mesmos valores existentes na variável correspondente da entidade original, com exceção das capacidades e conhecimentos, que são enriquecidos com as novas ações e princípios referentes às disciplinas de Senge.

$$
\begin{array}{|l}
entityLearnDisciplinesActionsAndPrinciples : (Entity \\
\quad \times DisciplinesActionsAndPrinciples) \rightarrow Entity \\
\hline
\forall\, old, new : Entity;\ actionAndPrinciple : \\
\quad DisciplinesActionsAndPrinciples \bullet \\
\quad (entityLearnDisciplinesActionsAndPrinciples(old, \\
\quad actionAndPrinciple) = new \Leftrightarrow \\
\quad (new.goals = old.goals \wedge \\
\qquad new.attributes = old.attributes \wedge \\
\qquad new.motivations = old.motivations \wedge \\
\qquad new.ownedresources = old.ownedresources \wedge \\
\qquad new.instsreq = old.instsreq \wedge \\
\qquad new.resourcesofplan = old.resourcesofplan \wedge \\
\qquad new.plans = old.plans \wedge \\
\qquad new.planforgoal = old.planforgoal \wedge \\
\qquad new.orgs = old.orgs \wedge \\
\qquad new.roles = old.roles \wedge \\
\qquad new.allplans = old.allplans \wedge \\
\qquad new.allgoals = old.allgoals \wedge \\
\qquad new.capableof = old.capableof \cup \\
\qquad\quad actionAndPrinciple.actions \wedge \\
\qquad new.store = old.store \cup \\
\qquad actionAndPrinciple.principles \wedge \\
\qquad actionAndPrinciple.principles \neq \varnothing \wedge \\
\qquad actionAndPrinciple.actions \neq \varnothing))
\end{array}
$$

5.3.2 O Agente da Quinta Disciplina

Finalmente, é possível então, a especificação do modelo formal para o agente da Quinta Disciplina, o *LearningOrgAgent*. Este agente é capaz de desenvolver as disciplinas de Senge. Desta forma, as restrições que se aplicam a esse esquema apenas enfatizam a necessidade de que as capacidades e conhecimentos associados às disciplinas sejam representados por conjuntos não vazios. Além disso, como conseqüência da inclusão do esquema *OrgAgent*, esse agente também é autônomo, possui memória e capacidades de lidar com planos e de desempenhar papéis em organizações.

LearningOrgAgent

OrgAgent

$personalMasteryCapabilities \neq \varnothing$
$mentalModelsCapabilities \neq \varnothing$
$systemsThinkingCapabilities \neq \varnothing$
$learningTeamCapabilities \neq \varnothing$
$buildingSharedVisionCapabilities \neq \varnothing$
$personalMasteryCapabilities \subset capableof$
$mentalModelsCapabilities \subset capableof$
$systemsThinkingCapabilities \subset capableof$
$learningTeamCapabilities \subset capableof$
$buildingSharedVisionCapabilities \subset capableof$

Nas próximas seções serão apresentadas formalizações referentes a cada disciplina da teoria da Quinta Disciplina: modelos mentais, domínio pessoal, aprendizagem em equipe, pensamento sistêmico e visão compartilhada.

5.3.3 Modelos Mentais

A disciplina de modelos mentais envolve o desenvolvimento de capacidades de reflexão e consulta, de forma a, primeiro, tornar o agente ciente das hipóteses e generalizações que influenciam tanto sua compreensão quanto suas interações com o ambiente; e, segundo, interagir com outros agentes para compartilhar visões pessoais e aprender os pressupostos dos demais agentes sobre diversas questões.

Para efetuar a formalização de modelos mentais, inicialmente, é apresentada aqui uma formalização para crenças conforme apresentado em [dL01]. Dessa forma, as crenças do agente são representadas como afirmações, negações, ou conjunções de termos de lógica de predicados de primeira ordem. As crenças do agente correspondem ao tipo *AgentBelief* e suas crenças expostas, ou seja, comunicadas, correspondem ao tipo *ExposedBelief*.

$$
\begin{aligned}
AGLiteral &::= pos\langle\langle Atom\rangle\rangle \\
&\mid negate\langle\langle Atom\rangle\rangle \\
AGBelief &::= agliteral\langle\langle AGLiteral\rangle\rangle \\
&\mid conjunct\langle\langle AGBelief \times AGBelief\rangle\rangle \\
AgentBelief &::= agbeliefs\langle\langle AGBelief\rangle\rangle \\
ExposedBelief &::= expbeliefs\langle\langle AGBelief\rangle\rangle
\end{aligned}
$$

Em seguida, há a necessidade de se especificar os processos de raciocínio e reflexão do agente.

O processo de raciocínio, *ReasoningProcess*, é o tipo que permite a realização da atividade que Senge caracteriza como a exibição da "escada da inferência": "ladder of inference - a common mental pathway of increasing abstraction, often leading to misguided beliefs" [6], ou seja, um atalho no raciocínio, de abstração crescente, que pode com freqüência levar a crenças errôneas. Esse tipo é especificado como uma seqüência não vazia de crenças. São também introduzidos os tipos *ReflectedBeliefReasoning* e *ExposedBeliefReasoning*, que correspondem à reflexão e exposição de um dado processo de raciocínio, respectivamente.

Adicionalmente, em [dL01] as capacidades de modelagem dos agentes são representadas em uma série de esquemas, mostrando que um agente é capaz de modelar entidades, objetos, agentes, etc.. No modelo apresentado nesta tese, define-se que um *LearningOrgAgent* é também capaz de modelar todas estas entidades.

Inicialmente, seguindo [dL01], define-se representações de modelos de entidades, objetos, agentes, agentes autônomos, agentes com memória. Além disso, são também incluidos modelos para agentes dos tipos *SAutoAgent*, *PlanningAgent*, e *OrgAgent*.

[6] Ver [SKR+94, p. 243]

$ReasoningProcess == \text{seq}_1\ AGBelief$
$ReflectedBeliefReasoning == AGBelief \times ReasoningProcess$
$ExposedBeliefReasoning == AGBelief \times ReasoningProcess$
$BeliefAndReasoning == AGBelief \times ReasoningProcess$
$EntityModel == Entity$
$ObjectModel == Object$
$NeutralObjectModel == NeutralObject$
$AgentModel == Agent$
$ServerAgentModel == ServerAgent$
$StoreAgentModel == StoreAgent$
$SAutoAgentModel == SAutoAgent$
$PlanningAgentModel == PlanningAgent$
$OrgAgentModel == OrgAgent$

Os modelos mentais também devem incluir os relacionamentos entre os vários componentes presentes no ambiente. Tais componentes correspondem exatamente às entidades, objetos, e aos diversos tipos de agentes. Os relacionamentos, que correspondem ao tipo *ComponentRelationship*, envolvem pares de componentes, neste modelo. O tipo *ComponentRelationshipModel*, que corresponde ao modelo de um relacionamento, é então, simplesmente definido por abreviação, indicando que o relacionamento entre esses componentes e o modelo do relacionamento são do mesmo tipo.

$$
\begin{aligned}
Component ::= {} & compEntity\langle\langle Entity\rangle\rangle \\
| {} & compObject\langle\langle Object\rangle\rangle \\
| {} & compNeutralObject\langle\langle NeutralObject\rangle\rangle \\
| {} & compAgent\langle\langle Agent\rangle\rangle \\
| {} & compServerAgent\langle\langle ServerAgent\rangle\rangle \\
| {} & compStoreAgent\langle\langle StoreAgent\rangle\rangle \\
| {} & compSAutoAgent\langle\langle SAutoAgent\rangle\rangle \\
| {} & compPlanningAgent\langle\langle PlanningAgent\rangle\rangle \\
| {} & compOrgAgent\langle\langle OrgAgent\rangle\rangle
\end{aligned}
$$

$ComponentRelationship ::= compRelationship\langle\langle Component \times Component\rangle\rangle$
$ComponentRelationshipModel == ComponentRelationship$

No esquema *BelModel* são incluídos os diversos modelos dos componentes que podem estar presentes no ambiente, modelos dos vários tipos de relacionamentos envolvendo tais componentes, e as crenças do agente. Nesse esquema, a variável *models* engloba todos os modelos de componentes que o agente possui. Além disso, também é especificado que, se o agente possuir mais de um modelo de componente, então ele deve também ter pelo menos um modelo de relacionamento. Além disso, define-se também que o conjunto de modelos de agentes autônomos com memória é um subconjunto dos modelos de agentes que apenas possuem memória. Esses, por sua vez, formam um subconjunto dos modelos dos agentes, e assim sucessivamente.

BelModel

$$
\begin{aligned}
&models : \mathbb{P}\, EntityModel \\
&agentbeliefs : \mathbb{P}\, AGBelief \\
&modelentities : \mathbb{P}\, EntityModel \\
&modelobjects : \mathbb{P}\, ObjectModel \\
&modelneutralobjects : \mathbb{P}\, NeutralObjectModel \\
&modelagents : \mathbb{P}\, AgentModel \\
&modelserveragents : \mathbb{P}\, ServerAgentModel \\
&modelstoreagents : \mathbb{P}\, StoreAgentModel \\
&modelsautoagents : \mathbb{P}\, SAutoAgentModel \\
&modelplanningagents : \mathbb{P}\, PlanningAgentModel \\
&modelorgagents : \mathbb{P}\, OrgAgentModel \\
&modelcomprelationship : \mathbb{P}\, ComponentRelationshipModel \\
\hline
&models = \bigcup \{modelentities, modelobjects, \\
&\quad modelneutralobjects, modelagents, modelserveragents, \\
&\quad modelstoreagents, modelsautoagents, \\
&\quad modelplanningagents, modelorgagents\} \\
&\#models > 1 \Rightarrow modelcomprelationship \neq \varnothing \\
&modelsautoagents \subseteq modelstoreagents \subseteq \\
&\quad modelagents \subseteq modelobjects \subseteq modelentities \\
&modelobjects = modelagents \cup modelneutralobjects \\
&modelserveragents = modelagents \setminus modelsautoagents
\end{aligned}
$$

Com base nos tipos definidos até aqui, é possível então a definição dos modelos mentais do agente que não incluam interações com agentes do tipo *LearningOrgAgent*. Desta forma, os mo-

delos mentais do agente, indicados pela variável *agentmodels* são do tipo *BelModel* e incluem as crenças do agente e seus modelos de entidades, objetos, e dos diversos tipos de agentes, sendo o tipo *OrgAgent* o mais sofisticado deles. Além disso, nesse esquema também é definido que há a necessidade de se desenvolver capacidades de reflexão (*developReflectionSkillsActions*) e argumentação (*developAdvocacySkillsActions*) e que estas fazem parte do conjunto de capacidades associado ao desenvolvimento da disciplina de modelos mentais (*MMActions*). São também definidas as funções *reflection* e *advocacy* que correspondem, respectivamente, aos processos de reflexão e argumentação do agente. A partir de triplas que incluem motivações, ações, crenças e um dado processo de raciocínio do agente, essas funções mapeiam as crenças e processo de raciocínio que são, respectivamente, sujeitos à introspecção e exposição por parte do agente.

IntraPersonalMentalModel

$$
\begin{array}{l}
agentmodels : BelModel \\
developReflectionSkillsActions : \mathbb{P}\, Action \\
developAdvocacySkillsActions : \mathbb{P}\, Action \\
intraPersonalMentalModelPrinciples : \mathbb{P}\, Attribute \\
reflection : \mathbb{P}\, Motivation \times \mathbb{P}\, Action \times \\
\quad BeliefAndReasoning \nrightarrow ReflectedBeliefReasoning \\
advocacy : \mathbb{P}\, Motivation \times \mathbb{P}\, Action \times \\
\quad BeliefAndReasoning \nrightarrow ExposedBeliefReasoning \\
\hline
developReflectionSkillsActions \cup \\
\quad developAdvocacySkillsActions = MMActions
\end{array}
$$

Em complementação ao especificado acima, o esquema *InterPersonalMentalModel* define os modelos mentais do agente que incluem interações envolvendo agentes do tipo *LearningOrgAgent*. De acordo com o esquema *InterPersonalMentalModel* o *LearningOrgAgent* deve questionar outros agentes, de modo a aprender seus modelos mentais.

InterPersonalMentalModel

$$
inquiry : LearningOrgAgent \nrightarrow ExposedBelief
$$

No início dessa seção foram definidos modelos de diversos componentes, incluindo entidades, objetos e agentes com diversos níveis de sofisticação, abrangendo até o tipo *OrgAgent*. De modo similar, nesse momento é definido um modelo para o tipo *LearningOrgAgent*, também por meio de abreviação. Além disso, tanto a representação de componentes no ambiente quanto os possíveis relacionamentos entre eles são estendidos para incluir o tipo *LearningOrgAgent*.

$$LearningOrgAgentModel == LearningOrgAgent$$
$$LearnComponent ::= compLearningOrgAgent\langle\langle LearningOrgAgent\rangle\rangle \mid compGeneral\langle\langle Component\rangle\rangle$$
$$LearnComponentRelationship ::= compLearnRelationship \langle\langle LearnComponent \times LearnComponent\rangle\rangle$$
$$LearnComponentRelationshipModel == LearnComponentRelationship$$

Além disso, são declarados os tipos básicos *TrustLOrgAgentInter*, *NonTrustLOrgAgentInter* e *LOrgAgentInteracting*. Estes tipos correspondem, respectivamente, a interações confiáveis, interações não confiáveis, e interações em geral envolvendo agentes do tipo *LearningOrgAgent*. São também definidos modelos para as interações (*LOrgAgentInteractingModel*), e modelos para agentes que são confiáveis (*TrustLOrgAgentInterModel*) e não confiáveis (*NonTrustLOrgAgentInterModel*) em suas interações.

$$[LOrgAgentInteracting, TrustLOrgAgentInter, NonTrustLOrgAgentInter]$$
$$LOrgAgentInteractingModel == LOrgAgentInteracting$$
$$TrustLOrgAgentInterModel == TrustLOrgAgentInter$$
$$NonTrustLOrgAgentInterModel == NonTrustLOrgAgentInter$$

O esquema *LearnBelModel* estende as definições apresentadas no esquema *BelModel* por meio da inclusão dos componentes, modelos e relacionamentos referentes ao tipo *LearningOrgAgent*. Além disso, são também incluídas referências às interações em geral, e às interações confiáveis e não confiáveis, em particular, envolvendo esses tipos de agentes. Adicionalmente, este esquema inclui a função *modelbelmodel*, que representa a capacidade de um dado agente de modelar os

modelos de outros agentes[7].

LearnBelModel

$$
\begin{array}{l}
BelModel \\
modellearningorgagents : \mathbb{P}_1\ LearningOrgAgentModel \\
modelbelmodel : Agent \nrightarrow LearningOrgAgent \\
modellearncomprelationship : \mathbb{P}\ LearnComponentRelationshipModel \\
modelLOrgAgInts : \mathbb{P}\ LOrgAgentInteractingModel \\
modelTrustLOrgAgInts : \mathbb{P}\ TrustLOrgAgentInterModel \\
modelNonTrustLOrgAgInts : \mathbb{P}\ NonTrustLOrgAgentInterModel \\
\hline
models = modelentities \cup modelobjects \cup \\
\quad modelneutralobjects \cup modelagents \cup \\
\quad modelserveragents \cup modelstoreagents \cup \\
\quad modelsautoagents \cup modelplanningagents \cup \\
\quad modelorgagents \cup modellearningorgagents \\
\#models > 1 \Rightarrow \\
\quad modellearncomprelationship \neq \varnothing \\
\mathrm{dom}\ modelbelmodel \subseteq modelagents
\end{array}
$$

Desta forma, no esquema *PeerLOAgMentalModel* temos a especificação de todos os tipos de modelos que um *LearningOrgAgent* pode representar, incluindo modelos de agentes que também sejam do mesmo tipo que o seu. Além disso, esse esquema também inclui o questionamento de outros agentes visando aprender seus modelos mentais.

PeerLOAgMentalModel

$$
\begin{array}{l}
LearnBelModel \\
InterPersonalMentalModel
\end{array}
$$

Concluindo, no esquema *MentalModel* são incluídos todos os tipos de modelos mentais que um agente pode ter e, adicionalmente, as funções de reflexão, argumentação e questionamento, bem como as capacidades necessárias e princípios associados ao desenvolvimento da disciplina de modelos mentais.

[7]Esta função é similar à *modelsociologicalagents* definida em [dL01, p. 106].

MentalModel

$$IntraPersonalMentalModel$$
$$PeerLOAgMentalModel$$

5.3.4 Domínio Pessoal

A disciplina de domínio pessoal é um processo que se baseia na avaliação contínua e clara da realidade, e em uma visão pessoal. Essa visão pessoal, por sua vez, corresponde a metas do indivíduo, representando a imagem do futuro que este deseja realizar. Esse processo gera uma força interna, denominada por Senge de tensão criativa. A resolução desta tensão deve significar um esforço para tornar a realidade mais próxima da visão pessoal [Sen90].

No modelo aqui apresentado, considera-se que tanto visão (*Vision*) quanto propósito (*Purpose*) correspondem a conjuntos de metas, e que valores (*Value*) correspondem a restrições de comportamento (*BehavioralConstraint*). Estas restrições são especificadas como um conjunto de crenças. Adicionalmente, define-se o tipo *RealityVision* como uma abreviação de um conjunto do tipo *View*, visando representar uma visão da realidade.

Vision

$$visions : \mathbb{P}\, Goal$$

Purpose

$$purposes : \mathbb{P}\, Goal$$

Value

$$values : \mathbb{P}\, BehavioralConstraint$$

$$RealityVision == \mathbb{P}\, View$$
$$Consistency ::= yes \mid no$$
$$BehavioralConstraint == \mathbb{P}\, AGBelief$$

Adicionalmente, neste modelo também se impõe a restrição que visões, propósitos e valores formem um conjunto consistente. São definidas, então, as funções: $isVisionPurposeConsistent$, $isValueVisionConsistent$ e $isValuePurposeConsistent$. Estas funções recebem respectivamente, pares de visões e propósitos, valores e visões, e valores e propósitos, e os mapeiam para um valor de $Consistency$. Supondo que $vision_1$, $purpose_1$, $value_1$ sejam, respectivamente, instâncias dos tipos $Vision$, $Purpose$ e $Value$, então, $isVisionPurposeConsistent\,(vision_1, purpose_1) = yes$ se o conjunto de fórmulas correspondentes a visões e propósitos forem logicamente consistentes. De modo similar, se nenhuma fórmula em $vision_1$ contradiz alguma fórmula de $value_1$, então vale $isValueVisionConsistent\,(value_1, vision_1) = yes$. Finalmente, se nenhuma fórmula em $purpose_1$ contradiz alguma fórmula de $value_1$, então vale $isValuePurposeConsistent\,(value_1, purpose_1) = yes$.

$$
\begin{array}{l}
isVisionPurposeConsistent : (\mathbb{P}\ Vision \times \mathbb{P}\ Purpose) \rightarrow Consistency \\
isValueVisionConsistent : (\mathbb{P}\ Value \times \mathbb{P}\ Vision) \rightarrow Consistency \\
isValuePurposeConsistent : (\mathbb{P}\ Value \times \mathbb{P}\ Purpose) \rightarrow Consistency
\end{array}
$$

É introduzida no modelo, então, a definição de diretrizes[8] por meio do esquema $GuidingIdea$. Essas diretrizes formam um conjunto consistente de visões, propósitos e valores.

GuidingIdea

$$
\begin{array}{l}
vision : \mathbb{P}\ Vision \\
purpose : \mathbb{P}\ Purpose \\
value : \mathbb{P}\ Value \\
\hline
isVisionPurposeConsistent(vision, purpose) = yes \land \\
isValueVisionConsistent(value, vision) = yes \land \\
isValuePurposeConsistent(value, purpose) = yes
\end{array}
$$

No esquema $PersonalVision$ é declarada a função $personalvision$, definindo que a visão pessoal do agente corresponde a um conjunto de metas que são influenciadas por suas motivações, metas associadas aos papéis desempenhados pelo agente, modelos mentais e por suas diretrizes mentais.

[8] O termo *guiding ideas* é traduzido em [Sen02] como "idéias governantes". Nesta tese, adotou-se o termo "diretrizes".

PersonalVision

$personalvision : \mathbb{P}\ Motivation \pfun \mathbb{P}\ Goal \pfun$
$\quad \mathbb{P}\ IntraPersonalMentalModel \pfun \mathbb{P}\ GuidingIdea \pfun \mathbb{P}\ Goal$

Em resumo, a disciplina de domínio pessoal envolve um conjunto de ações e princípios que visam o desenvolvimento de visões pessoais (*clarifypersonalvisions*), diretrizes (*developguidingideas*), e uma imagem clara da realidade (*enhancerealityvisions*). Adicionalmente, a tensão criativa (*creativetension*) deve produzir metas (*ResolutionGoal*) objetivando a redução dessa tensão.

$ResolutionGoal == \mathbb{P}\ Goal$

PersonalMastery

$developguidingideasactions : \mathbb{P}\ Action$
$enhancerealityvisionsactions : \mathbb{P}\ Action$
$clarifypersonalvisionsactions : \mathbb{P}\ Action$
$personalMasteryPrinciples : \mathbb{P}\ Attribute$
$developguidingideas : \mathbb{P}\ Motivation \times \mathbb{P}\ Action \pfun GuidingIdea$
$enhancerealityvisions : \mathbb{P}\ Motivation \times \mathbb{P}\ Action \pfun \mathbb{P}\ RealityVision$
$clarifypersonalvisions : \mathbb{P}\ Motivation \times \mathbb{P}\ Action \pfun \mathbb{P}\ PersonalVision$
$creativetension : \mathbb{P}\ View \times \mathbb{P}\ PersonalVision \pfun \mathbb{P}\ ResolutionGoal$
$personalvisions : \mathbb{P}\ PersonalVision$
$guidingidea : GuidingIdea$

$developguidingideasactions \cup enhancerealityvisionsactions \cup$
$\quad clarifypersonalvisionsactions = PMActions$

5.3.5 Aprendizagem em Equipe

O objetivo desta disciplina é o desenvolvimento de capacidades de aprendizagem em uma equipe, de forma que esta desenvolva habilidades para a ação coordenada, produzindo resultados extraordinários, e um desenvolvimento mais rápido de seus membros do que seria possível em outras circunstâncias.

Esta disciplina envolve o desenvolvimento de técnicas de diálogo e discussão. A técnica de

diálogo se refere à capacidade dos membros de uma equipe de suprimirem seus pressupostos individuais e entrarem em um estado autêntico de pensamento compartilhado. Por outro lado, a discussão em equipe envolve a apresentação e argumentação envolvendo pontos de vista particulares referentes a um dado contexto, com o objetivo de tomada de decisão ou definição de uma meta [Sen90].

Neste modelo formal, as técnicas de diálogo, discussão, indagação e defesa correspondem a protocolos. Por sua vez, protocolos são definidos por meio da declaração do tipo básico *Protocol*, uma vez que a especificação detalhada de cada tipo de protocolo está fora do escopo da pesquisa apresentada nesta tese. Adicionalmente, no esquema *TeamLearning* é definido que existem ações associadas a cada protocolo, as quais um agente deve ser capaz de executar, e que o desenvolvimento de ações envolvendo diálogos e indagações também dependem das motivações do agente.

$[Protocol]$

TeamLearning

$$
\begin{array}{l}
dialogdiscussactions, inquiryadvocacyactions : \mathbb{P}\, Action \\
teamLearningPrinciples : \mathbb{P}\, Attribute \\
dialogprotocols, discussionprotocols : \mathbb{P}\, Protocol \\
inquiryprotocols, advocacyprotocols : \mathbb{P}\, Protocol \\
actionsofdialogprotocols : Protocol \nrightarrow \mathbb{P}\, Action \\
actionsofdiscussionprotocols : Protocol \nrightarrow \mathbb{P}\, Action \\
actionsofinquiryprotocols : Protocol \nrightarrow \mathbb{P}\, Action \\
actionsofadvocacyprotocols : Protocol \nrightarrow \mathbb{P}\, Action \\
developdialogdiscussactions : \mathbb{P}\, Motivation \times \mathbb{P}\, Action \nrightarrow \mathbb{P}\, Action \\
developinquiryadvocacyactions : \mathbb{P}\, Motivation \times \mathbb{P}\, Action \nrightarrow \mathbb{P}\, Action \\
\hline
dialogdiscussactions \cup inquiryadvocacyactions = LTActions
\end{array}
$$

5.3.6 Pensamento Sistêmico

O pensamento sistêmico revela uma variedade de ações potenciais, que produzem os resultados desejados, mas que também, produzem conseqüências não planejadas. Esta disciplina envolve o reconhecimento de quatro níveis que operam simultaneamente: eventos, padrões de comportamento, estruturas sistêmicas e modelos mentais [SKR+94].

Para a formalização dessa disciplina há a necessidade inicial de se definir alguns tipos básicos relativos a um momento no tempo, um local físico, associações, eventos e modelos computacionais.

$[Time, Space, Links, Event, CompModel]$

Neste trabalho, o tipo *Context* refere-se à informação relacionada a tempo, local, e um conjunto, possivelmente vazio, de entidades envolvidas em alguma modificação ocorrida no ambiente.

$$Context == Time \times Space \times \mathbb{P}\, Entity$$

Os tipos *BehavioralPattern* e *Scenario* são ambos definidos como seqüências de eventos (*Event*) e representam, respectivamente, padrões de comportamentos e cenários. Adicionalmente, é declarada também uma biblioteca de estruturas sistêmicas genéricas (*GenericStructure*), que inclui: laços causais e arquétipos. Os arquétipos e laços aqui definidos, correspondem aos vários tipos apresentados por Senge.

$$BehavioralPattern == \mathrm{seq}_1\, Event$$
$$Scenario == \mathrm{seq}_1\, Event$$
$$Loops ::= ReinforcingLoops \mid BalancingLoops \mid DelayedReinforcingLoops \mid DelayedBalancingLoops$$
$$SystemsArchetypes ::= Fixes\_that\_backfire \mid Limits\_to\_growth \mid Shifting\_the\_burden \mid Tragedy\_of\_the\_commons \mid Accidental\_adversaries$$
$$GenericStructure ::= links\langle\langle Links\rangle\rangle \mid loops\langle\langle Loops\rangle\rangle \mid systemsarchetypes\langle\langle SystemsArchetypes\rangle\rangle$$

A disciplina de pensamento sistêmico também envolve simulações baseadas em modelos computacionais. Considera-se neste modelo, que exista uma biblioteca de modelos computacionais (*CompModelLib*) que os agentes possam consultar e atualizar. Nessa biblioteca são armazenados registros que associam um modelo computacional específico, padrões de comportamento e eventos. Desta forma, com base em padrões de comportamento, um agente pode tentar selecionar um modelo computacional que seja mais conveniente para realizar simulações. Como resultado dessas simulações, devem ser obtidos alguns estados potenciais do ambiente (*PotentialState*). Adicionalmente, também é declarada neste modelo uma biblioteca de planos que pode ser consultada e atualizada pelos agentes. Esses planos estão associados à realização de uma série de estados potenciais.

$$PotentialState == \mathbb{P}_1\ Attribute$$
$$PlanLib == \mathbb{P}\ Plan$$
$$CompModelLib == \mathbb{P}(\mathbb{P}\ CompModel \times \mathbb{P}\ BehavioralPattern \times \mathbb{P}\ Event)$$

O esquema *SystemsThinking* inclui uma biblioteca de estruturas genéricas e um conjunto de máquinas de inferência, que permitem ao agente raciocinar sobre estas estruturas. Nesse esquema, *planlib* corresponde a uma biblioteca de planos que está disponível para o agente. Adicionalmente, é incluída no esquema uma biblioteca de modelos computacionais, que também pode ser consultada pelo agente, um conjunto de eventos, e uma série de funções. A função *contextassessment* mapeia as percepções do agente para contextos. Tais contextos definem para um dado evento: um momento no tempo, um local no espaço, e as entidades associadas à percepção envolvendo esse contexto espaço-temporal. A função *eventanalyser* mapeia contextos para eventos. A função *behavioralpatternanalyser* mapeia um conjunto de eventos para um padrão de comportamento. A função *compmodelsdesign* representa a capacidade do agente de projetar um modelo computacional que corresponde à sua interpretação do relacionamento envolvendo um conjunto de padrões de comportamento e um conjunto de eventos. Além disso, o agente usa as bibliotecas de estruturas genéricas e de modelos computacionais para ajudá-lo nesse processo. O modelo computacional

produzido é então usado em simulações, as quais são efetuadas em diferentes cenários. O produto dessas simulações é um conjunto de estados potenciais futuros, tal como especificado na função de simulação (*simulation*). Finalmente, considerando as metas do agente, o estado potencial que tem maior chance de reduzir a tensão criativa do agente é selecionado, e um conjunto de planos objetivando atingir tal estado é produzido. O processo de produção de tais planos usa, adicionalmente, como uma base de referência, as informações contidas na biblioteca de planos. Os predicados que definem as restrições associadas ao processo descrito acima estão incluídos no esquema $\Delta LearningOrgAgentState$, de modo que possam ser cirscunscritos ao contexto de um agente específico.

SystemsThinking

$events : \mathbb{P}\ Event$
$behavioralpatterns : \mathbb{P}\ BehavioralPattern$
$compmodels : \mathbb{P}\ CompModel$
$simulatedpotentialstates : \mathbb{P}\ PotentialState$
$selectedpotentialstates : \mathbb{P}\ PotentialState$
$scenarios : \mathbb{P}\ Scenario$
$planlib : PlanLib$
$compmodellib : CompModelLib$
$genericstructures : \mathbb{P}\ GenericStructure$
$contextassessment : View \pfun \mathbb{P}\ Context$
$eventanalyser : \mathbb{P}\ Context \pfun \mathbb{P}\ Event$
$behavioralpatternanalyser : \mathbb{P}\ Event \pfun \mathbb{P}\ BehavioralPattern$
$compmodelsdesign : \mathbb{P}\ BehavioralPattern \times \mathbb{P}\ Event \times$
$\quad \mathbb{P}\ GenericStructure \times CompModelLib \pfun \mathbb{P}\ CompModel$
$behavioralpatternfromcompmodel : \mathbb{P}\ CompModel \pfun$
$\quad \mathbb{P}\ BehavioralPattern$
$simulation : \mathbb{P}\ CompModel \times \mathbb{P}\ Scenario \pfun \mathbb{P}\ PotentialState$
$potentialstatesanalyser : \mathbb{P}\ PersonalVision \times$
$\quad \mathbb{P}\ PotentialState \pfun \mathbb{P}\ PotentialState$
$planfrompotentialstate : \mathbb{P}\ PotentialState \times \mathbb{P}\ Plan \pfun \mathbb{P}\ Plan$
$modelscenariofrompotstate : \mathbb{P}\ PotentialState \pfun$
$\quad \mathbb{P}\ CompModel \times \mathbb{P}\ Scenario \times \mathbb{P}\ PotentialState$
$resultingplans : \mathbb{P}\ Plan$

O esquema $\Delta SystemsThinking$ apresenta uma operação sobre $SystemsThinking$ e mostra que a introdução de novas percepções ($systemsactualpercepts?$) resulta na produção de um novo conjunto de planos ($resultingplans!$). Além disso, a biblioteca de modelos computacionais é atualizada de maneira correspondente. Assume-se aqui, que podem haver diferentes cenários a serem considerados.

$\Delta SystemsThinking$

$\Delta LearningOrgAgentState$
$SystemsThinking$
$SystemsThinking'$
$resultingplans! : \mathbb{P}\ Plan$
$systemsactualpercepts? : View$
$modelscenariostate : \mathbb{P}\ CompModel \times$
$\quad \mathbb{P}\ Scenario \times \mathbb{P}\ PotentialState$
$compmodeltoadd : \mathbb{P}\ CompModel$
$behavioralpatterntoadd : \mathbb{P}\ BehavioralPattern$

$genericstructures' = genericstructures$
$planlib' = planlib$
$events' = eventanalyser(contextassessment$
$\quad (systemsactualpercepts?))$
$behavioralpatterns' = behavioralpatternanalyser(events')$
$compmodels' = compmodelsdesign(behavioralpatterns',$
$\quad events', genericstructures, compmodellib)$
$simulatedpotentialstates' = simulation(compmodels',$
$\quad scenarios')$
$selectedpotentialstates' = potentialstatesanalyser$
$\quad (personalmastery.personalvisions,$
$\quad simulatedpotentialstates')$
$modelscenariostate = modelscenariofrompotstate$
$\quad selectedpotentialstates'$
$compmodeltoadd = trifirstmodelscenariostate$
$behavioralpatterntoadd = behavioralpatternfromcompmodel$
$\quad compmodeltoadd$
$compmodellib' = compmodellib \cup \{(compmodeltoadd,$
$\quad behavioralpatterntoadd, events')\}$
$resultingplans! = planfrompotentialstate$
$\quad (selectedpotentialstates', planlib)$

As disciplinas apresentadas até o momento estão associadas principalmente a estados e representações internas do agente, e portanto, são aglutinadas, neste modelo formal, no esquema *IntraPersonalDisciplines*. Dessa forma, um pouco adiante, a definição de que um determinado agente desenvolve as disciplinas de Senge, tomará como base o agrupamento aqui apresentado e um outro agrupamento de disciplinas que envolvem interações: a visão compartilhada e a aprendizagem em equipe.

IntraPersonalDisciplines

$$PersonalMastery$$
$$SystemsThinking$$
$$IntraPersonalMentalModel$$

5.3.7 Desenvolvimento da Equipe Aprendiz

Com o objetivo de apresentar um modelo para uma equipe aprendiz, inicialmente nesta seção, é especificado o processo de desenvolvimento de uma equipe em uma equipe desse tipo.

O processo se baseia no tipo *TLearningTeam*, que modela uma equipe onde planos e metas compartilhados são desenvolvidos por agentes do tipo *LearningOrgAgent*.

TLearningTeam

$$Team$$

$$\text{dom } developcommonplans = \{learningmembers\}$$

O esquema *TLearningTeamIni* modela o estado inicial de *TLearningTeam*, com conjuntos vazios para planos compartilhados e diretrizes do grupo.

TLearningTeamIni

$\Delta TLearningTeam$

$developcommonplans' = \varnothing$
$teamguidingideas'.visions = \varnothing$
$teamguidingideas'.purposes = \varnothing$
$teamguidingideas'.values = \varnothing$
$learningmembers' = \varnothing$
$commonplans' = \varnothing$

Na verdade, neste modelo, assume-se que o processo de desenvolvimento da equipe envolve o desenvolvimento individual de cada agente membro em um agente do tipo *LearningOrgAgent*. Esse pressuposto encontra justificativa no fato de que apenas os agentes desse tipo desenvolvem as ações e conhecem os princípios associados à disciplina de aprendizagem em equipe. Portanto, quando todos os membros da equipe forem do tipo *LearningOrgAgent*, a equipe terá se desenvolvido totalmente, conforme especificado no esquema *TLearningTeamDeveloped*.

TLearningTeamDeveloped

$TLearningTeam$

$membersgroup = learningmembers$

No esquema *TLearningTeamMembersChange* é especificado que um conjunto de agentes de tipo *PlanningAgent* (*pas*?), onde tal conjunto é um subconjunto dos membros da equipe definida por *TLearningTeam*, desenvolve novas capacidades e aprende os princípios associados às disciplinas, causando a instanciação de um novo *LearningOrgAgent* para cada agente do tipo *PlanningAgent* correspondente. Cada novo *LearningOrgAgent* se torna um membro do conjunto *learningmembers* do *TLearningTeam*.

TLearningTeamMembersChange

$\Delta\, TLearningTeam$
$pas? : \mathbb{P}\, PlanningAgent$
$actionsAndPrinciples? : \mathbb{P}\, DisciplinesActionsAndPrinciples$

$resources' = resources$
$commongoals' = commongoals$
$developcommonplans' = developcommonplans$
$teamguidingideas' = teamguidingideas$
$commonplans' = commonplans$
$actionsAndPrinciples? \neq \varnothing$
$\#actionsAndPrinciples? = \#pas?$
$pas? \neq \varnothing$
$pas? \subseteq membersgroup$
$\forall\, pa : LearningOrgAgent \mid pa \in pas? \bullet$
$(\exists_1\, actsAndPrinciples : DisciplinesActionsAndPrinciples \mid$
$actsAndPrinciples \in actionsAndPrinciples? \bullet$
$learningmembers' = learningmembers \cup$
$\{entityLearnDisciplinesActionsAndPrinciples(pa,$
$actsAndPrinciples)\})$
$membersgroup' = membersgroup$

Na declaração apresentada a seguir, a relação *RelationTLearningTeam* associa dois estados de *TLearningTeam*, um anterior e outro posterior à operação *TLearningTeamMembersChange*.

$RelationTLearningTeam == \mathbb{P}\{\, TLearningTeamMembersChange \bullet$
$(\theta\, TLearningTeam, \theta\, TLearningTeam')\}$

Com base na relação definida acima é possível definir um histórico de mudanças de estado de uma equipe. Esse histórico mostra que, a cada estado, um certo número de agentes de tipo *LearningOrgAgent* compõe o conjunto dos membros da equipe. No início do histórico, o estado de *TLearningTeam* corresponde exatamente ao estado de *TLearningTeamIni*. Esta restrição é imposta pelo primeiro predicado de *histTLearningTeamDevelopment*, que especifica que o primeiro elemento da seqüência do histórico deve ser aquele representado pelo estado inicial de um *TLearningTeam*. Ao final do histórico, o estado de *TLearningTeam* corresponde exatamente ao

estado de *TLearningTeamDeveloped*. O terceiro predicado em *histTLearningTeamDevelopment* define que, nesse histórico, quaisquer dois estados contíguos estão relacionados por meio da relação *RelationTLearningTeam*. É assumido aqui, como uma simplificação, que o tamanho do histórico é igual ao número de membros da equipe, de forma que o final do histórico ocorre quando todos os agentes forem do tipo *LearningOrgAgent*.

opTLearningTeam

$$
\begin{array}{l}
TLearningTeam \\
nextChangeLTeamState : RelationTLearningTeam \\
\hline
TLearningTeam \in \{\mathsf{dom}\ nextChangeLTeamState\}
\end{array}
$$

histTLearningTeamDevelopment

$$
\begin{array}{l}
hist : \mathrm{seq}\ opTLearningTeam \\
\hline
\exists\, teamIni : TLearningTeam \bullet (\exists\, opLTeam : \\
\quad opTLearningTeam \mid opLTeam = head(\{1\} \upharpoonleft hist) \bullet \\
\quad (teamIni = (\lambda\, opTLearningTeam \bullet \\
\quad \theta TLearningTeam)\ opLTeam \wedge \\
\quad teamIni \in \{TLearningTeamIni\})) \\
\exists\, teamFin : TLearningTeam \bullet (\exists\, opLTeam : opTLearningTeam \mid \\
\quad opLTeam = head(\{\#hist\} \upharpoonleft hist) \bullet \\
\quad (teamFin = (\lambda\, opTLearningTeam \bullet \\
\quad \theta TLearningTeam)\ opLTeam \wedge \\
\quad teamFin \in \{TLearningTeamDeveloped\})) \\
\forall\, i : \mathbb{N}_1 \mid i \in \mathsf{dom}\ hist \setminus \{1\} \bullet \\
\quad (\exists\, lteamStateBefore, lteamStateAfter : TLearningTeam; \\
\quad lTeamRelation : RelationTLearningTeam \bullet \\
\quad (\exists\, opLTeamBefore, opLTeamAfter : opTLearningTeam \mid \\
\quad opLTeamBefore = head(\{i-1\} \upharpoonleft hist) \wedge \\
\quad opLTeamAfter = head(\{i\} \upharpoonleft hist) \wedge \\
\quad lTeamRelation = opLTeamBefore.nextChangeLTeamState \bullet \\
\quad (lteamStateBefore = (\lambda\, opTLearningTeam \bullet \\
\quad \theta TLearningTeam)\ opLTeamBefore \wedge \\
\quad lteamStateAfter = (\lambda\, opTLearningTeam \bullet \\
\qquad \theta TLearningTeam)\ opLTeamAfter \wedge \\
\quad lteamStateBefore\ \underline{lTeamRelation}\ lteamStateAfter)))
\end{array}
$$

Este processo progride: a cada novo estado de *TLearningTeam* há mais agentes membros do tipo *LearningOrgAgent*. Isto não é verdade apenas na situação em que todos os membros são do tipo *LearningOrgAgent*.

histTLearningTeamDevelopmentProgress

$$
\begin{array}{l}
histTLearningTeamDevelopment \\
lTeamRelation : RelationTLearningTeam \\
\hline
\forall i, j : \mathbb{N}_1 \mid i \in \mathrm{dom}\ hist \wedge j \in \mathrm{dom}\ hist \wedge j = i + 1 \bullet \\
\quad (\exists\, lteamStateBefore, lteamStateAfter : TLearningTeam \mid \\
\quad lteamStateBefore\ \underline{lTeamRelation}\ lteamStateAfter \bullet \\
\quad (\#(lteamStateBefore.learningmembers) < \\
\quad \#(lteamStateAfter.learningmembers) \wedge \\
\quad \#hist \leq \#lteamStateAfter.membersgroup))
\end{array}
$$

Considera-se neste modelo que a equipe aprendiz deve desenvolver suas diretrizes. Um agente do tipo *LearningOrgAgent* adapta suas diretrizes àquelas que emergem do processo de diálogo e discussão. Aqui se requer que estas diretrizes sejam consistentes com os valores, propósitos e visões mantidos pelo agente antes do início do processo de adaptação. As diretrizes resultantes tornam-se também diretrizes do agente.

$$
\begin{array}{l}
agentAdaptGuidingIdeas : (LearningOrgAgent \times \\
\quad GuidingIdea) \rightarrow GuidingIdea \\
\hline
\forall la : LearningOrgAgent;\ gi : GuidingIdea \bullet \\
\quad (\exists_1\, lagBefore, lagAfter : GuidingIdea \mid lagBefore = la.ownguidingideas \bullet \\
\quad (agentAdaptGuidingIdeas(la, gi) = lagAfter \Leftrightarrow \\
\quad (isVisionPurposeConsistent(lagBefore.visions, gi.purposes) = yes \\
\qquad \wedge\ isValueVisionConsistent(lagBefore.values, gi.visions) = yes \\
\qquad \wedge\ isValuePurposeConsistent(lagBefore.values, gi.purposes) = yes \\
\qquad \wedge\ isVisionPurposeConsistent(lagBefore.visions, lagAfter.purposes) = yes \wedge \\
\qquad isValueVisionConsistent(lagBefore.values, lagAfter.visions) = yes \wedge \\
\qquad isValuePurposeConsistent(lagBefore.values, lagAfter.purposes) = yes \wedge \\
\qquad isVisionPurposeConsistent(lagAfter.visions, gi.purposes) = yes \wedge \\
\qquad isValueVisionConsistent(lagAfter.values, gi.visions) = yes \wedge \\
\qquad isValuePurposeConsistent(lagAfter.values, gi.purposes) = yes \wedge \\
\qquad la.ownguidingideas = lagAfter)))
\end{array}
$$

Neste ponto, considera-se que foram produzidas as diretrizes de uma equipe se todos os seus membros tiverem adaptado suas próprias diretrizes durante o processo, e se ao final desse processo, as diretrizes compartilhadas tiverem se tornado um subconjunto das diretrizes adaptadas de cada agente. Assim, adicionalmente, as diretrizes compartilhadas pela equipe são consistentes com as diretrizes de cada agente membro.

$$
\begin{array}{|l}
dialogdiscuss : \mathbb{P}\, LearningOrgAgent \rightarrow GuidingIdea \\
\hline
\forall\, las : \mathbb{P}\, LearningOrgAgent;\ gi : GuidingIdea \bullet \\
\quad (\forall\, la : LearningOrgAgent \mid la \in las \bullet \\
\quad (\exists_1\, lag : GuidingIdea \bullet (dialogdiscuss\,(las) = gi \Leftrightarrow \\
\quad (agentAdaptGuidingIdeas(la, gi) = lag \wedge \\
\qquad la.ownguidingideas = lag \wedge gi.visions \subseteq lag.visions \wedge \\
\qquad gi.purposes \subseteq lag.purposes \wedge gi.values \subseteq lag.values))))
\end{array}
$$

Finalmente, na equipe aprendiz, especificada pelo esquema *LearningTeam*, todos os membros são do tipo *LearningOrgAgent*. As diretrizes da equipe (*teamguidingideas*) resultam de um processo de diálogo e discussão (*dialogdiscuss*). Define-se também, que há recursos e metas compartilhadas entre os membros (*learningmembers*) dessa equipe.

LearningTeam

$$
\begin{array}{|l}
TLearningTeamDeveloped \\
\hline
teamguidingideas = dialogdiscuss\,(learningmembers) \\
\exists\, r : Resource \mid r \in resources \bullet (\forall\, a : LearningOrgAgent \mid \\
\quad a \in learningmembers \bullet (\exists\, p : Plan \mid \\
\quad p \in a.plans \bullet r \in \{a.resourcesofplan\, p\})) \\
\forall\, g : Goal \mid g \in commongoals \bullet (\forall\, a : LearningOrgAgent \mid \\
\quad a \in learningmembers \bullet g \in a.goals)
\end{array}
$$

5.3.8 Visão Compartilhada

Como mencionado na seção anterior, diretrizes de equipe resultam de processos *dialogdiscuss*. Portanto, de modo similar, diretrizes compartilhadas entre um conjunto de equipes resultam de

processos *dialogdiscuss* que envolvem todas essas equipes. Tomando-se como base o pressuposto de que a intersecção entre quaisquer duas equipes de *learningteams* não é vazia, ou seja, que não há equipes isoladas em *learningteams*, então devem existir agentes que são membros de mais de uma equipe. Dessa forma, os processos *dialogdiscuss* devem permitir que sejam desenvolvidas diretrizes compartilhadas que reflitam a influência de todas as equipes.

Cabe lembrar também, que as diretrizes são compostas por visões, valores e propósitos. Desta forma, no esquema *SharedVision* as visões compartilhadas (*sharedvisions*) são obtidas a partir da função *visionsFromGuidingIdea*. Essa função, por sua vez, extrai as visões compartilhadas das diretrizes advindas de *interTeamsGuidingIdeas*.

SharedVision

$$
\begin{array}{l}
learningteams : \mathbb{P}_1\ LearningTeam \\
interTeamsGuidingIdeas : \mathbb{P}\ LearningTeam \rightarrow GuidingIdea \\
sharedvisions : \mathbb{P}\ Vision \\
visionsFromGuidingIdea : \mathbb{P}\ GuidingIdea \rightarrow \mathbb{P}\ Vision \\
\hline
sharedvisions = visionsFromGuidingIdea(\mathsf{ran}\ interTeamsGuidingIdeas) \\
\forall\ gt : \mathbb{P}\ learningteams;\ gi : GuidingIdea \bullet (\forall\ t_1, t_2 : LearningTeam \mid \\
\quad t_1 \in gt \land t_2 \in gt \bullet (\forall\ la_1, la_2 : LearningOrgAgent \mid \\
\quad la_1 \in t_1.learningmembers \land la_2 \in t_2.learningmembers \land \\
\quad (la_1 \in t_1.learningmembers \cap t_2.learningmembers \lor \\
\qquad la_2 \in t_1.learningmembers \cap t_2.learningmembers) \bullet \\
\quad (interTeamsGuidingIdeas\ gt = gi \Leftrightarrow dialogdiscuss\{la_1, la_2\} = \\
\quad t_1.teamguidingideas = t_2.teamguidingideas)))
\end{array}
$$

Nesse ponto é conveniente também agrupar as disciplinas de aprendizagem em equipe e visão compartilhada. Assim, o esquema *InterPersonalDisciplines* reúne as disciplinas que se baseiam em interações entre agentes do tipo *LearningOrgAgent*.

InterPersonalDisciplines

$$
\begin{array}{l}
InterPersonalMentalModel \\
TeamLearning \\
sharedvision : SharedVision
\end{array}
$$

5.3.9 O Agente e as Disciplinas

Como uma conclusão do que foi modelado até agora, define-se que um *LearningOrgAgent* desenvolve as disciplinas de Senge. Com base em suas motivações, nos princípios e nas ações associadas a cada disciplina, o agente desenvolve novas capacidades para agir. Além disso, as diretrizes desse agente são consistentes com todas as suas metas.

LearningOrgAgentAndDisciplines

$$
\begin{array}{l}
LearningOrgAgent \\
improveIntraPersonal : \mathbb{P}\, Motivation \times \mathbb{P}\, Action \times \\
\quad IntraPersonalDisciplines \rightarrow \mathbb{P}\, Action \\
improveInterPersonal : \mathbb{P}\, Motivation \times \mathbb{P}\, Action \times \\
\quad InterPersonalDisciplines \rightarrow \mathbb{P}\, Action \\
personalmastery : PersonalMastery \\
mentalmodels : MentalModel \\
systemsthinkingskill : SystemsThinking \\
\hline
\exists_1\, gi : GuidingIdea \mid gi = ownguidingideas \bullet \\
\quad (isVisionGoalConsistent(gi.visions, allgoals) = yes \land \\
\quad isPurposeGoalConsistent(gi.purposes, allgoals) = yes \land \\
\quad isValueGoalConsistent(gi.values, allgoals) = yes) \\
(mapset\ trifirst)(\mathsf{dom}\ improveIntraPersonal) = \{motivations\} \\
(mapset\ trisecond)(\mathsf{dom}\ improveIntraPersonal) = \{capableof\} \\
(mapset\ trifirst)(\mathsf{dom}\ improveInterPersonal) = \{motivations\} \\
(mapset\ trisecond)(\mathsf{dom}\ improveInterPersonal) = \{capableof\} \\
\mathsf{ran}\ improveIntraPersonal \neq \varnothing \land \mathsf{ran}\ improveInterPersonal \neq \varnothing
\end{array}
$$

5.3.10 Percepção, Ação e Estado do Agente

A percepção do *LearningOrgAgent* depende de suas motivações, modelos mentais, diretrizes e metas. As metas e diretrizes do agente são influenciadas pelo desenvolvimento da disciplina de domínio pessoal.

LearningOrgAgentPerception

$$
\begin{array}{l}
LearningOrgAgentAndDisciplines \\
OrgAgentPerception \\
learnorgperceives : \mathbb{P}\, Motivation \rightarrow MentalModel \rightarrow \\
\quad GuidingIdea \rightarrow \mathbb{P}\, Role \rightarrow \mathbb{P}\, Goal \rightarrow Environment \rightarrow View \\
\hline
\mathrm{dom}\ learnorgperceives = \{motivations\} \\
\mathrm{dom}(learnorgperceives\ motivations) = \{mentalmodels\} \\
\mathrm{dom}(learnorgperceives\ motivations\ mentalmodels) = \{personalmastery.guidingidea\} \\
\mathrm{dom}(learnorgperceives\ motivations\ mentalmodels\ personalmastery.guidingidea) = \{roles\}
\end{array}
$$

A ação desse agente é influenciada por suas motivações, modelos mentais, diretrizes, metas e planos. Além disso, como no esquema *LearningOrgAgentPerception*, as metas e diretrizes do agente são influenciadas pelo desenvolvimento da disciplina de domínio pessoal, uma vez que as metas consideradas são restritas ao conjunto *ResolutionGoal* obtido da função *creativetension*, definida no esquema *PersonalMastery*; e seus planos são influenciados pelo desenvolvimento de capacidades de pensamento sistêmico. As restrições associadas às metas e planos estão especificadas mais adiante, no esquema *LearningOrgAgentState*.

LearningOrgAgentAction

$$
\begin{array}{l}
LearningOrgAgentAndDisciplines \\
OrgAgentAction \\
learnorgact : \mathbb{P}\, Motivation \rightarrow MentalModel \rightarrow GuidingIdea \rightarrow \\
\quad \mathbb{P}\, ResolutionGoal \rightarrow \mathbb{P}\, Plan \rightarrow View \rightarrow Environment \rightarrow Actions \\
\hline
\mathrm{dom}\ learnorgact = \{motivations\} \\
\mathrm{dom}(learnorgact\ motivations) = \{mentalmodels\} \\
\mathrm{dom}(learnorgact\ motivations\ mentalmodels) = \{personalmastery.guidingidea\}
\end{array}
$$

O estado deste agente depende de suas percepções, ações e inclui o estado do agente *OrgAgent*, conforme definido no esquema *LOAgStateIncludes*. Adicionalmente, algumas restrições associadas à percepção e ação do agente são declaradas no esquema *LOAgStatePerceptionActionConstraints*. Desta forma, as metas resultantes da tensão criativa dependem das visões pessoais do agente e influenciam as funções *learnorgperceives* e *learnorgact*. Outrossim, na função *learnorgact* os planos

são influenciados pelo desenvolvimento de capacidades de pensamento sistêmico pelo agente.

LOAgStateIncludes

$$LearningOrgAgentPerception$$
$$LearningOrgAgentAction$$
$$OrgAgentState$$
$$SystemsThinking$$

LOAgStatePerceptionActionConstraints

$$LOAgStateIncludes$$

$$\mathsf{dom}(learnorgperceives\ motivations\ mentalmodels\ personalmastery.guidingidea\ roles) = personalmastery.creativetension(\{actualpercepts\}, personalmastery.personalvisions)$$
$$\mathsf{dom}(learnorgact\ motivations\ mentalmodels\ personalmastery.guidingidea) = \{personalmastery.creativetension(\{actualpercepts\}, personalmastery.personalvisions)\}$$
$$\mathsf{dom}((learnorgact\ motivations\ mentalmodels\ personalmastery.guidingidea)\ (personalmastery.creativetension(\{actualpercepts\}, personalmastery.personalvisions))) = \{systemsthinkingskill.resultingplans\}$$

De modo similar, há várias restrições no esquema *LOAgStateSystemsThinkingConstraints* que detalham a especificação das funções definidas no esquema *SystemsThinking*. Por exemplo, a função *contextassessment* é aplicada sobre *actualpercepts*, de modo que a construção de contextos depende da percepção do agente em relação ao ambiente; os eventos produzidos por *eventanalyser* são obtidos desses contextos; padrões de comportamento (*behavioralpatterns*) resultam da análise dos eventos. A construção e reutilização de modelos computacionais depende não só desses padrões e eventos, mas também de estruturas genéricas e modelos computacionais previamente disponíveis. Simulações efetuadas sobre um conjunto de cenários, usando os modelos computacionais selecionados, produzem estados potencias simulados (*simulatedpotentialstates*). Então, tendo como base a visão do agente, um conjunto de estados é selecionado, e a variável *resultingplans* corresponde ao conjunto de planos escolhido para ser realizado, de modo que o agente atinja os estados potencias selecionados (*selectedpotentialstates*).

LOAgStateSystemsThinkingConstraints

LOAgStateIncludes

$allplans \subseteq planlib$
$\mathsf{dom}\ contextassessment = \{actualpercepts\}$
$\mathsf{dom}\ eventanalyser = \mathsf{ran}\ contextassessment$
$events = eventanalyser\ (contextassessment\ (actualpercepts))$
$\mathsf{dom}\ behavioralpatternanalyser = \mathsf{ran}\ eventanalyser$
$(mapset\ tetrafirst)(\mathsf{dom}\ compmodelsdesign) = \mathsf{ran}\ behavioralpatternanalyser$
$behavioralpatterns = behavioralpatternanalyser\ (events)$
$compmodels = compmodelsdesign(behavioralpatterns, events, genericstructures, compmodellib)$
$simulatedpotentialstates = simulation(compmodels, scenarios)$
$selectedpotentialstates = potentialstatesanalyser(personalmastery.personalvisions,$
$\quad simulatedpotentialstates)$
$(mapset\ tetrasecond)(\mathsf{dom}\ compmodelsdesign) = \mathsf{ran}\ eventanalyser$
$(mapset\ first)(\mathsf{dom}\ simulation) \subseteq \mathsf{ran}\ compmodelsdesign$
$(mapset\ second)(\mathsf{dom}\ potentialstatesanalyser) \subseteq \mathsf{ran}\ simulation$
$(mapset\ second)(\mathsf{dom}\ planfrompotentialstate) \subseteq \{planlib\}$
$resultingplans = planfrompotentialstate(selectedpotentialstates, planlib)$

Finalmente, o esquema *LearningOrgAgentState* inclui as percepções e ações do agente e também todas as restrições apresentadas acima. Além disso, a função *willdo* depende dos planos (*resultingplans*) produzidos pelo pensamento sistêmico do agente (*systemsthinkingskill*). As percepções realmente efetuadas pelo agente são influenciadas pelas motivações, diretrizes, modelos mentais, metas e papéis desempenhados pelo agente.

LearningOrgAgentState

LOAgStateIncludes

LOAgStateSystemsThinkingConstraints
LOAgStatePerceptionActionConstraints
$actualpercepts = learnorgperceives\ motivations\ mentalmodels$
$\quad personalmastery.guidingidea\ roles\ allgoals\ posspercepts$
$willdo = learnorgact\ motivations\ mentalmodels$
$\quad personalmastery.guidingidea(\bigcup(\mathsf{ran}\ personalmastery.creativetension))$
$\quad systemsthinkingskill.resultingplans\ actualpercepts\ env$

5.3.11 Operações

Considerando operações deste tipo de agente, temos os esquemas $\Delta LearningOrgAgentState$ e $LearningOrgAgentInteracts$.

$\Delta LearningOrgAgentState$

$LearningOrgAgentState$
$LearningOrgAgentState'$

$\Delta OrgAgentState$
$learnorgperceives' = learnorgperceives$
$learnorgact' = learnorgact$

No esquema $LearningOrgAgentInteracts$ é especificado que a interação do agente com o ambiente (incluindo interações com outros agentes) é influenciada por suas percepções. Tais percepções servem como informação de entrada para o pensamento sistêmico e são influenciadas pelos modelos mentais e diretrizes desenvolvidas pelo agente. Finalmente, as ações que serão executadas pelo agente, dependem de suas novas motivações, modelos mentais, diretrizes, metas e planos, observando que, nesse caso, tais planos são produzidos pelo pensamento sistêmico do agente.

$LearningOrgAgentInteracts$

$\Delta LearningOrgAgentState$
$DeltaSystemsThinking$
$OrgAgentInteracts$

$actualpercepts' = learnorgperceives\ motivations'$
$\quad mentalmodels'personalmastery'.guidingidea\ roles'$
$\quad allgoals'\ posspercepts'$
$systemsactualpercepts? = actualpercepts'$
$resultingplans' = resultingplans!$
$willdo' = learnorgact\ motivations'\ mentalmodels'$
$\quad personalmastery'.guidingidea\ \{allgoals'\}$
$\quad resultingplans'\ actualpercepts'\ env'$

5.3.12 A Organização Aprendiz

Finalmente, a Organização Aprendiz pode então ser especificada como um tipo de organização formal, na qual todas as equipes são do tipo *LearningTeam*.

Em resumo, o esquema *LearningOrg* especifica, via inclusão de esquemas, que uma Organização Aprendiz é tal que possui estrutura, papéis, e normas que permitem que todas as equipes desenvolvam a disciplina de aprendizagem em equipe. Adicionalmente, todo agente na organização desenvolve a disciplina de domínio pessoal, reflete sobre seus modelos mentais e tenta aprender os modelos dos demais agentes, desenvolve capacidades associadas ao pensamento sistêmico e o conjunto dos agentes desenvolve a visão compartilhada.

Nesse esquema, a variável *sharedvision.sharedvisions* se refere à visão compartilhada da organização. Além disso, a variável *sharedvision.learningteams* é restrita de modo que as equipes que produzem essas visões correspondem exatamente às equipes que compõem a organização.

LearningOrg

$$
\begin{array}{l}
FormalOrg \\
learningteams : \mathbb{P}\, LearningTeam \\
sharedvision : SharedVision \\
\hline
sharedvision.learningteams = learningteams \\
sharedvision.sharedvisions \neq \varnothing \\
\forall\, lt : LearningTeam \mid lt \in learningteams \bullet \\
\quad (\exists_1\, te : Team \mid te \in teams \bullet (\#(te.membersgroup) = 0 \land \\
\quad te.resources = lt.resources \land \\
\quad te.commongoals = lt.commongoals \land \\
\quad te.teamguidingideas = lt.teamguidingideas \land \\
\quad te.commonplans = lt.commonplans \land \\
\quad te = (\lambda\, LearningTeam \bullet \theta Team)\, lt)) \\
\forall\, lt : LearningTeam \mid lt \in learningteams \bullet \\
\quad (\forall\, la : LearningOrgAgent \mid la \in lt.learningmembers \bullet \\
\quad (\exists\, r : Role \mid r \in roles \bullet perform(la, r, regset) = pos)) \\
\#teams = \#learningteams
\end{array}
$$

5.3.13 Análise e Adoção de Metas

Nesta seção, considera-se situações envolvendo avaliações e gerações de metas pelos agentes. Além disso, é especificada a ação de adoção e remoção de metas por um dado agente.

O esquema *AssessGoals* é apresentado em [dL01, p. 44] para representar o processo de avaliação de metas por um agente autônomo. Nesse esquema é introduzida a variável *goallibrary* representando um repositório de metas conhecidas pelo agente. As funções *motiveEffectGenerate* e *motiveEffectDestroy* devolvem um valor inteiro que representa, respectivamente, o efeito sobre as motivações do agente ao adotar um novo conjunto de metas, ou ao remover um conjunto previamente existente de metas. Ambas as funções são influenciadas pelas motivações, metas e percepções atuais do agente.

AssessGoals

$$
\begin{array}{l}
AutonomousAgentState \\
goallibrary : \mathbb{P}_1\ Goal \\
motiveEffectGenerate : \mathbb{P}\ Motivation \rightarrow \mathbb{P}\ Goal \rightarrow View \rightarrow \mathbb{P}\ Goal \rightarrow \mathbb{Z} \\
motiveEffectDestroy : \mathbb{P}\ Motivation \rightarrow \mathbb{P}\ Goal \rightarrow View \rightarrow \mathbb{P}\ Goal \rightarrow \mathbb{Z} \\
satisfyGenerate, satisfyDestroy : \mathbb{P}\ Goal \rightarrow \mathbb{Z} \\
\hline
goals \subseteq goallibrary \\
\forall\ gs : \mathbb{P}\ goallibrary \bullet (satisfyGenerate\ gs = \\
\quad motiveEffectGenerate\ motivations\ goals\ actualpercepts\ gs \wedge \\
\quad satisfyDestroy\ gs = motiveEffectDestroy\ motivations\ goals\ actualpercepts\ gs)
\end{array}
$$

Com base no esquema *AssessGoals*, é especificado como um *LearningOrgAgent* avalia suas motivações para gerar metas. No caso desse agente também é relevante o papel das suas diretrizes (*GuidingIdea*). Desta forma, esse tipo de agente tem que considerar também a relevância das novas metas em relação à sua visão pessoal, seu propósito e sua consistência em relação a seus valores.

LearningOrgAgentAssessGoals

$$
\begin{array}{l}
LearningOrgAgentState \\
AssessGoals \\
lorgAgMotiveEffectGenerate : \mathbb{P}\, Motivation \rightarrow GuidingIdea \rightarrow \mathbb{P}\, Role \rightarrow \mathbb{P}\, Goal \rightarrow \\
\quad View \rightarrow \mathbb{P}\, Goal \rightarrow \mathbb{Z} \\
lorgAgMotiveEffectDestroy : \mathbb{P}\, Motivation \rightarrow GuidingIdea \rightarrow \mathbb{P}\, Role \rightarrow \mathbb{P}\, Goal \rightarrow \\
\quad View \rightarrow \mathbb{P}\, Goal \rightarrow \mathbb{Z} \\
\hline
goals \subseteq goallibrary \\
\forall\, gs : \mathbb{P}\, goallibrary \bullet (satisfyGenerate\ gs = \\
\quad lorgAgMotiveEffectGenerate\ motivations \\
\quad personalmastery.guidingidea\ roles\ goals\ actualpercepts\ gs \land \\
\quad satisfyDestroy\ gs = lorgAgMotiveEffectGenerate \\
\qquad motivations\ personalmastery.guidingidea\ roles\ goals \\
\qquad actualpercepts\ gs)
\end{array}
$$

Adicionalmente, são definidas, também em [dL01, p. 44], as operações de geração e destruição de metas por um agente autônomo. Assim, no esquema *GenerateGoals* o agente cria um conjunto de metas da biblioteca (*goallibrary*) se esse conjunto apresentar o maior efeito motivacional, avaliado pela função *satisfyGenerate*. Similarmente, no esquema *DestroyGoals* o agente destrói um subconjunto de suas metas se a destruição desse subconjunto apresentar o maior efeito motivacional, avaliado pela função *satisfyDestroy*.

GenerateGoals

$$
\begin{array}{l}
\Delta AutonomousAgentState \\
AssessGoals \\
\hline
\exists\, gs : \mathbb{P}\, Goal \mid gs \subseteq goallibrary \bullet (\forall\, os : \mathbb{P}\, Goal \mid os \in (\mathbb{P}\, goallibrary) \bullet \\
\quad ((satisfyGenerate\ gs \geq satisfyGenerate\ os) \land goals' = goals \cup gs))
\end{array}
$$

DestroyGoals

$$
\begin{array}{l}
\Delta AutonomousAgentState \\
AssessGoals \\
\hline
\exists\, gs : \mathbb{P}\, Goal \mid gs \subseteq goallibrary \bullet (\forall\, os : \mathbb{P}\, Goal \mid os \in (\mathbb{P}\, goallibrary) \bullet \\
\quad ((satisfyDestroy\ gs \geq satisfyDestroy\ os) \land goals' = goals \setminus gs))
\end{array}
$$

No caso de um agente do tipo *LearningOrgAgent*, a geração e destruição de metas é similar ao apresentado acima e correspondem aos esquemas *LearningOrgAgentGenerateGoals* e *LearningOrgAgentDestroyGoals*, respectivamente. Porém, as avaliações quanto à geração e destruição de metas se baseiam nas funções *satisfyGenerate* e *satisfyDestroy* que têm suas definições alteradas de modo a incluir também as diretrizes do agente. Isto é uma conseqüência da inclusão do esquema *LearningOrgAgentAssessGoals*.

LearningOrgAgentGenerateGoals

$$\Delta LearningOrgAgentState$$
$$LearningOrgAgentAssessGoals$$

$$\exists\, gs : \mathbb{P}\, Goal \mid gs \subseteq goallibrary \bullet (\forall\, os : \mathbb{P}\, Goal \mid os \in (\mathbb{P}\, goallibrary) \bullet ((satisfyGenerate\ gs \geq satisfyGenerate\ os) \land goals' = goals \cup gs))$$

LearningOrgAgentDestroyGoals

$$\Delta LearningOrgAgentState$$
$$LearningOrgAgentAssessGoals$$

$$\exists\, gs : \mathbb{P}\, Goal \mid gs \subseteq goallibrary \bullet (\forall\, os : \mathbb{P}\, Goal \mid os \in (\mathbb{P}\, goallibrary) \bullet ((satisfyDestroy\ gs \geq satisfyDestroy\ os) \land goals' = goals \setminus gs))$$

Adicionalmente, os agentes podem adotar ou remover metas. Em [dL01, p.46] são definidas as funções *EntityAdoptGoals* e *EntityRemoveGoals*. A função *EntityAdoptGoals* recebe uma entidade e um conjunto de metas e instancia uma nova entidade, com as mesmas características da entidade original, porém incluindo as novas metas. De modo similar, a função *EntityRemoveGoals* instancia uma nova entidade que não possui um dado subconjunto de metas. Neste trabalho, essas duas funções foram estendidas para incluir as novas variáveis que compõem uma entidade neste modelo.

$$
\begin{array}{|l}
EntityAdoptGoals : (Entity \times \mathbb{P}\, Goal) \rightarrow Entity \\
\hline
\forall\, gs : \mathbb{P}\, Goal;\ old, new : Entity \bullet \\
\quad (EntityAdoptGoals(old, gs) = new \Leftrightarrow \\
\quad (new.goals = old.goals \cup gs \wedge \\
\qquad new.allgoals = old.allgoals \cup gs \wedge \\
\qquad new.capableof = old.capableof \wedge \\
\qquad new.attributes = old.attributes \wedge \\
\qquad new.store = old.store \wedge \\
\qquad new.motivations = old.motivations \wedge \\
\qquad new.ownedresources = old.ownedresources \wedge \\
\qquad new.instsreq = old.instsreq \wedge \\
\qquad new.resourcesofplan = old.resourcesofplan \wedge \\
\qquad new.plans = old.plans \wedge \\
\qquad new.planforgoal = old.planforgoal \wedge \\
\qquad new.orgs = old.orgs \wedge \\
\qquad new.roles = old.roles \wedge \\
\qquad new.allplans = old.allplans))
\end{array}
$$

$$
\begin{array}{|l}
EntityRemoveGoals : (Entity \times \mathbb{P}\, Goal) \rightarrow Entity \\
\hline
\forall\, gs : \mathbb{P}\, Goal;\ old, new : Entity \bullet \\
\quad (EntityRemoveGoals(old, gs) = new \Leftrightarrow \\
\quad (new.goals = old.goals \setminus gs \wedge \\
\qquad new.allgoals = old.allgoals \setminus gs \wedge \\
\qquad new.capableof = old.capableof \wedge \\
\qquad new.attributes = old.attributes \wedge \\
\qquad new.store = old.store \wedge \\
\qquad new.motivations = old.motivations \wedge \\
\qquad new.ownedresources = old.ownedresources \wedge \\
\qquad new.instsreq = old.instsreq \wedge \\
\qquad new.resourcesofplan = old.resourcesofplan \wedge \\
\qquad new.plans = old.plans \wedge \\
\qquad new.planforgoal = old.planforgoal \wedge \\
\qquad new.orgs = old.orgs \wedge \\
\qquad new.roles = old.roles \wedge \\
\qquad new.allplans = old.allplans))
\end{array}
$$

Analogamente, considera-se aqui a adoção de metas pelo agente do tipo *LearningOrgAgent*.

Neste caso, está explicitado que o agente deve considerar suas motivações e diretrizes para avaliar o seu interesse em adotar novas metas ou remover uma determinada meta do seu conjunto de metas.

A função *getLOrgAgFromGenerateGoals* permite extrair as definições referentes ao esquema *LearningOrgAgent* a partir de um tipo *LearningOrgAgentGenerateGoals*.

$$getLOrgAgFromGenerateGoals == (\lambda\ LearningOrgAgentGenerateGoals \bullet LearningOrgAgent)$$

Com a utilização da funçao declarada acima, é possível definir as funções *lOrgAgentAdoptGoals* e *lOrgAgentRemoveGoals*, tendo como base, respectivamente, as funções *EntityAdoptGoals* e *EntityRemoveGoals*, definidas anteriormente.

$$\begin{array}{|l}
lOrgAgentAdoptGoals : (LearningOrgAgent \times \mathbb{P}\ Goal) \rightarrow LearningOrgAgent \\
\hline
\forall\ gs : \mathbb{P}\ Goal;\ oldAg, newAg : LearningOrgAgent \bullet \\
\quad (\exists_1\ lag : GuidingIdea \mid lag = oldAg.ownguidingideas \bullet \\
\quad (\exists_1\ oldAgGGoals : LearningOrgAgentGenerateGoals \mid \\
\quad oldAg \in getLOrgAgFromGenerateGoals\ (oldAgGGoals) \bullet \\
\quad (lOrgAgentAdoptGoals(oldAg, gs) = newAg \Leftrightarrow \\
\quad (isVisionGoalConsistent(lag.visions, gs) = yes \\
\qquad \wedge \\
\qquad isPurposeGoalConsistent(lag.purposes, gs) = yes \\
\qquad \wedge \\
\qquad isValueGoalConsistent(lag.values, gs) = yes \\
\qquad \wedge \\
\qquad gs \subseteq oldAgGGoals.goallibrary \\
\qquad \wedge \\
\qquad (\forall\ os : \mathbb{P}\ Goal \mid os \in (\mathbb{P}\ oldAgGGoals.goallibrary) \bullet \\
\qquad\quad oldAgGGoals.satisfyGenerate\ gs \geq \\
\qquad\quad oldAgGGoals.satisfyGenerate\ os) \wedge \\
\qquad EntityAdoptGoals(oldAg, gs) = newAg))))
\end{array}$$

$$
\begin{array}{|l}
lOrgAgentRemoveGoals : (LearningOrgAgent \times \mathbb{P} \\
\quad Goal) \rightarrow LearningOrgAgent \\
\hline
\forall\, gs : \mathbb{P}\, Goal;\ oldAg, newAg : LearningOrgAgent \\
\quad \bullet\, (\exists_1\, lag : GuidingIdea \mid lag = \\
\quad oldAg.ownguidingideas \bullet (\exists_1\, oldAgGGoals : \\
\quad LearningOrgAgentGenerateGoals \mid oldAg \in \\
\quad getLOrgAgFromGenerateGoals\,(oldAgGGoals) \bullet (\exists_1 \\
\quad newGoalSet : \mathbb{P}\, Goal \mid newGoalSet = oldAg.goals \setminus gs \bullet \\
\quad (lOrgAgentRemoveGoals(oldAg, gs) = newAg \Leftrightarrow \\
\quad (isVisionGoalConsistent(lag.visions, newGoalSet) = yes \wedge \\
\qquad isPurposeGoalConsistent(lag.purposes, newGoalSet) = yes \wedge \\
\qquad isValueGoalConsistent(lag.values, newGoalSet) = yes \wedge \\
\qquad gs \subseteq oldAgGGoals.goallibrary \wedge \\
\qquad (\forall\, os : \mathbb{P}\, Goal \mid os \in (\mathbb{P} \\
\qquad\quad oldAgGGoals.goallibrary) \bullet oldAgGGoals.satisfyGenerate \\
\qquad\quad gs \geq oldAgGGoals.satisfyGenerate\ os) \wedge \\
\qquad EntityRemoveGoals(oldAg, gs) = newAg)))))
\end{array}
$$

Nos esquemas *LearningOrgAgentAdoptGoals* e *LearningOrgAgentRemoveGoals* são definidas a adoção e remoção de metas pelo agente do tipo *LearningOrgAgent*, com base nas funções *lOrgAgentAdoptGoals* e *lOrgAgentRemoveGoals*, respectivamente. Portanto, tanto a adoção quanto a remoção de um conjunto de metas por um *LearningOrgAgent* dependem da satisfação motivacional associada a essa operação e à consistência destas metas com as diretrizes do agente.

Além disso, é utilizado o esquema *LearningOrgAgentGenerateGoals* para definir que um agente adota uma meta como se ele próprio a tivesse gerado [9]. Assim, o *LearningOrgAgent* adota as metas se estas oferecerem o maior retorno para a função *satisfyGenerate*, dentre as metas disponíveis para o agente. São consideradas disponíveis as metas presentes na biblioteca *goallibrary*.

[9] Esta estratégia é similar à adotada em [dL01].

LearningOrgAgentAdoptGoals

$\Delta LearningTeam$
$aa? : LearningOrgAgent$
$gs? : \mathbb{P}\ Goal$

$aa? \in learningmembers$
$learningmembers' = (learningmembers \setminus \{aa?\}) \cup \{lOrgAgentAdoptGoals(aa?, gs?)\}$
$membersgroup' = membersgroup$
$resources' = resources$
$teamguidingideas' = teamguidingideas$
$commonplans' = commonplans$
$developcommonplans' = developcommonplans$
$commongoals' = \textbf{if}(\forall\ a : LearningOrgAgent \mid a \in learningmembers' \bullet$
$\quad gs? \subseteq a.goals)\ \textbf{then}\ commongoals \cup (gs? \setminus (gs? \cap commongoals))$
$\quad \textbf{else}\ commongoals$

LearningOrgAgentRemoveGoals

$\Delta LearningTeam$
$aa? : LearningOrgAgent$
$gs? : \mathbb{P}\ Goal$

$aa? \in learningmembers$
$gs? \subseteq aa?.goals$
$learningmembers' = (learningmembers \setminus \{aa?\}) \cup \{lOrgAgentRemoveGoals(aa?, gs?)\}$
$membersgroup' = membersgroup$
$resources' = resources$
$teamguidingideas' = teamguidingideas$
$commonplans' = commonplans$
$developcommonplans' = developcommonplans$
$commongoals' = \textbf{if}((gs? \cap commongoals) \subset commongoals)$
$\quad \textbf{then}\ commongoals \setminus (gs? \cap commongoals)$
$\quad \textbf{else}\ developCommonGoals$
$\qquad (learningmembers')$

5.3.14 Interações, Conhecimento e Modelos Mentais

Nesta seção é apresentada uma formalização para alguns dos processos envolvendo interações entre agentes do tipo *LearningOrgAgent* e como essas interações afetam seus conhecimentos e modelos mentais. Esta apresentação é subdividida em quatro subseções. Na primeira, são abordados aspectos relativos à comunicação entre os agentes, tais como geração, envio, recepção e interpretação de mensagens. Na segunda subseção, é apresentado um modelo para o conhecimento e modelos mentais do agente. Na subseção seguinte, um modelo de interações entre os agentes é apresentado, o qual especifica interações e conversações. Na última parte, dois tipos específicos de conversações - diálogo e discussão - são formalizados, tomando como base a disciplina de aprendizagem em equipe, onde Senge define esses tipos de conversações. É importante destacar ainda, que nesta seção o termo "agente" será usado para indicar um agente do tipo *LearningOrgAgent*. Além disso, os termos "crença" e "conhecimento" são considerados equivalentes, não sendo apresentada nesse trabalho uma distinção formal entre os mesmos.

Comunicação

Com o objetivo de apresentar formalmente aspectos relativos à comunicação entre agentes, inicialmente, é definido o tipo *Message*. Embora existam vários trabalhos que definem detalhadamente as mensagens e a comunicação entre agentes [FFMM94, fIPA00], é suficiente, para este trabalho, definir uma mensagem como um conjunto de *Attribute*. Desta forma, uma mensagem pode conter informações sobre alguma entidade em particular, ou sobre o ambiente em geral.

$$Message == \mathbb{P}\, Attribute$$

Nesta formalização adota-se o pressuposto que toda comunicação ocorre em um determinado contexto de conversação, que é formalizado pelo tipo *SubjectContext*. Tais contextos podem incluir crenças e metas dos agentes.

$$SubjectContext ::= bels\langle\langle AGBelief\rangle\rangle \mid subjgoals\langle\langle \mathbb{P}\, Goal\rangle\rangle$$

As percepções do agente incluem uma série de informações obtidas do ambiente, inclusive as mensagens dirigidas a esse agente. No esquema *MessageExtraction* define-se que o processo de extração de uma mensagem a partir de perceptos envolve ações específicas (*messageExtractionActions*). Obviamente, o agente deve ser capaz de realizar essas ações. A função *getMessage* recebe um conjunto de ações e um conjunto de perceptos e devolve uma mensagem. Além disso, a extração da mensagem só ocorre se o agente selecionar, usando a função *learnorgact*, as ações necessárias à realização dessa atividade.

MessageExtraction

$$\begin{array}{l}
LearningOrgAgentState \\
messageExtractionActions : \mathbb{P}\, Action \\
getMessage : (\mathbb{P}\, Action \times View) \rightarrow Message \\
\hline
messageExtractionActions \subset internalperactions \\
(mapset\, first)(\mathrm{dom}\, getMessage) = \{messageExtractionActions\} \\
\forall\, v : View;\ acts : \mathbb{P}\, Action;\ m : Message \bullet \\
\quad (getMessage(acts, v) = m \Leftrightarrow \\
\quad (acts = messageExtractionActions \wedge \\
\qquad messageExtractionActions \neq \varnothing \wedge \\
\qquad messageExtractionActions = learnorgact\ motivations \\
\qquad\quad mentalmodels\ personalmastery.guidingidea\ \{allgoals\} \\
\qquad\quad resultingplans\ actualpercepts\ env \wedge \\
\qquad v = learnorgperceives\ motivations\ mentalmodels \\
\qquad\quad personalmastery.guidingidea\ roles\ allgoals\ posspercepts))
\end{array}$$

No esquema *ExtractMessage* é apresentado o processo de extração. Esse processo envolve a recepção de novos perceptos (*newPercepts*?) e a extração de uma determinada mensagem (*extractedMessage*!).

ExtractMessage

$$
\begin{array}{l}
LearningOrgAgentInteracts \\
MessageExtraction \\
newPercepts? : View \\
extractedMessage! : Message \\
\hline
willdo' = messageExtractionActions \\
extractedMessage! = getMessage(messageExtractionActions, newPercepts?)
\end{array}
$$

Uma vez extraída dos perceptos, uma mensagem é interpretada pelo agente. No esquema *MessageInterpretation* define-se que o processo de interpretação de uma mensagem envolve ações específicas (*messageInterpretationActions*) que o agente deve ser capaz de realizar. A função *interpretMessage* recebe um conjunto de ações e uma mensagem e devolve uma interpretação interna (*View*). Além disso, a interpretação da mensagem só ocorre se o agente selecionar, usando a função *learnorgact*, as ações necessárias à realização dessa atividade.

MessageInterpretation

$$
\begin{array}{l}
LearningOrgAgentState \\
messageInterpretationActions : \mathbb{P}\, Action \\
interpretMessage : (\mathbb{P}\, Action \times Message) \rightarrowtail View \\
\hline
messageInterpretationActions \subset capableof \\
(mapset\ first)(\mathrm{dom}\ interpretMessage) = \\
\quad \{messageInterpretationActions\} \\
\forall\, v : View;\ acts : \mathbb{P}\, Action;\ m : Message \bullet \\
\quad (interpretMessage(acts, m) = v \Leftrightarrow \\
\quad (acts = messageInterpretationActions \land \\
\qquad messageInterpretationActions \neq \varnothing \land \\
\qquad messageInterpretationActions = learnorgact\ motivations \\
\qquad\quad mentalmodels\ personalmastery.guidingidea\ \{allgoals\} \\
\qquad\quad resultingplans\ actualpercepts\ env))
\end{array}
$$

Logo, dada uma mensagem extraída (*extractedMessage*?), a operação *InterpretMessage* produz uma mensagem interpretada (*interpretedMessage*!). A mensagem interpretada é, então, armazenada na memória (*store*) do agente.

InterpretMessage

$$
\begin{array}{l}
LearningOrgAgentInteracts \\
MessageInterpretation \\
extractedMessage? : Message \\
interpretedMessage! : Message \\
\hline
willdo' = messageInterpretationActions \\
interpretedMessage! = interpretMessage \\
\quad (messageInterpretationActions, extractedMessage?) \\
store' = store \cup interpretedMessage!
\end{array}
$$

As imagens e conceitos internos, tais como aqueles produzidos pelo processo de interpretação de mensagens, podem ser recuperados pelo agente de sua memória. No esquema *ViewRecalling* define-se que o processo de recuperação de um conceito interno envolve ações específicas (*recallActions*) que o agente deve ser capaz de realizar. A função *recallView* recebe um conjunto de ações e uma imagem interna (*View*) e devolve essa mesma imagem interna.

ViewRecalling

$$
\begin{array}{l}
LearningOrgAgentState \\
recallActions : \mathbb{P}\, Action \\
recallView : (\mathbb{P}\, Action \times View) \rightarrowtail\!\!\!\!\!\!\rightarrow View \\
\hline
recallActions \subset internalperactions \\
(mapset\ first)(\mathrm{dom}\ recallView) = \{recallActions\} \\
\forall\, v1, v2 : View;\ acts : \mathbb{P}\, Action \bullet (recallView \\
\quad (acts, v1) = v2 \Leftrightarrow \\
\quad (acts = recallActions \wedge \\
\qquad recallActions \neq \varnothing \wedge \\
\qquad v1 \subset store \wedge \\
\qquad recallActions = learnorgact\ motivations\ mentalmodels \\
\qquad\quad personalmastery.guidingidea\ \{allgoals\} \\
\qquad\quad resultingplans\ actualpercepts\ env))
\end{array}
$$

O processo de recuperação de um conceito interno, apresentado no esquema *RecallView*, envolve a recepção de uma dada imagem interna(*whatView*?) e a devolução dessa imagem (*recalledView*!), caso ela esteja armazenada na memória (*store*) do agente.

RecallView

$LearningOrgAgentInteracts$
$ViewRecalling$
$whatView? : View$
$recalledView! : View$

$willdo' = recallActions$
$recalledView! = recallView(recallActions, whatView?)$

Alternativamente, o agente pode ser capaz de inferir conceitos que não estejam previamente armazenados em sua memória, conforme descrito no esquema *ViewInfer*. Define-se nesse esquema que o processo de inferência de um conceito envolve ações específicas (*inferActions*), as quais o agente deve ser capaz de realizar. A função *inferView* recebe um conjunto de ações e um conceito (*View*) e devolve esse mesmo conceito, desde que tenha sido possível inferí-lo a partir dos conceitos ou imagens armazenados na memória do agente.

ViewInfer

$LearningOrgAgentState$
$inferActions : \mathbb{P}\, Action$
$inferView : (\mathbb{P}\, Action \times View) \rightarrowtail View$

$inferActions \subset capableof$
$(mapset\ first)(\mathrm{dom}\ inferView) = \{inferActions\}$
$\forall\, v1, v2 : View;\ acts : \mathbb{P}\, Action \bullet$
$\quad (inferView\ (acts, v1) = v2 \Leftrightarrow$
$\quad (acts = inferActions \land$
$\qquad inferActions \neq \varnothing \land$
$\qquad v1 \cap store = \varnothing \land$
$\qquad inferActions = learnorgact\ motivations\ mentalmodels$
$\qquad\quad personalmastery.guidingidea\ \{allgoals\}$
$\qquad\quad resultingplans\ actualpercepts\ env))$

O processo de recuperação de um conceito interno, apresentado no esquema *InferView*, envolve a recepção de um dado conceito (*whatView?*) e a devolução do mesmo(*inferredView!*), caso o agente tenha conseguido inferí-lo. Além disso, a memória (*store*) do agente é atualizada com a inclusão

do novo conceito.

InferView

$$
\begin{array}{l}
LearningOrgAgentInteracts \\
ViewInfer \\
whatView? : View \\
inferredView! : View \\
\hline
willdo' = inferActions \\
inferredView! = inferView(inferActions, whatView?) \\
store' = store \cup inferredView!
\end{array}
$$

O agente é também capaz de produzir mensagens que podem ser transmitidas em algum momento no futuro. No esquema *MessageProduction* define-se que o processo de produção de uma mensagem envolve ações específicas (*messageProductionActions*), as quais o agente deve ser capaz de realizar. Essa criação é influenciada pelas motivações e modelos mentais do agente. A função *produceMessage* recebe um conjunto de ações e devolve uma mensagem.

MessageProduction

$$
\begin{array}{l}
LearningOrgAgentState \\
messageProductionActions : \mathbb{P}\, Action \\
produceMessage : \mathbb{P}\, Action \rightarrowtail\!\!\!\!\!\!\rightarrow Message \\
\hline
messageProductionActions \subset capableof \\
\mathsf{dom}\, produceMessage = \{messageProductionActions\} \\
\forall\, m : Message;\ acts : \mathbb{P}\, Action \bullet \\
\quad (produceMessage\ acts = m \Leftrightarrow \\
\quad (acts = messageProductionActions \land \\
\qquad messageProductionActions \neq \varnothing \land \\
\qquad messageProductionActions = learnorgact\ motivations \\
\qquad\quad mentalmodels\ personalmastery.guidingidea\ \{allgoals\} \\
\qquad\quad resultingplans\ actualpercepts\ env))
\end{array}
$$

A operação de produção de uma mensagem, apresentada no esquema *ProduceMessage*, ocorre no contexto de interações do agente com o ambiente. A partir de um novo estado do ambiente (*newEnv*?), novas percepções serão obtidas pelo agente. O agente pode, então, optar por criar

uma mensagem (*producedMessage*!) que poderá ser transmitida posteriormente.

ProduceMessage

$$
\begin{array}{l}
LearningOrgAgentInteracts \\
MessageProduction \\
newEnv? : \mathbb{P}\, Attribute \\
producedMessage! : Message \\
\hline
env' = newEnv? \\
willdo' = messageProductionActions \\
producedMessage! = produceMessage\ messageProductionActions
\end{array}
$$

O agente transmite mensagens usando ações que são um subconjunto de suas capacidades. Além disso, a função *expose* inclui um índice em seu domínio para que seja possível especificar seqüências de interações.

MessageExposition

$$
\begin{array}{l}
LearningOrgAgentState \\
exposeActions : \mathbb{P}\, Action \\
expose : (\mathbb{P}\, Action \times Message \times \mathbb{N}_1) \rightarrowtail\!\!\!\!\!\!\rightarrow Message \\
\hline
exposeActions \subset learningTeamCapabilities \\
(mapset\ trifirst)(\mathrm{dom}\ expose) = \{exposeActions\} \\
\forall\, mint, mexp : Message;\ acts : \mathbb{P}\, Action;\ ind : \mathbb{N}_1 \bullet \\
\quad (expose(acts, mint, ind) = mexp \Leftrightarrow \\
\quad (acts = exposeActions \land \\
\qquad exposeActions \neq \varnothing \land \\
\qquad exposeActions = learnorgact\ motivations\ mentalmodels \\
\qquad\quad personalmastery.guidingidea\ \{allgoals\} \\
\qquad\quad resultingplans\ actualpercepts\ env))
\end{array}
$$

Uma mensagem produzida usando a operação *ProduceMessage* e armazenada internamente na memória do agente, pode ser transmitida.

ExposeMessage

$LearningOrgAgentInteracts$
$MessageExposition$
$ind? : \mathbb{N}_1$
$producedMessage? : Message$
$exposedMessage! : Message$

$env' = env$
$willdo' = exposeActions$
$exposedMessage! = expose(exposeActions, producedMessage?, ind?)$

Interpretação, Conhecimento e Modelos Mentais

Nesta subseção, são apresentados modelos para conhecimento e modelos mentais de um dado agente. Adota-se aqui o pressuposto que os modelos mentais do agente incluem tanto seus conhecimentos como suas crenças.

Um agente conhece um conceito se este pode ser recuperado de sua memória ou se o agente pode inferí-lo, tendo como base os conceitos armazenados em sua memória. No esquema *LOrgAgentKnowledgeIncludes* define-se que a capacidade de conhecer um conceito envolve ações específicas (*knowingActions*), as quais o agente deve ser capaz de realizar. As funções *knowsAView* e *knowsAViewState* recebem um conjunto de ações e um conceito e devolvem, respectivamente, informações do tipo *Consistency* e *Attribute*.

LOrgAgentKnowledgeIncludes

$ViewInfer$
$ViewRecalling$
$knowingActions : \mathbb{P}\, Action$
$knowsAView : (\mathbb{P}\, Action \times View) \rightarrow Consistency$
$knowsAViewState : (\mathbb{P}\, Action \times View) \rightarrow \mathbb{P}\, Attribute$
$knows, doesNotKnow : Attribute$

$knowingActions \subset capableof$
$(mapset\ first)(\mathrm{dom}\ knowsAView) = \{knowingActions\}$
$(mapset\ first)(\mathrm{dom}\ knowsAViewState) = \{knowingActions\}$
$\mathrm{ran}\ knowsAViewState = \{\{knows\}, \{doesNotKnow\}\}$

A função *knowsAView* devolve *yes* quando o conceito recebido estiver armazenado na memória do agente ou puder ser inferido a partir do conteúdo desta. Caso contrário, esta função devolve *no*. Os esquemas *LOrgAgentKnowledgeKnows* e *LOrgAgentKnowledgeDoesNotKnow* descrevem, respectivamente, o primeiro e o segundo caso.

LOrgAgentKnowledgeKnows

LOrgAgentKnowledgeIncludes

$$
\begin{array}{l}
\forall\, v : View;\ acts : \mathbb{P}\, Action \bullet (\exists\, v1 : \\
\quad View \mid v1 \subset store \bullet (knowsAView(acts, v) = yes \Leftrightarrow \\
\quad (acts = knowingActions \wedge \\
\qquad knowingActions \neq \varnothing \wedge \\
\qquad knowingActions = learnorgact\ motivations\ mentalmodels \\
\qquad\quad personalmastery.guidingidea\{allgoals\} \\
\qquad\quad resultingplans\ actualpercepts\ env \wedge \\
\qquad (recallView(acts, v1) = v \vee \\
\qquad\quad inferView(acts, v1) = v))))
\end{array}
$$

LOrgAgentKnowledgeDoesNotKnow

LOrgAgentKnowledgeIncludes

$$
\begin{array}{l}
\forall\, v : View;\ acts : \mathbb{P}\, Action \bullet (\forall\, v1 : \\
\quad View \mid v1 \subset store \bullet (knowsAView(acts, v) \neq yes \Leftrightarrow \\
\quad (acts = knowingActions \wedge \\
\qquad knowingActions \neq \varnothing \wedge \\
\qquad knowingActions = learnorgact\ motivations\ mentalmodels \\
\qquad\quad personalmastery.guidingidea\{allgoals\} \\
\qquad\quad resultingplans\ actualpercepts\ env \wedge \\
\qquad (recallView(acts, v1) \neq v \wedge \\
\qquad\quad inferView(acts, v1) \neq v))))
\end{array}
$$

Similarmente, no esquema *LOrgAgentKnowledgeState* a função *knowsAViewState* devolve o atributo *knows* quando o conceito recebido estiver armazenado na memória do agente ou puder ser inferido a partir do conteúdo desta. Caso contrário, esta função devolve *doesNotKnow*.

LOrgAgentKnowledgeState

$$
\begin{array}{l}
LOrgAgentKnowledgeIncludes \\
\hline
\forall\, v : View;\ acts : \mathbb{P}\, Action;\ c : Consistency \bullet \\
\quad (knowsAViewState(acts, v) = \{knows\} \Leftrightarrow \\
\quad (knowsAView(acts, v) = yes \land c = yes)) \\
\forall\, v : View;\ acts : \mathbb{P}\, Action;\ c : Consistency \bullet \\
\quad (knowsAViewState(acts, v) = \{doesNotKnow\} \Leftrightarrow \\
\quad (knowsAView(acts, v) \neq yes \land c \neq yes))
\end{array}
$$

Finalmente, o esquema *LearningOrgAgentKnowledge* descreve, de modo resumido, que existem conceitos conhecidos e desconhecidos pelo agente.

$$
\begin{aligned}
LearningOrgAgentKnowledge \mathrel{\widehat=}\ & LOrgAgentKnowledgeKnows \land \\
& LOrgAgentKnowledgeDoesNotKnow \land LOrgAgentKnowledgeState
\end{aligned}
$$

A operação descrita pelo esquema *LearningOrgAgentKnowsView* recebe um dado conceito como entrada (*whatView?*) e devolve uma resposta (*answer!*) que será *yes* se o agente conhecer esse conceito, ou seja, se ele estiver armazenado em sua memória ou se puder inferí-lo. Caso contrário, a resposta será *no*.

LearningOrgAgentKnowsView

$$
\begin{array}{l}
LearningOrgAgentInteracts \\
LearningOrgAgentKnowledge \\
whatView? : View \\
answer! : Consistency \\
\hline
willdo' = knowingActions \\
answer! = knowsAView(knowingActions, whatView?)
\end{array}
$$

O esquema *LOrgAgentInts* descreve um agente que é capaz de realizar interações e tem conhecimentos armazenados em sua memória. Além disso, esse agente extrai mensagens obtidas de suas percepções e as interpreta. O agente pode, também, se comunicar por meio da produção e transmissão de mensagens.

$$LOrgAgentInts \mathrel{\widehat=} LearningOrgAgentKnowsView \land ExtractMessage \land InterpretMessage \land RecallView \land ProduceMessage \land ExposeMessage \land InferView$$

No esquema *LOrgAgentInteracting* é especificado que um agente, participando de interações, pode ter modelos destas interações, e também, pode mudá-los. As interações podem ser modeladas como confiáveis ou não confiáveis.

LOrgAgentIntsModels

$$LOrgAgentInts$$
$$modelOfAgentInteractions : LOrgAgentInteracting \rightarrow LOrgAgentInteractingModel$$
$$changeModelOfAgentInteractions : (LOrgAgentInteracting \times LOrgAgentInteractingModel) \rightarrow LOrgAgentInteractingModel$$

$$\mathsf{ran}\ modelOfAgentInteractions = mentalmodels.modelLOrgAgInts$$
$$\mathsf{ran}\ changeModelOfAgentInteractions = mentalmodels.modelLOrgAgInts$$

$$LOrgAgentInteracting \mathrel{\widehat=} [LOrgAgentIntsModels]$$

Em resumo, o esquema *WhatKnow* mostra alguns exemplos do que pode ser representado em relação ao conhecimento de um dado agente. O agente a_i conhece seus próprios conceitos v_1, sua interpretação destes, e sua interpretação para a mensagem enviada por um agente a_j.

WhatKnow

$$interactionPool : \mathbb{P}\, LOrgAgentInteracting$$

$$\forall\, a_i, a_j : LOrgAgentInteracting \mid a_i \in interactionPool \land a_j \in interactionPool \land a_i \neq a_j \bullet (\exists\, v_1, v_2 : View;\ m_j : Message;\ ind_j : \mathbb{N}_1 \bullet (a_i.knowsAView\ (a_i.knowingActions, v_1) = yes \land a_i.knowsAView(a_i.knowingActions, a_i.interpretMessage\ (a_i.messageInterpretationActions, v_1)) = yes \land a_i.knowsAView(a_i.knowingActions, a_i.interpretMessage\ (a_i.messageInterpretationActions, a_j.expose\ (a_j.exposeActions, m_j, ind_j))) = yes))$$

Interações entre Agentes

O esquema *SendMessage* formaliza o processo de envio de mensagem de um dado agente para um grupo de agentes receptores. Aqui, supõe-se que o emissor realiza uma difusão (*broadcast*) da mensagem, ou seja, esse agente divulga sua mensagem para todos os agentes receptores simultâneamente.

SendMessage

$$
\begin{array}{l}
sender : LOrgAgentInteracting \\
receivers : \mathbb{P}\, LOrgAgentInteracting \\
messageSent : Message \\
\hline
sender \notin receivers \\
receivers \neq \varnothing
\end{array}
$$

Cada agente do grupo de *receivers* recebe em *messagesReceived* sua respectiva mensagem.

ReceiveMessage

$$
\begin{array}{l}
sender : LOrgAgentInteracting \\
receivers : \mathbb{P}\, LOrgAgentInteracting \\
messagesReceived : \mathbb{P}\, Message \\
\hline
sender \notin receivers \\
receivers \neq \varnothing \\
\#messagesReceived = \#receivers
\end{array}
$$

Uma interação completa é descrita pelo esquema *Interaction* e envolve o envio e recebimento de uma mensagem. Cada agente receptor tem sua própria interpretação (*interpretedMessage*) de sua respectiva mensagem recebida. Esta, por sua vez, é um membro de *messagesReceived*. Entretanto, todo agente receptor tem conhecimento de que a mensagem recebida é influenciada pela exposição efetuada pelo agente emissor, e que esta exposição é influenciada pelos modelos mentais do emissor. Portanto, cada agente no grupo sabe que os outros agentes têm suas próprias interpretações de suas respectivas mensagens recebidas.

Interaction

$$
\begin{array}{l}
SendMessage \\
ReceiveMessage \\
agents : \mathbb{P}\, LOrgAgentInteracting \\
interactionIndex : \mathbb{N}_1 \\
\hline
agents = \{sender\} \cup receivers \\
\exists_1\, internalMessage : Message \bullet messageSent = \\
\quad sender.expose(sender.exposeActions, internalMessage, \\
\quad interactionIndex) \\
\forall\, ag : LOrgAgentInteracting \mid ag \in receivers \bullet \\
\quad (\exists\, messRec : Message \mid messRec \in \\
\quad messagesReceived \bullet (messRec = ag.interpretMessage \\
\quad (ag.messageInterpretationActions, messageSent) \wedge \\
\quad ag.knowsAView(ag.knowingActions, messRec) = yes)) \\
\forall\, a\_k, a\_j : LOrgAgentInteracting \mid a\_k \in \\
\quad receivers \wedge \\
\quad a\_j \in receivers \bullet a\_k.knowsAView \\
\quad (a\_k.knowingActions, a\_j.interpretMessage \\
\quad (a\_j.messageInterpretationActions, messageSent)) = yes
\end{array}
$$

Define-se nesse modelo que, no estado inicial de uma interação (*InteractionIni*), há uma mensagem enviada, mas que esta ainda não foi recebida pelos agentes receptores. Além disso, cada estado de uma interação tem um índice (*interactionIndex*) associado. Tal índice permite o acesso a cada um dos estados em uma seqüência de interações. No estado inicial, esse índice é igual a 1, indicando o início de uma seqüência.

InteractionIni

$$
\begin{array}{l}
Interaction \\
\hline
interactionIndex = 1 \\
messagesReceived = \varnothing
\end{array}
$$

Após a recepção de uma mensagem, os mesmos agentes continuam envolvidos em interações e desempenhando os mesmos papéis de emissor e receptores. Além disso, cada receptor tem sua própria cópia da mensagem recebida, que é um membro de *messagesReceived*, e o índice de

interações é incrementado.

RecMsgIncreaseIndex

$\Delta Interaction$

$interactionIndex' = interactionIndex + 1$
$sender' = sender$
$receivers' = receivers$
$agents' = agents$
$\#messagesReceived' = \#receivers'$

Cada agente receptor sabe que a mensagem transmitida esteve sujeita à influência dos modelos mentais do emissor.

RecMsgSenderInternalView

$\Delta Interaction$

$\exists\, senderInternalViewOfMessage : Message \bullet messageSent' =$
$\quad sender.expose(sender.exposeActions, senderInternalViewOfMessage, interactionIndex')$

Cada agente receptor tem conhecimento de sua própria interpretação de sua mensagem recebida.

RecMsgInterpret

$\Delta Interaction$

$\forall\, ag : LOrgAgentInteracting \mid ag \in receivers' \bullet$
$\quad (\exists\, messRec : Message \mid messRec \in messagesReceived' \bullet$
$\quad\quad (messRec = ag.interpretMessage(ag.messageInterpretationActions, messageSent') \wedge$
$\quad\quad ag.knowsAView(ag.knowingActions, messRec) = yes))$

Adicionalmente, cada agente do grupo sabe que os demais possuem suas próprias interpretações de suas mensagens.

RecMsgAllInterpret

$\Delta Interaction$

$$\forall a\_k, a\_j : LOrgAgentInteracting \mid a\_k \in receivers \land a\_j \in receivers \bullet$$
$$a\_k.knowsAView(a\_k.knowingActions, a\_j.interpretMessage$$
$$(a\_j.messageInterpretationActions, messageSent')) = yes$$

Além disso, ao receberem suas mensagens, cada agente receptor pode mudar o modelo que possuía do agente emissor. O modelo pode ser de agente honesto ou de agente desonesto.

RecMsgUpdateModels

$\Delta Interaction$

$$\forall ag : LOrgAgentInteracting \mid ag \in receivers' \bullet$$
$$(\exists senderOldModel, senderNewModel : LOrgAgentInteractingModel \mid$$
$$senderOldModel \in ag.mentalmodels.modelLOrgAgInts \land$$
$$senderOldModel = sender.modelOfAgentInteractions\,(ag) \bullet$$
$$(ag.changeModelOfAgentInteractions(sender, senderOldModel) = senderNewModel$$
$$\land\ ag.mentalmodels.modelLOrgAgInts = \{senderNewModel\} \cup$$
$$ag.mentalmodels.modelLOrgAgInts \setminus \{senderOldModel\}))$$

Finalmente, a recepção de uma mensagem (*ReceivingMessage*) envolve todos os aspectos descritos acima.

$$ReceivingMessage \mathrel{\widehat{=}} RecMsgIncreaseIndex \land RecMsgSenderInternalView \land$$
$$RecMsgInterpret \land RecMsgAllInterpret \land RecMsgUpdateModels$$

Em algumas interações o mesmo emissor pode enviar mais de uma mensagem.

SameSenderNewMessage

$\Delta Interaction$

$interactionIndex' = interactionIndex + 1$
$sender' = sender$
$receivers' = receivers$
$agents' = agents$
$messagesReceived' = \varnothing$
$\exists\, senderInternalViewOfMessage : Message \bullet$
$\quad messageSent' = sender.expose(sender.exposeActions,$
$\quad senderInternalViewOfMessage, interactionIndex')$

Alternativamente, um outro agente pode passar a ser o emissor. Neste caso, é assumida aqui a hipótese simplificadora de que o conjunto de agentes participantes de interações não se altera. A única alteração prevista é que um dos agentes receptores passe a ser emissor e, simultaneamente, o emissor anterior passe a atuar como receptor. A nova mensagem é armazenada em $messageSent'$ e o conjunto $messagesReceived'$ torna-se vazio.

ChangeSender

$\Delta Interaction$
$newSender? : LOrgAgentInteracting$

$interactionIndex' = interactionIndex + 1$
$agents' = agents$
$messagesReceived' = \varnothing$
$newSender? \in receivers$
$receivers' = \{sender\} \cup (receivers \setminus \{$
$\quad newSender?\})$
$sender' = newSender?$
$\exists\, senderInternalViewOfMessage : Message \bullet$
$\quad messageSent' = sender'.expose\ (sender'.exposeActions,$
$\quad senderInternalViewOfMessage, interactionIndex')$

Finalmente, quando a interação termina, não há mais mensagens nem agentes envolvidos em interações.

InteractionEnd

$$Interaction$$

$$interactionIndex > 0$$
$$agents = \varnothing$$

Nesse ponto é definida uma relação que mapeia um estado de *Interaction* para o próximo, baseado em [DS89]. O tipo *InteractRel* corresponde ao conjunto que contém todas as relações envolvendo o tipo *Interaction*.

$$RelationRecMsg == \mathbb{P}\{ReceivingMessage \bullet (\theta Interaction, \theta Interaction')\}$$
$$RelationNewMsg == \mathbb{P}\{SameSenderNewMessage \bullet (\theta Interaction, \theta Interaction')\}$$
$$RelationChgSnd == \mathbb{P}\{ChangeSender \bullet (\theta Interaction, \theta Interaction')\}$$
$$InteractRel == RelationRecMsg \cup RelationNewMsg \cup RelationChgSnd$$

Operações sobre um dado estado *opInteraction* definem uma relação entre estados antes e depois. Por exemplo, seja $rel : InteractRel$ e $s1, s2 : opInteraction$, então $s1 \; \underline{rel} \; s2$ if $s1 \in \mathsf{dom}\, rel$, e após a aplicação de op sobre $s1$, o sistema estará no estado $s2$ [DS89].

opInteraction

$$Interaction$$
$$nextInt : InteractRel$$

$$Interaction \in \{\mathsf{dom}\, nextInt\}$$

A seguir, define-se que um histórico (*History*) é uma seqüência de interações onde o primeiro elemento da seqüência corresponde ao estado *InteractionIni*, e que cada estado nesta seqüência está relacionado com o próximo por intermédio de uma operação *nextInt*.

History

$$
\begin{array}{l}
hist : \text{seq}\ opInteraction \\
\hline
\exists\, interactIni : Interaction \bullet (\exists\, opint : opInteraction \mid \\
\quad opint = head(\{1\} \upharpoonright hist) \bullet \\
\quad (interactIni = (\lambda\, opInteraction \bullet \theta Interaction)\ opint \wedge \\
\quad interactIni \in \{InteractionIni\})) \\
\forall\, i : \mathbb{N}_1 \mid i \in \text{dom}\ hist \setminus \{1\} \bullet \\
\quad (\exists\, interactILess1, interactI : Interaction;\ nextIntILess1 : \\
\quad InteractRel \bullet (\exists\, opintILess1, opintI : opInteraction \mid \\
\quad opintILess1 = head(\{i-1\} \upharpoonright hist) \wedge \\
\quad opintI = head(\{i\} \upharpoonright hist) \wedge \\
\quad nextIntILess1 = opintILess1.nextInt \bullet \\
\quad (interactILess1 = (\lambda\, opInteraction \bullet \theta Interaction)\ opintILess1 \wedge \\
\quad interactI = (\lambda\, opInteraction \bullet \theta Interaction)\ opintI \wedge \\
\quad interactILess1\ \underline{nextIntILess1}\ interactI)))
\end{array}
$$

O objetivo aqui é especificar seqüências nas quais agentes continuamente enviam e recebem mensagens. Para a definição de uma seqüência de estados que progrida, é necessária a especificação de uma seqüência particular de operações, tal como a esquematizada abaixo. Nesse esquema, os números entre parênteses correspondem aos índices das interações e `ini` corresponde ao estado *InteractionIni*, `changesender` corresponde ao estado *ChangeSender*, `receivingmsg` corresponde ao estado *ReceivingMessage*, `samesender` corresponde ao estado *SameSenderNewMessage*, e `end` corresponde ao estado *InteractionEnd*.

```
ini(1) - changesender(1) - Interaction(2) - receivingmsg(2) - Interaction(3)
ou changesender(3) - Interaction(4) - receivingmsg(4) - Interaction(5)
ou samesender(3)   - Interaction(4) - receivingmsg(4) - Interaction(5)
ou end
```

Assim, a primeira operação de uma seqüência precisa ser uma mudança de emissor. De fato, é possível perceber que neste tipo de seqüência, cada operação com um índice par deve ser uma operação de recebimento de mensagem. Analogamente, cada operação de índice ímpar deve ser ou uma mudança de emissor ou uma outra emissão de mensagem por um agente que já era emissor.

Desta forma, especifica-se o esquema *Progress*, que é uma seqüência de interações que segue as regras relativas a índices de interações discutidas acima. Adicionalmente, há a necessidade de se definir o conjunto dos números ímpares.

$$Odd == \{int : \mathbb{Z} \bullet 2 * int + 1\}$$

Progress

$$
\begin{array}{l}
History \\
rrmsg : RelationRecMsg \\
rnmsg : RelationNewMsg \\
rchgs : RelationChgSnd \\
\hline
\forall i : \mathbb{N}_1 \mid i \in \mathrm{dom}\, hist \wedge i = 1 \bullet \\
\quad (\exists interactI : Interaction;\ nextIntI : InteractRel \bullet \\
\quad (\exists opintI : opInteraction \mid opintI = head(\{i\} \upharpoonright hist) \wedge \\
\quad nextIntI = opintI.nextInt \bullet nextIntI = rchgs)) \\
\forall i : \mathbb{N}_1 \mid i \in \mathrm{dom}\, hist \wedge i \notin Odd \bullet \\
\quad (\exists interactI : Interaction;\ nextIntI : InteractRel \bullet \\
\quad (\exists opintI : opInteraction \mid opintI = head(\{i\} \upharpoonright hist) \wedge \\
\quad nextIntI = opintI.nextInt \bullet nextIntI = rrmsg)) \\
\forall i : \mathbb{N}_1 \mid i \in \mathrm{dom}\, hist \wedge i > 2 \wedge i \in Odd \bullet \\
\quad (\exists interactI : Interaction;\ nextIntI : InteractRel \bullet \\
\quad (\exists opintI : opInteraction \mid opintI = head(\{i\} \upharpoonright hist) \wedge \\
\quad nextIntI = opintI.nextInt \bullet ((nextIntI = rchgs) \vee \\
\quad (nextIntI = rnmsg))))
\end{array}
$$

Conversações: Diálogo e Discussão

Neste ponto, com base na definição de seqüências de interações, especifica-se que uma sessão de conversação envolve diversas interações entre agentes de um grupo.

Conversation

$$
\begin{array}{l}
session : Progress \\
\hline
\end{array}
$$

O início de uma conversação corresponde ao início do histórico de interações.

ConversationInit

$$Conversation$$

$$\#session.hist = 1$$

Em uma conversação, quando ocorre a mudança de emissor, o histórico reflete cumulativamente a operação. Além disso, a operação do último estado no histórico atual é definida como "mudança de emissor" (*RelationChgSnd*), e a operação do novo estado é definida como "recebendo mensagem" (*RelationRecMsg*). Para refletir a mudança de papéis na conversação, o novo agente emissor é removido do conjunto de recebedores e, ao mesmo tempo, o antigo emissor é incluído nesse conjunto.

ConversationChangeSender

$$\Delta Conversation$$
$$sender? : LOrgAgentInteracting$$
$$receivers! : \mathbb{P}\, LOrgAgentInteracting$$
$$rchgs : RelationChgSnd$$
$$rrmsg : RelationRecMsg$$

$$sender? \in (lastsession.hist).agents$$
$$\exists\, newState : opInteraction;$$
$$\quad senderInternalViewOfMessage : Message \mid$$
$$\quad newState.interactionIndex =$$
$$\quad (last\ session.hist).interactionIndex + 1 \land$$
$$\quad newState.agents = (lastsession.hist).agents \land$$
$$\quad newState.sender = sender? \land$$
$$\quad newState.sender \neq (lastsession.hist).sender \land$$
$$\quad receivers! = \{(lastsession.hist).sender\} \cup$$
$$\qquad (lastsession.hist).receivers \setminus \{sender?\} \land$$
$$\quad newState.receivers = receivers! \land$$
$$\quad newState.messageSent = sender?.expose$$
$$\qquad (sender?.exposeActions, senderInternalViewOfMessage,$$
$$\qquad newState.interactionIndex) \land$$
$$\quad newState.messagesReceived = \varnothing \land$$
$$\quad newState.nextInt = rrmsg \bullet$$
$$\qquad session'.hist = session.hist \oplus \{(\#session.hist + 1, newState)\}$$
$$(lastsession.hist).nextInt = rchgs$$

Em uma conversação, o mesmo emissor pode enviar várias mensagens. Além disso, a operação do último estado no histórico atual é definida como "nova mensagem" (*RelationNewMsg*), e a operação do novo estado é definida como "recebendo mensagem" (*RelationRecMsg*). Os agentes que desempenham os papéis de receptores e emissor permanecem os mesmos.

ConversationNewMessage

$\Delta Conversation$
$messageSent! : Message$
$rnmsg : RelationNewMsg$
$rrmsg : RelationRecMsg$

$\exists newState : opInteraction;$
$senderInternalViewOfMessage : Message \mid$
$newState.interactionIndex =$
$(last\ session.hist).interactionIndex + 1 \land$
$newState.agents = (lastsession.hist).agents \land$
$newState.sender = (lastsession.hist).sender \land$
$newState.receivers = (lastsession.hist).receivers \land$
$newState.messageSent = newState.sender.expose$
$(newState.sender.exposeActions,$
$senderInternalViewOfMessage, newState.interactionIndex)$
$\land$
$messageSent! = newState.messageSent \land$
$newState.messagesReceived = \varnothing \land$
$newState.nextInt = rrmsg \bullet$
$session'.hist = session.hist \oplus \{(\#session.hist + 1, newState)\}$
$(lastsession.hist).nextInt = rnmsg$

O esquema *ConversationReceivingMessage* apresenta o processo de recebimento de mensagens em uma conversação. Analogamente ao caso apresentado acima, os agentes que desempenham os papéis de receptores e emissor permanecem os mesmos. A operação do novo estado é definida como "nova mensagem" (*RelationNewMsg*).

ConversationReceivingMessage

$$
\begin{array}{l}
\Delta Conversation \\
messageSent? : Message \\
rnmsg : RelationNewMsg \\
rrmsg : RelationRecMsg \\
\hline
\exists newState : opInteraction \mid \\
\quad newState.interactionIndex = \\
\quad (last\ session.hist).interactionIndex + 1 \land \\
\quad newState.agents = (lastsession.hist).agents \land \\
\quad newState.sender = (lastsession.hist).sender \land \\
\quad newState.receivers = (lastsession.hist).receivers \land \\
\quad newState.messageSent = messageSent? \land \\
\quad newState.messagesReceived = \{a : LOrgAgentInteracting \mid \\
\qquad a \in newState.receivers \bullet \\
\qquad a.interpretMessage(a.messageInterpretationActions, \\
\qquad messageSent?)\} \land newState.nextInt = rnmsg \bullet \\
\quad session'.hist = session.hist \oplus \{(\#session.hist + 1, newState)\}
\end{array}
$$

Nesta seção, o interesse é especificar dois tipos de conversação em particular: diálogo e discussão. Em [SKR+94], algumas características destas conversações são apresentadas. Aqui, tais características são modeladas como protocolos. Esses protocolos têm uma série de restrições (*ProtocolConstraint*) a eles associadas.

$$ProtocolConstraint == \mathbb{P}_1\ Attribute$$

Como mencionado acima, são tratados aqui dois tipos de protocolos: diálogo e discussão.

$$ProtocolMode ::= DialogProtocol \mid DiscussProtocol$$

O esquema *Protocol* define que um protocolo tem uma série de restrições definidas e o tipo do protocolo.

Protocol

$constraints : \mathbb{P}\ ProtocolConstraint$
$protocolmode : ProtocolMode$

O esquema *DialogDiscussProtocol* define que diálogo e discussão são protocolos. No caso do protocolo de diálogo, o tipo associado é *DialogProtocol* e no caso de discussão o tipo associado é *DiscussProtocol* .

DialogDiscussProtocol

$dialogProt : Protocol$
$discussProt : Protocol$

$dialogProt.protocolmode = DialogProtocol$
$discussProt.protocolmode = DiscussProtocol$

O tipo construído *SwitchProtocol* indica se em uma conversação há mudança de tipo de protocolo (*Switch_yes*) ou não (*Switch_no*).

$SwitchProtocol ::= Switch\_yes \mid Switch\_no$

Adicionalmente, Senge também afirma que, de forma a gerenciar uma sessão de conversação que obedeça a estes protocolos e a balancear o emprego de diálogos e discussões, um tipo particular de agente é necessário: o *facilitador* (*FacilitatorLOrgAg*). Inicialmente, as variáveis necessárias à definição do *facilitador* são declaradas no esquema *FacLOrgAgIncludes*.

FacLOrgAgIncludes

$LearningOrgAgent$
$enforceCoherencyWithSubjectContext : Conversation \times SubjectContext \rightarrow Actions$
$enforceCoherencyWithProtocol : Conversation \times DialogDiscussProtocol \rightarrow Actions$
$enforceCoherencyOfMessage : Message \times SubjectContext \rightarrow Consistency$
$enforceAdherenceOfMessage : Message \times \mathbb{P}\ ProtocolConstraint \rightarrow Consistency$
$currentSubjectContext : SubjectContext$
$currentProtocolMode : ProtocolMode$
$switchProtocolDecision : SwitchProtocol$

Aqui, o agente *facilitador* é modelado como um *LearningOrgAgent*, com as capacidades adicionais (*FacLOrgAgCapabilities*) que permitam-no garantir que todas as mensagens comunicadas sejam coerentes com o contexto da conversação (*FacLOrgAgEnforceCoherency*), e também que as regras associadas ao tipo de protocolo em uso num dado instante sejam respeitadas (*FacLOrgAgEnforceAdherence*). O pressuposto, nesse caso, é que todos os agentes envolvidos nas sessões conhecem tais protocolos, pois todos eles são do tipo *LearningOrgAgent*. Além disso, arbitrariamente, uma sessão pode ser alterada do modo diálogo para o modo discussão, ou vice-versa. Finalmente, o *facilitador* examina cada mensagem em cada interação.

FacLOrgAgCapabilities

FacLOrgAgIncludes

$$
\begin{array}{l}
\exists\, r : Role \mid r \in roles \bullet r.ident = facilitator \\
\mathsf{ran}\ enforceCoherencyWithSubjectContext \subset \{capableof\} \\
\mathsf{ran}\ enforceCoherencyWithProtocol \subset \{capableof\} \\
(mapset\ first)(\mathsf{dom}\ enforceCoherencyWithSubjectContext) = \\
\quad (mapset\ first)(\mathsf{dom}\ enforceCoherencyWithProtocol) \\
currentProtocolMode = DialogProtocol \lor \\
\quad currentProtocolMode = DiscussProtocol \\
(mapset\ second)(\mathsf{dom}\ enforceCoherencyWithSubjectContext) = \{currentSubjectContext\}
\end{array}
$$

FacLOrgAgEnforceCoherency

FacLOrgAgIncludes

$$
\begin{array}{l}
\forall\, conv : Conversation \mid conv \in (mapset\ first) \\
\quad (\mathsf{dom}\ enforceCoherencyWithSubjectContext) \bullet \\
\qquad (\forall\, subj : SubjectContext \mid subj \in (mapset\ second)(\mathsf{dom} \\
\qquad\quad enforceCoherencyWithSubjectContext) \bullet \\
\qquad\qquad (\forall\, int : Interaction \mid \langle int \rangle\ \mathsf{in}\ conv.session.hist \bullet \\
\qquad\qquad\quad (\exists\, msg : Message;\ ac : \mathbb{P}\, Action \mid msg = int.messageSent \bullet (\\
\qquad\qquad\quad ((enforceCoherencyOfMessage(msg, \\
\qquad\qquad\qquad\qquad subj) = yes) \Rightarrow (ac = \varnothing)) \land \\
\qquad\qquad\quad ((enforceCoherencyOfMessage(msg, \\
\qquad\qquad\qquad\qquad subj) = no)) \Rightarrow (ac \neq \varnothing))))))
\end{array}
$$

FacLOrgAgEnforceAdherence

$$
\begin{array}{l}
FacLOrgAgIncludes \\
\hline
\forall\, conv : Conversation \mid conv \in (mapset\ first)(\mathsf{dom}\ enforceCoherencyWithProtocol) \bullet \\
\quad (\forall\, dgp : DialogDiscussProtocol \mid dgp \in (mapset\ second) \\
\qquad (\mathsf{dom}\ enforceCoherencyWithProtocol) \bullet \\
\qquad\quad (\forall\, int : Interaction \mid \langle int \rangle\ \mathsf{in}\ conv.session.hist \bullet \\
\qquad\qquad (\exists\, msg : Message;\ ac : \mathbb{P}\, Action \mid msg = int.messageSent \bullet (\\
\qquad\qquad\quad ((enforceAdherenceOfMessage(msg, \\
\qquad\qquad\qquad dgp.dialogProt.constraints) = yes \Rightarrow (ac = \varnothing)) \wedge \\
\qquad\qquad\quad ((enforceAdherenceOfMessage(msg, \\
\qquad\qquad\qquad dgp.dialogProt.constraints) = no) \Rightarrow (ac \neq \varnothing)) \wedge \\
\qquad\qquad\quad ((enforceAdherenceOfMessage(msg, \\
\qquad\qquad\qquad dgp.discussProt.constraints) = yes) \Rightarrow (ac = \varnothing)) \wedge \\
\qquad\qquad\quad ((enforceAdherenceOfMessage(msg, \\
\qquad\qquad\qquad dgp.discussProt.constraints) = no) \Rightarrow (ac \neq \varnothing))))))
\end{array}
$$

Com base nos esquemas de apoio declarados acima, é possível agora especificar o agente *facilitador*. É importante notar aqui que, nos casos de violação de coerência ou aderência o *facilitador* deve realizar algumas ações corretivas. O detalhamento destas ações foge do escopo deste trabalho, sendo suficiente declarar a existência das mesmas.

$$
\begin{aligned}
FacilitatorLOrgAg \mathrel{\widehat=} {} & FacLOrgAgCapabilities \wedge FacLOrgAgEnforceCoherency \wedge \\
& FacLOrgAgEnforceAdherence
\end{aligned}
$$

Uma vez especificado o agente facilitador, é possível agora a descrição de diálogo e discussão. Um diálogo é um tipo de conversação, com um assunto específico e com a presença de um agente *facilitador*.

Dialog

$$
\begin{array}{l}
Conversation \\
subjectContext : SubjectContext \\
facilitatorAgent : FacilitatorLOrgAg \\
\hline
facilitatorAgent.currentSubjectContext = subjectContext \\
facilitatorAgent.currentProtocolMode = DialogProtocol
\end{array}
$$

Analogamente, uma discussão é um tipo de conversação, com um assunto específico e com a presença de um agente *facilitador*.

Discussion

$$
\begin{array}{l}
Conversation \\
subjectContext : SubjectContext \\
facilitatorAgent : FacilitatorLOrgAg \\
\hline
facilitatorAgent.currentSubjectContext = subjectContext \\
facilitatorAgent.currentProtocolMode = DiscussProtocol
\end{array}
$$

Uma sessão pode envolver diálogo, discussão ou ambos. Além disso, há uma número máximo de vezes que um mesmo agente pode atuar como emissor (*individualSenderLimit*).

DialogDiscussionSession

$$
\begin{array}{l}
Dialog \\
Discussion \\
currentProtocolMode : ProtocolMode \\
switchProtocol : SwitchProtocol \\
durationLimit : \mathbb{N}_1 \\
individualSenderLimit : \mathbb{N}_1 \\
\hline
currentProtocolMode = facilitatorAgent.currentProtocolMode \\
switchProtocol = Switch\_yes \Rightarrow \\
\quad currentProtocolMode = \textbf{if } currentProtocolMode = DialogProtocol \\
\qquad \textbf{then } DiscussProtocol \\
\qquad \textbf{else } DialogProtocol
\end{array}
$$

A qualquer instante, após o início das interações, o grupo de agentes pode definir a duração da sessão. Assume-se que a duração corresponde a um limite superior sobre o número de interações e que este limite precisa ser definido como um valor superior ao número de participantes.

$$
\begin{array}{l}
groupDefineDurationLimit : (\mathbb{P}\, LOrgAgentInteracting) \\
\quad \nrightarrow \mathbb{N}_1 \\
\hline
\forall\, g : \mathbb{P}\, LOrgAgentInteracting \bullet \\
\quad groupDefineDurationLimit\,(g) \geq \#g
\end{array}
$$

DialogDiscussionSessionDefineDurationLimit

$\Delta DialogDiscussionSession$

$session' = session$
$subjectContext' = subjectContext$
$facilitatorAgent' = facilitatorAgent$
$currentProtocolMode' = currentProtocolMode$
$switchProtocol' = switchProtocol$
$\#session.hist \geq 1$
let $ags == (head\ session.hist).agents \bullet (\#ags \neq 0 \land$
$durationLimit' = groupDefineDurationLimit\ (ags) \land$
$individualSenderLimit' = durationLimit'\ \mathsf{div}\ \#ags)$

Em uma sessão completa, envolvendo diálogo e discussão, o número de interações deve ser maior ou igual ao número de participantes de modo a permitir que cada agente possa desempenhar o papel de emissor. Ao término, todos os agentes concordam com o encerramento da sessão. Assume-se aqui também, como uma hipótese simplificadora, que o grupo permanece o mesmo durante toda a sessão.

DialogDiscussionComplete

$\Delta DialogDiscussionSession$
$\Delta FacilitatorLOrgAg$

$facilitatorAgent' = facilitatorAgent$
$subjectContext' = subjectContext$
$currentProtocolMode' = facilitatorAgent'.currentProtocolMode$
$\#session.hist > 1$
let $ags == (head\ session.hist).agents \bullet \#session.hist \geq \#ags$
let $ags == (head\ session.hist).agents \bullet (\forall\ a1 : LOrgAgentInteracting \mid a1 \in ags \bullet$
$(\exists\ i : \mathbb{N} \mid i \in \mathsf{dom}\ session.hist \bullet$
$(\exists\ int : opInteraction \mid int = head(\{i\} \upharpoonright session.hist) \bullet int.sender = a1)))$

A qualquer instante, após o início das interações, o grupo pode definir que a sessão está encerrada.

$DDSessionUnfolding ::= DDSessionEnd \mid DDSessionAdvance$

$groupDefineSessionEnd : (\mathbb{P}\, LOrgAgentInteracting) \nrightarrow DDSessionUnfolding$

DialogDiscussionSessionTermination

$\Delta DialogDiscussionSession$

$session' = session$
$subjectContext' = subjectContext$
$facilitatorAgent' = facilitatorAgent$
$currentProtocolMode' = currentProtocolMode$
$switchProtocol' = switchProtocol$
$\#session.hist \geq 1$
$\textbf{let}\ ags == (headsession.hist).agents \bullet groupDefineSessionEnd\,(ags) = DDSessionEnd$

Após o processo de mudança de protocolo, o *facilitador* informa os participantes do novo protocolo a ser usado na sessão.

$facilitatorChgProt : FacilitatorLOrgAg \nrightarrow ProtocolMode$

$\forall\, af : FacilitatorLOrgAg \mid af \in (\mathsf{dom}\, facilitatorChgProt) \land$
$af.switchProtocolDecision = Switch\_yes \bullet$
$((af.currentProtocolMode = \textbf{if}\ af.currentProtocolMode = DialogProtocol$
$\textbf{then}\ DiscussProtocol$
$\textbf{else}\ DialogProtocol) \land$
$\mathsf{ran}\, facilitatorChgProt = \{af.currentProtocolMode\})$

Para que ocorra uma mudança no tipo de protocolo na sessão, é necessária a intervenção do agente *facilitador*.

DialogDiscussionProtocolChange

$\Delta DialogDiscussionSession$

$session' = session$
$subjectContext' = subjectContext$
$facilitatorAgent' = facilitatorAgent$
$switchProtocol' = switchProtocol$
$\#session.hist \geq 1$
$currentProtocolMode' = facilitatorChgProt\, facilitatorAgent$

Com o desenvolvimento do diálogo, o *facilitador* tenta garantir que nenhum agente atue no papel de emissor todo o tempo. Para tal, aqui considera-se que o quociente entre a duração e o número de agentes no grupo é um limite superior sobre o número de vezes que um dado agente pode desempenhar o papel de emissor. Se um agente tentar exceder esse limite, o grupo pode decidir pela mudança de emissor, ou até permitir que esse agente envie sua mensagem. Neste caso, o grupo pode também aumentar a duração da sessão. Caso contrário, algum agente pode não conseguir tornar-se emissor antes do término da sessão.

$$groupDefineSender : (\mathbb{P}\, LOrgAgentInteracting) \nrightarrow LOrgAgentInteracting$$

$$\forall g : \mathbb{P}\, LOrgAgentInteracting \bullet (\exists ag : LOrgAgentInteracting \mid ag \in g \bullet groupDefineSender(g) = ag)$$

Finalmente, o desenvolvimento de uma sessão de conversação que envolva diálogo e discussão, inclui mudanças nos agentes desempenhando o papel de emissor, e um dado emissor enviando diversas mensagens.

DialogDiscussionDevelopment

$$\Delta DialogDiscussionSession$$
$$ConversationChangeSender$$
$$ConversationReceivingMessage$$
$$ConversationNewMessage$$

$$facilitatorAgent' = facilitatorAgent$$
$$subjectContext' = subjectContext$$
$$session' = session$$
$$subjectContext' = subjectContext$$
$$currentProtocolMode' = currentProtocolMode$$
$$switchProtocol' = switchProtocol$$
$$\#session.hist \geq 1$$
$$\textbf{let } ags == (lastsession.hist).agents \bullet (\#ags \neq 0 \land sender? = groupDefineSender\,(ags) \land$$
$$(((last\ session.hist).sender \neq sender? \land ConversationChangeSender)$$
$$\lor$$
$$ConversationNewMessage) \land$$
$$durationLimit' = groupDefineDurationLimit\,(ags) \land$$
$$individualSenderLimit' = durationLimit' \text{ div } \#ags)$$

5.4 Propriedades do Modelo Formal

Nesta seção, investiga-se as características do modelo apresentado nas seções anteriores. Estas características, ou propriedades, são uma conseqüência da interpretação de aspectos da teoria de Senge e sua representação formal no modelo especificado. Desta forma, em princípio, estas propriedades poderiam ser verificadas formalmente no modelo. Entretanto, neste trabalho, adotou-se a estratégia de apenas apresentar uma argumentação informal, ao invés de uma demonstração formal, para explicar como estas propriedades encontram-se presentes no arcabouço formal da Quinta Disciplina.

5.4.1 Definições

Inicialmente, são introduzidas diversas definições que permitirão a investigação de características do modelo.

Sejam a_j e a_i agentes que interagem entre si.

$$| \ a_i, a_j : LOrgAgentInteracting$$

Sejam m_i e m_j mensagens que contêm informações referentes a algum contexto (o estado atual do ambiente, por exemplo) ou propostas para mudar tal estado (metas).

$$| \ m_i, m_j : Message$$

Sejam v_i e v_j visões ou conceitos, do tipo *View*. Tais conceitos resultam de interpretações efetuadas pelos agentes a_i e a_j para cada mensagem m_i e m_j, respectivamente.

$$| \ v_i, v_j : View$$

Sejam sm_i e sm_j seqüências de mensagens.

$$| \ sm_i, sm_j : \text{seq } Message$$

Sejam ind_i e ind_j índices de interações.

$| \; ind_i, ind_j : \mathbb{N}_1$

Seja $goal_s$ um conjunto de metas.

$| \; goal_s : \mathbb{P}\, Goal$

(a) Uma mensagem m_i é consistente com os modelos mentais de um agente a_i se não contiver predicados que sejam logicamente inconsistentes com os predicados contidos nos modelos mentais de a_i.

$$
\begin{array}{|l}
consistentMessage : LOrgAgentInteracting \times Message \rightarrow \\
\quad Consistency \\
\hline
\forall\, ag : LOrgAgentInteracting;\ m_i : Message \bullet \\
\quad (\exists\, internalViewOfMessage : Message \bullet \\
\quad (consistentMessage(ag, m_i) = yes \Leftrightarrow \\
\quad (internalViewOfMessage = (ag.produceMessage \\
\qquad ag.messageProductionActions) \wedge \\
\qquad ag.knowsAView(ag.knowingActions, \\
\qquad\quad internalViewOfMessage) = yes \wedge \\
\qquad m_i = ag.expose(ag.exposeActions, \\
\qquad\quad internalViewOfMessage, ind_i) \wedge \\
\qquad ag.knowsAView(ag.knowingActions, m_i) = yes)))
\end{array}
$$

(b) Um agente a_i é *trustagent* se expõem apenas mensagens que sejam consistentes com seus modelos mentais, isto é, a_i expõe apenas informações nas quais ele acredite ou conheça. Caso contrário, a_i é *nontrustagent*.

$$
\begin{array}{|l}
trustAgent : LOrgAgentInteracting \rightarrow Consistency \\
nonTrustAgent : LOrgAgentInteracting \rightarrow Consistency \\
\hline
\forall\, ag : LOrgAgentInteracting;\ m_i : Message \bullet \\
\quad (trustAgent\ ag = yes \Leftrightarrow consistentMessage(ag, m_i) = yes) \\
\forall\, ag : LOrgAgentInteracting \bullet (\exists\, m_i : Message \bullet \\
\quad (nonTrustAgent\ ag = yes \Leftrightarrow \\
\quad (\neg\ (consistentMessage(ag, m_i) = yes))))
\end{array}
$$

(c) Uma interação entre agentes a_i e a_j é $TRUST$ se durante sua interação a_i tiver em seus modelos mentais um modelo *trustagent* de a_j e, reciprocamente, se a_j tiver em seus modelos mentais um modelo *trustagent* de a_i. Inversamente, uma interação é $NOTTRUST$ se pelo menos um dentre a_i e a_j, não tiver em seus modelos mentais um modelo *trustagent* de seu respectivo parceiro.

$$
\begin{array}{|l}
\textit{TrustInteraction} \\
Interaction \\
\hline
\#agents = 2 \\
\#receivers = 1 \\
\exists\, model1, model2 : LOrgAgentInteractingModel; \\
\quad trustModel1, trustModel2 : TrustLOrgAgentInterModel \bullet \\
\quad (\exists_1\ ag : LOrgAgentInteracting \bullet \\
\quad (ag.modelOfAgentInteractions\,(sender) = model1 \wedge \\
\quad sender.modelOfAgentInteractions\,(ag) = model2 \wedge \\
\quad model1 = trustModel1 \wedge \\
\quad model2 = trustModel2 \wedge \\
\quad trustModel2 \in \\
\quad sender.mentalmodels.modelTrustLOrgAgInts \wedge \\
\quad trustModel1 \in ag.mentalmodels.modelTrustLOrgAgInts))
\end{array}
$$

NonTrustInteraction

Interaction

$$
\begin{array}{l}
\#agents = 2 \\
\#receivers = 1 \\
\exists\, model1, model2 : LOrgAgentInteractingModel; \\
\quad trustModel1, trustModel2 : NonTrustLOrgAgentInterModel \\
\quad \bullet\, (\exists_1\, ag : LOrgAgentInteracting \bullet \\
\quad (ag.modelOfAgentInteractions\,(sender) = model1 \land \\
\quad sender.modelOfAgentInteractions\,(ag) = model2 \land \\
\quad ((model1 = trustModel1 \land \\
\qquad trustModel1 \in \\
\qquad ag.mentalmodels.modelNonTrustLOrgAgInts) \lor \\
\qquad (model2 = trustModel2 \land \\
\qquad\quad trustModel2 \in \\
\qquad\quad sender.mentalmodels.modelNonTrustLOrgAgInts))))
\end{array}
$$

(d) Um agente a_i é *believer* se, na ausência de interações prévias com um agente qualquer a_j, a_i assumir implicitamente um modelo *trustagent* de a_j.

BelieverLOrgAgentInter

LOrgAgentInteracting

$$
\begin{array}{l}
\forall\, ag : LOrgAgentInteracting \mid ag \notin \mathsf{dom} \\
\quad modelOfAgentInteractions \bullet (\exists_1\, trustModel : \\
\quad TrustLOrgAgentInterModel \bullet (modelOfAgentInteractions = \\
\quad modelOfAgentInteractions \cup \{(ag \mapsto trustModel)\} \land \\
\quad trustModel \in ag.mentalmodels.modelTrustLOrgAgInts))
\end{array}
$$

(e) Um agente a_i é *skeptical* se, na ausência de interações prévias com um agente qualquer a_j, a_i assumir implicitamente um modelo *nontrustagent* de a_j.

SkepticalLOrgAgentInter

$$
\begin{array}{l}
LOrgAgentInteracting \\
\hline
\forall\, ag : LOrgAgentInteracting \mid ag \notin \mathsf{dom} \\
\quad modelOfAgentInteractions \bullet (\exists_1\, trustModel : \\
\quad NonTrustLOrgAgentInterModel \bullet (modelOfAgentInteractions \\
\quad = modelOfAgentInteractions \cup \{(ag \mapsto trustModel)\} \wedge \\
\quad trustModel \in ag.mentalmodels.modelNonTrustLOrgAgInts))
\end{array}
$$

(f) Um agente a_i revisa seu modelo *trustagent* de a_j para *nontrustagent*, se a_i for capaz de perceber que não há consistência entre os modelos mentais, propostas, mensagens e acordos expostos por a_j e suas ações.

(g) Um agente a_i revisa seu modelo *nontrustagent* de a_j para *trustagent*, se a_j for capaz de justificar para a_i qualquer inconsistência percebida por a_i entre os modelos mentais, propostas, mensagens e acordos expostos por a_j e suas ações.

(h) Uma sessão de diálogo/discussão envolvendo os agentes a_i e a_j produz com sucesso um conjunto de metas $goal_s$ de metas compartilhadas se, ao final da sessão, forem válidos:

(h_1) a_i acredita em $goal_s$ e

(h_2) a_j acredita em $goal_s$ e

(h_3) a_i acredita que a_j acredita na interpretação de a_j de $goal_s$ e

(h_4) a_j acredita que a_i acredita na interpretação de a_i de $goal_s$

5.4.2 Propriedades

O modelo formal apresenta as seguintes propriedades:

1. Interações confiáveis são necessárias.

A análise do modelo formal construído mostra que não é possível a construção de um conjunto de metas compartilhadas (analogamente, para visões, valores e propósitos) se os agentes envolvidos no processo não confiarem uns nos outros.

O contexto de análise envolve os agentes a_i e a_j, e uma seqüência inicial de emissão e recepção de mensagens, efetuada pelos agentes visando estabelecer um conjunto de metas sobre as quais ambos concordem. Aqui, as hipóteses iniciais são:

($h0$) Não existem registros anteriores de interações entre a_i e a_j.

($h1$) Ambos a_i e a_j são agentes do tipo *believer*.

A análise envolve o seguinte processo:

A. O agente a_i envia a mensagem m_i.

Com base no esquema *ReceivingMessage* é possível investigar o que o agente a_j conhece após a recepção de m_i.

A1. Primeiro, a_i transmitiu a mensagem m_i.

$$\exists\, senderInternalViewOfMessage : Message \bullet \\ m_i = a_i.expose(a_i.exposeActions, \\ senderInternalViewOfMessage, ind_i)$$

A2. Então, a mensagem recebida é interpretada por a_j.

$$\exists\, messRec : Message \bullet \\ (messRec = a_j.interpretMessage(\\ a_j.messageInterpretationActions, m_i) \wedge \\ a_j.knowsAView(a_j.knowingActions, messRec)) = yes$$

De acordo com ($h1$), a_j é do tipo *believer*. Portanto, a_j crê que a_i acredita em m_i.

$$a_j.knowsAView(a_j.knowingActions, \\ a_i.knowsAViewState(a_i.knowingActions, m_i)) = yes$$

Assim, a_j acredita que a_i acredite em m_i, ou seja, que a_i acredite que o estado do ambiente corresponda à informação contida em m_i, e a_j pode usar essa crença para fundamentar suas próximas deliberações.

B. Nesta etapa a_j pode enviar a mensagem m_j referente ao contexto desta sessão de diálogo/discussão. Assim, analogamente, a_i recebe m_j e conhece:

$$
\begin{aligned}
&\exists\, messRec : Message \bullet \\
&\quad (messRec = a_i.interpretMessage(\\
&\qquad\qquad a_i.messageInterpretationActions, m_j) \wedge \\
&\quad a_i.knowsAView(a_i.knowingActions, messRec) = yes \wedge \\
&\quad a_i.knowsAView(a_i.knowingActions, \\
&\qquad a_j.knowsAViewState(a_j.knowingActions, m_j)) = yes
\end{aligned}
$$

Novamente, a_i acredita que a_j acredite no que está declarado em m_j. Pode-se assumir que este processo prossiga por várias interações envolvendo atividades de negociação ou argumentação, de modo que, quando a sessão for considerada encerrada por ambos a_i e a_j, cada agente terá um modelo mais rico sobre as crenças de seu parceiro correspondente.

Em relação ao conjunto de metas compartilhadas, há duas possibilidades ao final da sessão:

(a) houve um acordo sobre um conjunto de metas $goal_s$, ou

(b) não houve acordo sobre as metas.

C. Supondo-se que ocorra o caso (a), uma vez que o caso (b) dispensa análises adicionais, então é válido afirmar que ambos os agentes acreditam nesse conjunto de metas.

$$
\begin{aligned}
&a_i.knowsAView(a_i.knowingActions, (\bigcup(goal_s))) = yes \\
&a_j.knowsAView(a_j.knowingActions, (\bigcup(goal_s))) = yes
\end{aligned}
$$

Sejam ind_{if} e ind_{jf}, respectivamente, os índices da interação final entre os agentes a_i e a_j; $msgoal_i$ e $msgoal_j$ mensagens que confirmam a aquiescência de a_i e a_j com $goal_s$; e

$vsgoal_i$ e $vsgoal_j$ interpretações efetuadas por a_i e a_j sobre suas respectivas mensagens. Formalmente:

$$
\begin{array}{|l}
msgoal_i, msgoal_j : Message \\
vsgoal_i, vsgoal_j : View \\
ind_{if}, ind_{jf} : \mathbb{N}_1 \\
\hline
\exists\, senderInternalViewOfMessage_i : Message \bullet \\
\quad msgoal_i = a_i.expose(a_i.exposeActions, \\
\quad senderInternalViewOfMessage_i, ind_{if}) \wedge \\
\quad (vsgoal_j = a_j.interpretMessage \\
\quad (a_j.messageInterpretationActions, msgoal_i) \wedge \\
\quad a_j.knowsAView(a_j.knowingActions, vsgoal_j) = yes \wedge \\
\quad a_j.knowsAView(a_j.knowingActions, \\
\qquad a_i.knowsAViewState(a_i.knowingActions, msgoal_i)) = yes \\
\exists\, senderInternalViewOfMessage_j : Message \bullet \\
\quad msgoal_j = a_j.expose(a_j.exposeActions, \\
\quad senderInternalViewOfMessage_j, ind_{jf}) \wedge \\
\quad (vsgoal_i = a_i.interpretMessage \\
\quad (a_i.messageInterpretationActions, msgoal_j) \wedge \\
\quad a_i.knowsAView(a_i.knowingActions, vsgoal_i) = yes \wedge \\
\quad a_i.knowsAView(a_i.knowingActions, \\
\qquad a_j.knowsAViewState(a_j.knowingActions, msgoal_j)) = yes
\end{array}
$$

Então, o que é conhecido por cada agente em cada estágio corresponde a:

$$
\begin{array}{ll}
(k1) & a_j.knowsAView(a_j.knowingActions, \\
 & a_i.knowsAViewState(a_i.knowingActions, \\
 & \qquad a_i.expose(a_i.exposeActions, senderInternalViewOfMessage_i, \\
 & \qquad\quad ind_{if}))) = yes
\end{array}
$$

$$
\begin{array}{ll}
(k2) & a_i.knowsAView(a_i.knowingActions, \\
 & a_j.knowsAViewState(a_j.knowingActions, \\
 & \qquad a_j.expose(a_j.exposeActions, senderInternalViewOfMessage_j, \\
 & \qquad\quad ind_{jf}))) = yes
\end{array}
$$

Se a_i e a_j, ao término da sessão, ainda modelam o parceiro correspondente como *trustagent*,

então pode-se afirmar que a_i acredita que a_j acredite em sua interpretação $vsgoal_j$ da meta $goal_s$. Se, entretanto, ao final da sessão, pelo menos um dentre a_i e a_j modela o respectivo parceiro como *nontrustagent*, então uma dentre $(k1)$ ou $(k2)$ não é válida e, portanto, não é respeitado o conjunto de condições h_1, h_2, h_3 e h_4. Como conseqüência, $goal_s$ não representa, ao término da sessão, um conjunto de metas compartilhadas entre a_i e a_j.

2. Os agentes precisam ter características de *trustagent*.

   Inicialmente, deve-se observar que *ExposedBelief* é um tipo associado às crenças do agente, ou seja, que estão armazenadas na memória do agente. Assim, de acordo com a função *advocacy*, definida no esquema *IntraPersonalMentalModel*, o agente tem motivações e efetua ações de modo a expor suas crenças e processo de raciocínio. Portanto, o agente é do tipo *trustagent*.

3. As motivações dos agentes precisam ser consistentes com as disciplinas.

   Primeiramente, na formalização de modelos mentais, as funções *reflection* e *advocacy* dependem das motivações do agente. No caso da disciplina de domínio pessoal, as diversas funções: *personalvision*, *developguidingideas*, *enhancerealityvisions*, *clarifypersonalvisions*, e *creativetension*, são todas baseadas nas motivações do agente.

   A formalização da disciplina de aprendizagem em equipe introduz um conjunto de ações e protocolos, entre eles *inquiry* e *advocacy* que, conforme comentado acima no estudo sobre modelos mentais, são influenciados pelas motivações dos agentes. Além disso, há ainda a função *developdialogdiscussactions*, que também se baseia nas motivações do agente.

   No caso da disciplina de pensamento sistêmico é declarada a função *potentialstatesanalyser*, que depende de *PersonalVision*, e que, por sua vez, se baseia nas motivações do agente.

   Finalmente, a disciplina de visão compartilhada é desenvolvida por equipes de agentes que geram acordos através de processos *dialogdiscuss*, que usam ações de *dialogdiscuss*. O desenvolvimento destas ações também depende das motivações dos agentes.

Em resumo, um agente que atue em conformidade com as cinco disciplinas precisa ter motivações que sejam consistentes com as ações e protocolos associados a cada uma das disciplinas.

4. Os agentes precisam ser tenazes.

   Conforme a especificação do esquema *LearningOrgAgent*, este tipo de agente desenvolve as cinco disciplinas independentemente de considerações temporais, ou seja, as disciplinas determinam o modo como o agente pensa, age e interage. Logo, os agentes desenvolvem persistentemente as disciplinas.

5. Os agentes precisam ser *cooperativos*.

   De acordo com [Wei01] uma cooperação é um tipo de coordenação entre agentes não antagônicos na qual os participantes obtêm sucesso ou falham juntos. Em contraste, em uma competição o sucesso de um participante implica na falha dos demais.

   Na formalização para modelos mentais apresentada aqui, requer-se que um agente exponha suas crenças de modo que, em uma equipe de agentes, os demais aprendam os modelos mentais desse agente. O mesmo vale para a construção de visão compartilhada.

   É importante considerar-se aqui que um "índice" de sucesso de uma organização aprendiz está associado a sua capacidade de desenvolver visões compartilhadas. No entanto, esta é uma realização coletiva, na qual os participantes obtêm sucesso ou falham juntos. Logo, pode-se concluir que, de fato, a organização aprendiz é um cenário de cooperação.

6. Mudanças organizacionais emergem das motivações individuais.

   Segundo o modelo formal produzido, todos os agentes da organização desenvolvem as cinco disciplinas e as disciplinas determinam como os agentes pensam, agem e interagem. Também mostrou-se acima que o desenvolvimento das disciplinas por um dado agente é influenciado por suas motivações. Além disso, foi apresentado que metas, valores e visões compartilhadas podem emergir de interações entre agentes e também dependem do desenvolvimento das

disciplinas pelos agentes envolvidos. Logo, tanto realizações coletivas, como a produção de visões compartilhadas, como as ações individuais que causam mudanças na organização emergem de motivações individuais.

7. A rotatividade de agentes precisa ser baixa na organização.

   O processo de interação descrito na seção anterior apresenta uma situação em que não há mudanças nos agentes que são membros da equipe durante todo o processo. Entretanto, levando-se em conta tais mudanças, é possível observar que o processo de interações subjacente à construção de visões e modelos mentais compartilhados é afetado conforme descrito abaixo.

   Considerando-se uma equipe com n membros e a descrição de sessões de interações apresentada previamente, sabe-se que são necessários pelo menos n ciclos de interações para que cada agente tenha a oportunidade de desempenhar o papel de emissor. Supondo-se que na etapa k, com $k < n$, k agentes já tenham atuado como emissores, e que ocorra a incorporação de um novo agente na equipe, o número de componentes será igual a $n + 1$. Logo, serão necessárias pelo menos $n + 1$ etapas para que todos os agentes atuem como emissores. Porém, nesse caso, o novo membro também tem que atuar como receptor para todas as k apresentações que ocorreram antes que ele fosse um membro da equipe. Assim, pelo menos $k + n + 1$ etapas são necessárias para cada novo agente incorporado à equipe em uma dada etapa k.

   Agora, considerando-se o caso em que um agente substitua um antigo membro da equipe, então pelo menos $k + n$ etapas são necessárias para cada novo agente incorporado à equipe em substituição a membros da equipe em uma dada etapa k.

   Assim, quanto mais freqüentes as inclusões ou substituições de membros de uma equipe, menos eficiente será o processo de produção de conhecimento compartilhado via interações.

   De fato, para cada agente na equipe, uma certa quantidade de tempo é gasta na modelagem

dos modelos mentais e visões dos demais. Se a rotatividade for alta nas equipes, todo este esforço poderá ser perdido.

8. O número de membros de equipes aprendizes precisa ser limitado.

   Para que se possa aumentar a eficiência do processo de interações, o número de membros em uma dada equipe precisa ser limitado. Caso contrário, as equipes terão que investir muito tempo envolvidas em sessões de diálogo e discussão. Assim, para poder enfrentar essa complexidade, a organização aprendiz precisa ser subdividida em uma certa quantidade de equipes.

   Em cada equipe T, cada membro tem um modelo mais detalhado e complexo dos demais membros. Esse modelo emerge de ciclos de comunicação/deliberação/ação. Os membros de outras equipes na organização precisam modelar apenas T: suas metas, atividades, responsabilidades e membros, sem precisar detalhar o modelo de cada membro de T.

5.4.3 Observações

Em relação à propriedade (1), é conveniente uma reflexão a respeito da efetiva transformação de $goal_s$ num conjunto compartilhado de metas. Considerando-se que no estado final as crenças dos agentes referem-se a suas respectivas interpretações de $goal_s$, uma possível maneira de se avaliar a adoção compartilhada das metas é verificar a consistência lógica entre os predicados de $vsgoal_i$, $vsgoal_j$ e $goal_s$. Entretanto, apenas a verificação da consistência lógica dessas interpretações não é suficiente para garantir que $goal_s$ se tornou um conjunto compartilhado de metas. Adicionalmente, também é importante verificar se as semânticas dessas interpretações são coerentes. Por exemplo, a_i e a_j podem concordar com a meta: "aumentar a receita em dez por cento". Entretanto, esses agentes podem interpretar em seus modelos mentais o termo receita como receita bruta e receita líquida, respectivamente. Assim, idealmente, todos os termos usados nas sessões de diálogo/discussão deveriam ter a mesma semântica para os agentes envolvidos. Uma possível

solução para esse problema é a utilização de glossários, nos quais os vários termos necessários para o desenvolvimento das atividades das equipes estejam definidos, e também manuais de procedimentos ou registros de melhores práticas. No caso de equipes de agentes artificiais, a solução, equivalente a esta, é o uso de ontologias artificiais [Gru93], um recurso onde significados, conceitos, e seus relacionamentos, são definidos e armazenados. Entretanto, a disponibilidade desse tipo de base de conhecimento não garante que os agentes a tenham consultado e, mesmo que isso ocorra, que tenham interpretado corretamente seu conteúdo. Além disso, existe ainda a possibilidade de um agente adotar, intencionalmente, uma interpretação distinta daquela registrada, em benefício próprio. Um modo mais pragmático de verificar se um conjunto de metas emergiu das sessões é definir ciclos que alternem sessões de diálogo/discussão e ações, ou seja, ciclos de deliberação e monitoração; de modo que seja possível a avaliação dos resultados e desempenho da equipe e corrigir problemas de interpretação.

Quanto à propriedade (2), é interessante considerar se um agente a_j, do tipo *nontrustagent*, poderia fazer parte desta organização aprendiz formalizada e participar de interações do tipo *TRUST* com algum agente a_i pertencente à organização. É claro que, primeiramente há a necessidade de tornar mais flexível a formalização referente à função *advocacy* de modo que o agente a_j seja também capaz de expor noções ou conceitos nos quais ele não creia. Assim, objetivando manter interações do tipo *TRUST* com a_i, a_j deve "construir" um modelo *trustagent* de si próprio para a_i. Portanto, a_j precisa agir em conformidade com cada crença ou intenção por ele comunicada. Caso contrário, a_i pode modelá-lo como *nontrustagent*, impossibilitando o estabelecimento de uma relação do tipo *TRUST* entre esses agentes. Desta forma, respeitados esses requisitos, mesmo que a_j seja "potencialmente" *nontrustagent*, a_i não será capaz de reconhecer tal fato, pelo menos no contexto das interações entre a_i e a_j, e essas interações serão *TRUST*.

Com relação à propriedade (5), em um cenário competitivo um agente expõe seus modelos mentais apenas se acreditar que, ao fazê-lo, irá se beneficiar de alguma forma, em detrimento do desempenho dos demais agentes.

5.5 Considerações Finais

O modelo construído para a teoria da Quinta Disciplina, utiliza a mesma estratégia de estruturação e modularização empregada no arcabouço SMART: com base em tipos mais simples de agentes, novos agentes, mais complexos, são especificados. Desta forma, o modelo pode ser visto como uma estrutura em camadas[10], onde a camada de mais alto nível corresponde ao modelo do agente que é membro da organização aprendiz preconizada pela teoria de Senge.

- Algumas características e limitações desta formalização

  Além das propriedades discutidas na seção anterior, o modelo produzido apresenta algumas características e limitações que são conseqüência direta, ou indireta, das escolhas efetuadas durante o desenvolvimento do projeto, tais como: a teoria organizacional selecionada, o formalismo adotado, e o arcabouço formal escolhido.

  - A escolha da linguagem Z

    A linguagem Z é adequada para representar estados, com boa expressividade, e recursos que facilitam a estruturação da especificação e reutilização dos tipos definidos na formalização. Entretanto este formalismo é limitado para especificar processos concorrentes. Neste caso, o uso de uma linguagem híbrida como CSP-OZ [Fis97] poderia ter produzido melhores resultados. A linguagem CSP-OZ resulta da combinação de CSP [Hoa78] e Object-Z, um dialeto de Z orientado a objetos. Por exemplo, especificações de interações, como sessões de diálogos e discussões, poderiam ser realizadas com maior simplicidade e concisão. Entretanto, tal mudança implicaria em uma re-especificação do próprio SMART em Object-Z, o que estaria fora do escopo do projeto apresentado nesta tese.

  - A escolha do arcabouço SMART

[10] Ver o diagrama de estrutura de esquemas no apêndice B.

O arcabouço SMART foi construído para servir de base para a construção de outros modelos formais e arquiteturas orientadas a agentes. Como conseqüência do seu uso, o modelo construído para a Quinta Disciplina foi influenciado por algumas opções de estruturação adotadas no SMART, tal como a opção por definir esquemas particulares de percepção e ação para cada tipo de agente. Entretanto, esse tipo de estruturação mostrou-se útil para o desenvolvimento do modelo e desta pesquisa. Além disso, a própria definição da autonomia do agente da *LearningOrgAgent* fica também fundamentada na definição adotada no SMART, ou seja, que o conjunto de motivações do agente define seu processo de criação e adoção de metas.

- A escolha da teoria organizacional

  A teoria de Senge apresenta uma visão particular de quais são os requisitos necessários para a construção de uma organização que aprende: o foco está sempre no indivíduo. Assim, características presentes nesses indivíduos (ou desenvolvidas por eles) devem permitir a emergência dessa organização flexível, adaptável e que é capaz de aprender. Entretanto, outros aspectos organizacionais importantes como planejamento, coordenação, controle, divisão e atribuição de tarefas, definições de critérios de competência para alocação de papéis organizacionais, entre outros; não são descritos em sua teoria. Além disso, Senge não define claramente o processo pelo qual as visões de cada indivíduo podem dar origem a uma visão compartilhada por todos os membros da organização.

  Como conseqüência, o modelo reflete, em maior ou menor grau, tais características: por exemplo, na especificação da visão compartilhada considera-se que uma vez que os agentes sejam do tipo *LearningOrgAgent* e participem de sessões de diálogo e discussão, então deve emergir uma visão compartilhada entre os agentes envolvidos no processo. Essa condição é um pré-requisito para que uma determinada organização possa ser caracterizada como uma Organização Aprendiz.

- O modelo

Nesse modelo, conflitos potenciais entre planos pessoais e planos associados aos papéis não são levados em conta. Outrossim, para possíveis conflitos entre os objetivos e planos definidos em papéis distintos, caso esses papéis venham a ser desempenhados pelo mesmo agente em uma organização. De fato há outros pressupostos implícitos no caso dos papéis: que os planos e metas definidos em cada papel sejam consistentes, e também que o conjunto de recursos gerenciados pelo papel seja coerente com esses planos e metas.

O processo de desenvolvimento dos agentes para que se transformem em agentes do tipo *LearningOrgAgent* é modelado pela aplicação da função *entityLearnDisciplinesActions AndPrinciples*. Entretanto, como conseqüência da aplicação dessa função, todos os agentes da *LearningOrg* têm as mesmas capacidades e conhecimentos relativos às disciplinas de Senge.

A definição das camadas que apresentam a incorporação de novas capacidades, tais como: memória, manipulação de planos, e participação em organizações, serve como apoio para a produção de um modelo mais rico do agente, que pode ser situado em uma organização. Entretanto, as definições detalhadas dos elementos que compõem essas camadas de apoio não constam da teoria da Quinta Disciplina. Esses elementos foram construídos com base em elementos apresentados em [dL01] e conceitos básicos de teoria organizacional presentes em [Rob00, Chi00].

CAPÍTULO 6

Estudo de Caso

Neste capítulo é apresentado um estudo de caso, que foi desenvolvido a partir do modelo formal exposto no capítulo anterior. Inicialmente, é descrito o estudo de caso, em conjunto com as simplificações efetuadas no modelo apresentado no capítulo 5 visando a realização deste estudo. Na seção 6.2, é apresentada a implementação desse modelo simplificado e alguns dos testes realizados com seus respectivos resultados. Na seção final, são apresentados comentários sobre este estudo, a técnica utilizada e suas limitações e implicações. O material apresentado nesse capítulo, em forma resumida e simplificada, foi apresentado em [SS04a].

6.1 Introdução

Para este estudo de caso, o nível de abstração do modelo apresentado no capítulo passado foi modificado, de forma a representar apenas o nível de detalhamento essencial para a apresentação deste estudo.

A ferramenta utilizada para realizar este estudo é: ZETA versão 1.5 [GB03]. Esta ferramenta foi escolhida por permitir que fossem definidas valorações para os diversos tipos presentes no modelo formal, por exemplo, *Agent*, *LTeam*, entre outros.

O estudo de caso apresentado neste trabalho define uma lanchonete fictícia. Esta lanchonete serve, então, como um exemplo de utilização do modelo formal da Quinta Disciplina. A lanchonete

é composta por três empregados: um gerente, um atendente de balcão e um cozinheiro. Esses três também compõem a única equipe da organização.

Os empregados são modelados como agentes de um tipo formal que corresponde a uma versão simplificada do tipo *LearningOrgAgent*. Este novo tipo aglutina, num único esquema, as variáveis e predicados definidos nas diversas camadas de agentes do modelo formal do capítulo passado. A função dessa mudança de abstração é a de simplificar o processo de produção do estudo de caso. Essa mudança de abstração é mais adequada para a construção do estudo de caso, embora a estratégia de modularização e estruturação em camadas, adotada para a construção do modelo formal da Quinta Disciplina, seja mais legível e didática. Além disso, os parágrafos textuais que antecedem a apresentação dos parágrafos formais deste estudo de caso são mais sucintos, pois referem-se a definições de conceitos já expostos no capítulo anterior.

Este estudo permitiu validar a especificação em relação a alguns dos pressupostos da teoria de Senge, uma vez que existem estados representando uma situação inicial para o agente, modelos mentais, domínio pessoal, e visão compartilhada entre outros, e que as operações incluídas no estudo de caso podem ser efetuadas, produzindo estados válidos no sistema.

Os testes efetuados representam situações de atendimento, gerência e cozinha. Assim, para cada par incluindo um estado de ambiente e uma ação, obtém-se um novo estado e é selecionada uma nova ação pelo agente.

6.2 Implementação

Nesta seção, é apresentada a implementação do estudo de caso, os testes e seus resultados.

6.2.1 Variáveis Genéricas e Tipos Básicos

São definidas, inicialmente, as variáveis genéricas e os tipos básicos. Na próxima seção, esses tipos são refinados para permitir a definição do estudo de caso.

As constantes genéricas descritas aqui, introduzem operações que servem de apoio para a construção do modelo. As funções *mapset* e *mapseq* foram definidas em [dL01] e são reproduzidas aqui apenas para facilitar a leitura da especificação. A função *mapset* recebe uma função e um conjunto e aplica a função a cada elemento do conjunto. De modo similar, *mapseq* recebe uma função e uma seqüência e aplica a função a cada elemento da seqüência.

$$[X, Y]$$

$$mapseq == \lambda f : \mathsf{assumed}\ X \pfun Y \bullet$$
$$\lambda xs : \mathsf{assumed}\ \mathsf{seq}\ X \bullet$$
$$\{n : \mathbb{N} \mid n \in 1 \mathinner{..} \#xs \bullet (n, f(xsn))\}$$

$$[X, Y]$$

$$mapset == \lambda f : \mathsf{assumed}\ X \pfun Y \bullet$$
$$\lambda xs : \mathsf{assumed}\ \mathbb{P}\ X \bullet$$
$$\{x : X \mid x \in xs \bullet fx\}$$

As funções *trifirst*, *trisecond* e *trithird* recebem uma tripla e devolvem, respectivamente, o primeiro, o segundo e o terceiro elemento da tripla.

$$[X, Y, V]$$

$$trifirst == \lambda x : \mathsf{assumed}\ X;\ y : \mathsf{assumed}\ Y;\ v : \mathsf{assumed}\ V \bullet x$$

$$[X, Y, V]$$

$$trisecond == \lambda x : \mathsf{assumed}\ X;\ y : \mathsf{assumed}\ Y;\ v : \mathsf{assumed}\ V \bullet y$$

$$[X, Y, V]$$

$$trithird == \lambda x : \mathsf{assumed}\ X;\ y : \mathsf{assumed}\ Y;\ v : \mathsf{assumed}\ V \bullet v$$

As funções *tetrafirst*, *tetrasecond*, *tetrathird* e *tetrafourth* recebem uma tupla com quatro elementos e devolvem, respectivamente, o primeiro, o segundo, o terceiro e o quarto elemento da

tupla.

$$[X, Y, V, W]$$

$$tetrafirst == \lambda\, x : \text{assumed } X;\ y : \text{assumed } Y;\ v : \text{assumed } V;\ w : \text{assumed } W \bullet x$$

$$[X, Y, V, W]$$

$$tetrasecond == \lambda\, x : \text{assumed } X;\ y : \text{assumed } Y;\ v : \text{assumed } V;\ w : \text{assumed } W \bullet y$$

$$[X, Y, V, W]$$

$$tetrathird == \lambda\, x : \text{assumed } X;\ y : \text{assumed } Y;\ v : \text{assumed } V;\ w : \text{assumed } W \bullet v$$

$$[X, Y, V, W]$$

$$tetrafourth == \lambda\, x : \text{assumed } X;\ y : \text{assumed } Y;\ v : \text{assumed } V;\ w : \text{assumed } W \bullet w$$

A seguir, são introduzidos alguns dos tipos básicos usados neste estudo de caso. Assim, são introduzidos tipos para ação, átomo, predicados, motivação, agentes, papéis e recursos. Também são definidos o ambiente, planos e tipos básicos que introduzem as disciplinas de Senge, cujas definições serão, posteriormente, detalhadas.

$[Const, Var, FunSym, PredSym, Agent, AGENTNAME, Action]$
$[Motivation, Atom, Role, RoleOrg, Resource]$
$Attribute == \mathbb{P}\, Atom$
$Goal == \mathbb{P}_1\, Attribute$
$Actions == \mathbb{P}\, Action$
$Environment == \mathbb{P}_1\, Attribute$
$View == \mathbb{P}_1\, Attribute$
$Plan == \text{seq}\, Action$
$[PersonalMastery, PersonalVision, GuidingIdea, SharedVision, TEAM]$
$[SystemsThinking, IntraPersonalDisciplines, InterPersonalDisciplines]$

6.2.2 Refinamentos Sobre os Tipos Básicos

Nesta seção, os tipos básicos apresentados acima, são refinados. O processo é o mesmo para todos os tipos e é definido a seguir.

- Para um tipo básico que não será detalhado, basta definir algumas instâncias que correspondam a estes tipos. Por exemplo:

$$TipoBasico_1 ::= Exemplo_1DeTipoBasico_1 \mid Exemplo_2DeTipoBasico_1$$

- Para um tipo básico tb é definida uma função construtora tbf e um esquema $tbSchema$, que detalha o tipo. Assim o tipo pode ser refinado como:

$$tb ::= tbf\langle\langle tbSchema\rangle\rangle$$

Inicialmente, são definidos três tipos de recursos e motivações, e três nomes de agentes para os quais serão produzidas valorações posteriormente. As declarações das variáveis $motivationServices$, $motivationCook$ e $motivationManager$, definem os conjuntos associados às motivações de cada um

dos agentes. Embora, como uma simplificação, esses conjuntos sejam definidos de forma idêntica neste estudo, nada impede que eles sejam definidos a partir de qualquer combinação dos tipos de motivações definidos no modelo.

$AGENTNAME ::= CarlosCook \mid OsvaldoServs \mid MariManager$
$Motivation ::= ProfessionalDevelopment \mid EarnMoney$
$motivationServices == \{EarnMoney\}$
$motivationCook == \{EarnMoney\}$
$motivationManager == \{EarnMoney\}$
$Resource ::= Resource1 \mid Resource2 \mid Resource3$
$Consistency ::= yes \mid no$

Em seguida, os tipos básicos que precisam ser mais detalhados são apresentados com suas respectivas funções construtoras e esquemas que permitem esse detalhamento. Tomando como exemplo o tipo *GuidingIdea*, sua função construtora é denominada *guidingidea* e o seu esquema é denominado *GuidingIdeaSchema*. Este esquema é definido um pouco adiante. Para permitir a construção do estudo de caso, são definidas valorações associadas a este esquema. Analogamente, o mesmo processo é adotado para cada um dos tipos declarados a seguir.

$Atom ::= atom\langle\langle AtomSchema\rangle\rangle$
$RoleOrg ::= roleorg\langle\langle RoleOrgSchema\rangle\rangle$
$Agent ::= agent\langle\langle AgentSchema\rangle\rangle$
$AgentState ::= agentstate\langle\langle AgentStateSchema\rangle\rangle$
$AgentModelBase ::= agentmodel\langle\langle AgentModelSchema\rangle\rangle$
$GuidingIdea ::= guidingidea\langle\langle GuidingIdeaSchema\rangle\rangle$
$Vision ::= vision\langle\langle VisionSchema\rangle\rangle$
$Purpose ::= purpose\langle\langle PurposeSchema\rangle\rangle$
$Value ::= value\langle\langle ValueSchema\rangle\rangle$
$MentalModel ::= mentalmodel\langle\langle MentalModelSchema\rangle\rangle$
$PersonalVision ::= personalvisiongen\langle\langle PersonalVisionSchema\rangle\rangle$
$PersonalMastery ::= personalmasterygen\langle\langle PersonalMasterySchema\rangle\rangle$
$SystemsThinking ::= systemsthinkinggen\langle\langle SystemsThinkingSchema\rangle\rangle$
$SharedVision ::= sharedvisiongen\langle\langle SharedVisionSchema\rangle\rangle$
$LTeam ::= lteamgen\langle\langle LTeamSchema\rangle\rangle$
$LOrg ::= lorggen\langle\langle LOrgSchema\rangle\rangle$
$AGPredicate ::= positive\langle\langle Atom\rangle\rangle \mid negative\langle\langle Atom\rangle\rangle$
$AGBelief ::= agpredicate\langle\langle AGPredicate\rangle\rangle \mid conjunct\langle\langle AGBelief \times AGBelief\rangle\rangle$
$BehavioralConstraint == \mathbb{P}\, AGBelief$

As diversas ações que podem ser executadas pelos agentes da organização, são apresentadas abaixo. São incluídas aqui, tanto ações relativas ao desenvolvimento das disciplinas de Senge, quanto as ações associadas ao desempenho de atividades específicas por parte dos agentes, tais como: recepção de um novo cliente, recebimento de um pedido, preparo de lanches, e gerenciamento da lanchonete.

Action ::= *Attend* | *Manage* | *Cook* |
DevelopGuidingIdeasActions |
EnhanceRealityVisionsActions |
ClarifyPersonalVisionsActions |
MentalModelsActions | *TeamLearningActions* |
SystemsThinkingActions | *SharedVisionActions* |
ReflectionActions |
ArgumentationActions |
PerceptiveActions |
ReceiveNewCustomer |
GreetTheCustomer | *DisplayMenu* |
ReceiveCustomersQuestions |
AnswerCustomersQuestions |
InquiryDoubtsOfTheCustomer |
ReceiveOrder | *SuggestSideDishes* |
InformInvalidOrder | *ReceiveOrderConfirmation* |
ReceiveCustomerGivesUp |
ConfirmOrder | *InformAmountOfTheBill* |
ConfirmOrderWithPrice | *ForwardOrderToKitchen* |
CheckStateOfOrder | *ReceivePayment* | *CheckPayment* |
ComputeChange | *ReturnChange* | *ReceivePreparedDish* |
DeliverDishToCustomer | *OfferAdditionalSeasonings* |
GoodAppetiteSeeYouSoon |
ReceiveAnOrderForCooking |
PrepareMeal | *DeliverMeal* |
CheckStockLevelOfPreviouslyCookedSideDishes |
OpenTheShop | *CloseTheShop* |
EvaluateCustomerSatisfaction | *EvaluateCustomerService* |
EvaluateCookService |
DefineMinimumStockLevelOfPreviouslyCookedSideDishes |
DefineQualityStandardsForServices |
DefineQualityStandardsForMeals |
DefineRewardsForWellPerformedWork | *ManageTheShop*

Em seguida são especificadas as ações e princípios das disciplinas. As ações são definidas com base nas declarações efetuadas logo acima. Quanto aos princípios, as declarações destes se baseiam em tipos que serão definidos mais adiante como valorações específicas de conjuntos do

tipo *Attribute*.

$$
\begin{array}{|l}
AllDisciplinesActions, PMActions : \mathbb{P}\, Action \\
STActions, MMActions, SVActions, LTActions : \mathbb{P}\, Action \\
\hline
PMActions = \{DevelopGuidingIdeasActions, EnhanceRealityVisionsActions, \\
\qquad ClarifyPersonalVisionsActions\}; \\
MMActions = \{MentalModelsActions, ReflectionActions, ArgumentationActions\}; \\
STActions = \{SystemsThinkingActions\}; \\
LTActions = \{TeamLearningActions\}; \\
SVActions = \{SharedVisionActions\}; \\
PMActions \neq \varnothing \land STActions \neq \varnothing \land MMActions \neq \varnothing \land SVActions \neq \varnothing \land LTActions \neq \varnothing \\
AllDisciplinesActions = PMActions \cup STActions \cup MMActions \cup SVActions \cup LTActions
\end{array}
$$

$$
\begin{array}{|l}
AllDisciplinesPrinciples, PMPrinciples : \mathbb{P}\, Attribute \\
STPrinciples, MMPrinciples : \mathbb{P}\, Attribute \\
SVPrinciples, LTPrinciples : \mathbb{P}\, Attribute \\
\hline
PMPrinciples = \{\{PersonalMasteryPrinciples\}\}; \\
STPrinciples = \{\{SystemsThinkingPrinciples\}\}; \\
MMPrinciples = \{\{MentalModelsPrinciples\}\}; \\
SVPrinciples = \{\{SharedVisionPrinciples\}\}; \\
LTPrinciples = \{\{TeamLearningPrinciples\}\}; \\
PMPrinciples \neq \varnothing \land \\
STPrinciples \neq \varnothing \land \\
MMPrinciples \neq \varnothing \land \\
SVPrinciples \neq \varnothing \land \\
LTPrinciples \neq \varnothing \\
AllDisciplinesPrinciples = PMPrinciples \cup STPrinciples \cup \\
\qquad MMPrinciples \cup SVPrinciples \cup LTPrinciples
\end{array}
$$

DisciplinesActionsAndPrinciples

$$
\begin{array}{|l}
actions : \mathbb{P}\, Action \\
principles : \mathbb{P}\, Attribute \\
\hline
actions \subseteq AllDisciplinesActions \\
principles \subseteq AllDisciplinesPrinciples \\
actions \neq \varnothing \\
principles \neq \varnothing
\end{array}
$$

Agora são apresentados os tipos básicos relativos a crenças e modelos, necessários para a especificação de modelos mentais, incluindo modelos para entidades, agentes, e interações confiáveis e não confiáveis.

$$AgentBelief ::= agbeliefs\langle\langle AGBelief\rangle\rangle$$
$$OrgBelief ::= orgbeliefs\langle\langle AGBelief\rangle\rangle$$
$$ExposedBelief ::= expbeliefs\langle\langle AGBelief\rangle\rangle$$
$$LearningOrgAgent == Agent$$
$$ReasoningProcess == \mathrm{seq}_1\ AGBelief$$
$$ReflectedBeliefReasoning == AGBelief \times ReasoningProcess$$
$$Entity == Agent$$
$$ExposedReasoningProcess == \mathrm{seq}_1\ ExposedBelief$$
$$ExposedBeliefReasoning == ExposedBelief \times ExposedReasoningProcess$$
$$EntityModel == AgentModelBase$$
$$AgentModel == AgentModelBase$$
$$ComponentModel ::= agm\langle\langle AgentModel\rangle\rangle \mid em\langle\langle EntityModel\rangle\rangle$$
$$Component ::= compAgent\langle\langle Agent\rangle\rangle$$
$$ComponentRelationship ::= compRelationship\langle\langle Component \times Component\rangle\rangle$$
$$ComponentRelationshipModel ::= compRelationshipModel\langle\langle ComponentModel \times ComponentModel\rangle\rangle$$
$$BeliefAndReasoning == AGBelief \times ReasoningProcess$$
$$LearningOrgAgentModel == AgentModel$$
$$LearnComponent ::= compLearningOrgAgent\langle\langle LearningOrgAgent\rangle\rangle \mid compGeneral\langle\langle Component\rangle\rangle$$
$$LearnComponentRelationship ::= compLearnRelationship\langle\langle LearnComponent \times LearnComponent\rangle\rangle$$
$$LearnComponentModel ::= cm\langle\langle ComponentModel\rangle\rangle \mid loam\langle\langle LearningOrgAgentModel\rangle\rangle$$
$$LearnComponentRelationshipModel ::= compLearnRelationshipModel\langle\langle LearnComponentModel \times LearnComponentModel\rangle\rangle$$
$$[LOrgAgentInteractions]$$
$$[TrustLOrgAgentInter, NonTrustLOrgAgentInter]$$
$$LOrgAgentInteractionsModel == LOrgAgentInteractions$$
$$TrustLOrgAgentInterModel == TrustLOrgAgentInter$$
$$NonTrustLOrgAgentInterModel == NonTrustLOrgAgentInter$$

O esquema que define os modelos mentais para este estudo de caso é apresentado a seguir. Entre as simplificações adotadas estão a exclusão das funções *reflection* e *advocacy*, que não serão utilizadas neste estudo.

MentalModelSchema

$$
\begin{array}{l}
developReflectionSkillsActions : \mathbb{P}\ Action \\
developAdvocacySkillsActions : \mathbb{P}\ Action \\
intraPersonalMentalModelPrinciples : \mathbb{P}\ Attribute \\
models : \mathbb{P}\ EntityModel \\
agentbeliefs : \mathbb{P}\ AGBelief \\
modelentities : \mathbb{P}\ EntityModel \\
modelagents : \mathbb{P}\ AgentModel \\
modellearningorgagents : \mathbb{P}_1\ LearningOrgAgentModel \\
modelbelmodel : AgentModel \nrightarrow LearningOrgAgentModel \\
modelLOrgAgInts : \mathbb{P}(LearningOrgAgentModel \times LearningOrgAgentModel) \\
modelTrustLOrgAgInts : \mathbb{P}(LearningOrgAgentModel \times LearningOrgAgentModel) \\
modelNonTrustLOrgAgInts : \mathbb{P}(LearningOrgAgentModel \times LearningOrgAgentModel) \\
\hline
developReflectionSkillsActions \cup \\
\quad developAdvocacySkillsActions = MMActions \\
modelLOrgAgInts = modelTrustLOrgAgInts \cup \\
\qquad modelNonTrustLOrgAgInts; \\
modelTrustLOrgAgInts \cap modelNonTrustLOrgAgInts = \varnothing;
\end{array}
$$

Quanto às visões, propósitos e valores, as funções de avaliação de consistência são apresentadas a seguir. É importante notar que, os domínios e imagens das funções já estão completamente especificados, de acordo com os dados do presente estudo de caso. Por exemplo, segundo estas definições, as visões, propósitos e valores do atendente são consistentes.

$$
\begin{array}{l}
isVisionPurposeConsistent : (\mathbb{P}\ Vision \times \mathbb{P}\ Purpose) \rightarrow Consistency \\
\hline
isVisionPurposeConsistent = \{((\{vision1\}, \{purpose1\}), yes), \\
\quad ((\{visionsOsvaldoServs01\}, \{purposesOsvaldoServs01\}), yes), \\
\quad ((\{visionsCarlosCook01\}, \{purposesCarlosCook01\}), yes), \\
\quad ((\{visionsMariManager01\}, \{purposesMariManager01\}), yes)\};
\end{array}
$$

$$isValueVisionConsistent : (\mathbb{P}\ Value \times \mathbb{P}\ Vision) \rightarrow Consistency$$

$$\begin{aligned} isValueVisionConsistent = \{&((\{value1\}, \{vision1\}), yes), \\ &((\{value2\}, \{vision1\}), yes), \\ &((\{valuesOsvaldoServs01\}, \{visionsOsvaldoServs01\}), yes), \\ &((\{valuesCarlosCook01\}, \{visionsCarlosCook01\}), yes), \\ &((\{valuesMariManager01\}, \{visionsMariManager01\}), yes)\}; \end{aligned}$$

$$isValuePurposeConsistent : (\mathbb{P}\ Value \times \mathbb{P}\ Purpose) \rightarrow Consistency$$

$$\begin{aligned} isValuePurposeConsistent = \{&((\{value1\}, \{purpose1\}), yes), \\ &((\{value2\}, \{purpose1\}), yes), \\ &((\{valuesOsvaldoServs01\}, \{purposesOsvaldoServs01\}), yes), \\ &((\{valuesCarlosCook01\}, \{purposesCarlosCook01\}), yes), \\ &((\{valuesMariManager01\}, \{purposesMariManager01\}), yes)\}; \end{aligned}$$

Em seguida, são apresentados os esquemas para visão, propósito, valores, e diretrizes.

VisionSchema

$$visions : \mathbb{P}\ Goal$$

PurposeSchema

$$purposes : \mathbb{P}\ Goal$$

ValueSchema

$$values : \mathbb{P}\ BehavioralConstraint$$

GuidingIdeaSchema

$$visions : \mathbb{P}\ Vision$$
$$purposes : \mathbb{P}\ Purpose$$
$$values : \mathbb{P}\ Value$$

$$\begin{aligned} &isVisionPurposeConsistent(visions, purposes) = yes \wedge \\ &\quad isValueVisionConsistent(values, visions) = yes \wedge \\ &\quad isValuePurposeConsistent(values, purposes) = yes \end{aligned}$$

Com o intuito de apresentar a definição do esquema associado à disciplina de domínio pessoal, inicialmente, é especificado o esquema relativo à visão pessoal do agente.

PersonalVisionSchema

$$personalvision : \mathbb{P}\ Motivation \pfun MentalModel \pfun GuidingIdea \pfun \mathbb{P}\ Goal$$

O esquema que especifica o domínio pessoal para este estudo é apresentado a seguir. Ele é similar ao apresentado no capítulo anterior. Houve mudanças apenas nos conjuntos imagem das funções *enhancerealityvisions* e *creativetension*. Nessas funções os tipos *RealityVision* e *ResolutionGoal* foram substituídos, respectivamente, pelos tipos básicos nos quais suas definições se baseiam.

PersonalMasterySchema

$$\begin{aligned}
&developguidingideasactions : \mathbb{P}\ Action \\
&enhancerealityvisionsactions : \mathbb{P}\ Action \\
&clarifypersonalvisionsactions : \mathbb{P}\ Action \\
&personalMasteryPrinciples : \mathbb{P}\ Attribute \\
&developguidingideas : \mathbb{P}\ Motivation \times \mathbb{P}\ Action \pfun \\
&\qquad GuidingIdea \\
&enhancerealityvisions : \mathbb{P}\ Motivation \times \mathbb{P}\ Action \pfun \\
&\qquad \mathbb{P}(\mathbb{P}\ View) \\
&clarifypersonalvisions : \mathbb{P}\ Motivation \times \mathbb{P}\ Action \pfun \\
&\qquad \mathbb{P}\ PersonalVision \\
&creativetension : \mathbb{P}\ View \times \mathbb{P}\ PersonalVision \pfun \\
&\qquad \mathbb{P}(\mathbb{P}\ Goal) \\
&personalvisions : \mathbb{P}\ PersonalVision \\
&guidingidea : GuidingIdea; \\
\hline
&developguidingideasactions \cup \\
&\quad enhancerealityvisionsactions \cup \\
&\quad clarifypersonalvisionsactions = PMActions;
\end{aligned}$$

A seguir é introduzido o esquema relativo ao pensamento sistêmico. Neste esquema também houve uma mudança no nível de abstração da especificação: todas as funções declaradas no esquema presente no capítulo 5 foram substituídas por uma única função que mapeia percepções do ambiente para um conjunto de planos. Como este estudo não explora características específicas da disciplina de pensamento sistêmico, esse nível de abstração é suficiente neste caso.

SystemsThinkingSchema

$$producePlans : View \pfun \mathbb{P}\, Plan$$

A visão compartilhada envolvendo um conjunto de equipes e diretrizes é apresentada a seguir.

SharedVisionSchema

$$learningteams : \mathbb{P}_1\, LTeam$$
$$interTeamsGuidingIdeas : \mathbb{P}\, LTeam \rightarrow GuidingIdea$$
$$sharedvisions : \mathbb{P}\, Vision$$
$$visionsFromGuidingIdea : \mathbb{P}\, GuidingIdea \rightarrow \mathbb{P}\, Vision$$

Além disso, o tipo *Atom* precisa ser especificado, pois é utilizado na instanciação de crenças, metas, e atributos.

AtomSchema

$$head : PredSym$$
$$terms : \text{seq}\, Term$$

6.2.3 Valorações Iniciais

Nesta seção são apresentadas diversas valorações iniciais que servem de apoio para a construção deste estudo de caso.

Para o tipo *Atom* há várias definições de valorações que se referem a predicados, metas e atributos. Os predicados são usados para definir o estado atual do ambiente, os objetivos dos agentes ou ainda valores pessoais, tais como honestidade, integridade, etc.. Além disso, são declaradas algumas constantes associadas ao tipo *Const*, tais como: $Customer1$, $Order1$, e *Meals*.

$$Term ::= const\langle\langle Const\rangle\rangle \mid var\langle\langle Var\rangle\rangle \mid functor\langle\langle FunSym \times \text{seq}\, Term\rangle\rangle$$

$$
\begin{aligned}
PredSym ::= {} & Pred1 \mid Pred2 \mid Pred3 \mid Pred4 \mid \\
& WaitingForService \mid WaitingForInformation \mid \\
& SelectingMeal \mid ConfirmingOrder \mid ConfirmedOrder \mid \\
& WaitingCustomerPayment \mid Served \mid BeingServed \mid \\
& HasPaid \mid PlacedOrder \mid WaitingForMeal \mid DeliveredOrderToKitchen \mid \\
& OrderReadyFromKitchen \mid Idle \mid AnalyzingOrder \mid Cooking \mid \\
& ShopClosed \mid ShopOpen \mid \\
& DefinedQualityStandardsForServices \mid DefinedQualityStandardsForMeals \mid \\
& DefinedRewardsForWellPerformedWork \mid EvaluatedCustomerService \mid \\
& EvaluatedCustomerSatisfaction \mid \\
& Honesty \mid EthicalPractice \mid Integrity \mid \\
& Reputation \mid ServeGoodMeals \mid ServeLotsOfMeals \mid \\
& ServeBestQualityMeals \mid CookBestQualityMeals \mid OfferBestFastFoodService \mid \\
& ImproveProfits \mid ImproveSalary \mid \\
& CareerDevelopment \mid \\
& DevelopInterpersonalSkills \mid Wealth \mid Status \\
Const ::= {} & Const1 \mid Const2 \mid Const3 \mid Const4 \mid \\
& Customer1 \mid Customer2 \mid Order1 \mid Order2 \mid \\
& Orders \mid Meals \mid Shop \mid Worker \mid Self
\end{aligned}
$$

Em seguida são introduzidas algumas metas que incluem: servir bons lanches, lanches de boa qualidade, aumentar lucros e salários.

$$
\begin{aligned}
& atServeGoodMeals == atom\langle\!| \; head == ServeGoodMeals, \\
& \qquad terms == \langle const(Meals)\rangle \; |\!\rangle \\
& atServeLotsOfMeals == atom\langle\!| \; head == ServeLotsOfMeals, \\
& \qquad terms == \langle const(Meals)\rangle \; |\!\rangle \\
& atServeBestQualityMeals == atom\langle\!| \; head == \\
& \qquad ServeBestQualityMeals, terms == \langle const(Meals)\rangle \; |\!\rangle \\
& atCookBestQualityMeals == atom\langle\!| \; head == \\
& \qquad CookBestQualityMeals, terms == \langle const(Meals)\rangle \; |\!\rangle \\
& atOfferBestFastFoodService == atom\langle\!| \; head == \\
& \qquad OfferBestFastFoodService, terms == \langle const(Shop)\rangle \; |\!\rangle \\
& atImproveProfits == atom\langle\!| \; head == ImproveProfits, \\
& \qquad terms == \langle const(Shop)\rangle \; |\!\rangle \\
& atImproveSalary == atom\langle\!| \; head == ImproveSalary, \\
& \qquad terms == \langle const(Worker)\rangle \; |\!\rangle
\end{aligned}
$$

Além disso são definidos objetivos de mais alto nível, tais como: o desenvolvimento da carreira e relacionamentos interpessoais, riqueza e status.

$$
\begin{array}{l}
atCareerDevelopment == atom(\!| \, head == CareerDevelopment, \\
\qquad\qquad terms == \langle const(Self) \rangle \, |\!) \\
atDevelopInterpersonalSkills == atom(\!| \, head == \\
\qquad\qquad DevelopInterpersonalSkills, terms == \langle const(Self) \rangle \, |\!) \\
atWealth == atom(\!| \, head == Wealth, \\
\qquad\qquad terms == \langle const(Self) \rangle \, |\!) \\
atStatus == atom(\!| \, head == Status, \\
\qquad\qquad terms == \langle const(Self) \rangle \, |\!)
\end{array}
$$

Adicionalmente, são necessárias valorações para átomos que serão usados na definição de valores que incluem: honestidade, integridade, comportamento ético e reputação.

$$
\begin{array}{l}
atHonesty == atom(\!| \, head == Honesty, \\
\qquad\qquad terms == \langle const(Self) \rangle \, |\!) \\
atEthicalPractice == atom(\!| \, head == EthicalPractice, \\
\qquad\qquad terms == \langle const(Self) \rangle \, |\!) \\
atIntegrity == atom(\!| \, head == Integrity, \\
\qquad\qquad terms == \langle const(Self) \rangle \, |\!) \\
atReputation == atom(\!| \, head == Reputation, \\
\qquad\qquad terms == \langle const(Self) \rangle \, |\!)
\end{array}
$$

Para permitir a definição de estados do ambiente, são declarados os átomos a seguir. Desta forma, são incluídas definições para estados que indicam atendente desocupado, cliente $Customer1$ a espera de atendimento, cliente $Customer1$ a espera do lanche, pedido pronto e entregue pela cozinha, lanchonete aberta, lanchonete fechada, etc..

$$
\begin{array}{l}
atIdle == atom(\!| \, head == Idle, \\
\qquad\qquad terms == \langle const(Self) \rangle \, |\!) \\
atWaitingServiceC1 == atom(\!| \, head == WaitingForService, \\
\qquad\qquad terms == \langle const(Customer1) \rangle \, |\!) \\
atWaitingInfoC1 == atom(\!| \, head == WaitingForInformation, \\
\qquad\qquad terms == \langle const(Customer1) \rangle \, |\!)
\end{array}
$$

$$
\begin{aligned}
&atSelectingMealC1 == atom\langle\!|\ head == SelectingMeal,\\
&\qquad terms == \langle const(Customer1)\rangle\ |\!\rangle\\
&atConfirmingOrderC1 == atom\langle\!|\ head == ConfirmingOrder,\\
&\qquad terms == \langle const(Customer1)\rangle\ |\!\rangle\\
&atConfirmedOrderC1 == atom\langle\!|\ head == ConfirmedOrder,\\
&\qquad terms == \langle const(Customer1)\rangle\ |\!\rangle\\
&atWaitingCustomerPaymentC1 == atom\langle\!|\ head ==\\
&\quad WaitingCustomerPayment, terms == \langle const(Customer1)\rangle\ |\!\rangle\\
&atWaitingCustomerPaymentC2 == atom\langle\!|\ head ==\\
&\quad WaitingCustomerPayment, terms == \langle const(Customer2)\rangle\ |\!\rangle\\
&atServedC1 == atom\langle\!|\ head == Served,\\
&\qquad terms == \langle const(Customer1), const(Order1)\rangle\ |\!\rangle\\
&atBeingServedC1 == atom\langle\!|\ head == BeingServed,\\
&\qquad terms == \langle const(Customer1)\rangle\ |\!\rangle\\
&atHasPaidC1 == atom\langle\!|\ head == HasPaid,\\
&\qquad terms == \langle const(Customer1), const(Order1)\rangle\ |\!\rangle\\
&atPlacedOrderC1 == atom\langle\!|\ head == PlacedOrder,\\
&\qquad terms == \langle const(Customer1), const(Order1)\rangle\ |\!\rangle\\
&atWaitingForMealC1 == atom\langle\!|\ head == WaitingForMeal,\\
&\qquad terms == \langle const(Customer1), const(Order1)\rangle\ |\!\rangle\\
&atDeliveredOrderToKitchenO1 == atom\langle\!|\ head ==\\
&\quad DeliveredOrderToKitchen, terms == \langle const(Order1)\rangle\ |\!\rangle\\
&atOrderReadyFromKitchenO1 == atom\langle\!|\ head ==\\
&\quad OrderReadyFromKitchen, terms == \langle const(Order1)\rangle\ |\!\rangle\\
&atWaitingServiceC2 == atom\langle\!|\ head == WaitingForService,\\
&\qquad terms == \langle const(Customer2)\rangle\ |\!\rangle\\
&atWaitingInfoC2 == atom\langle\!|\ head == WaitingForInformation,\\
&\qquad terms == \langle const(Customer2)\rangle\ |\!\rangle\\
&atSelectingMealC2 == atom\langle\!|\ head == SelectingMeal,\\
&\qquad terms == \langle const(Customer2)\rangle\ |\!\rangle\\
&atConfirmingOrderC2 == atom\langle\!|\ head == ConfirmingOrder,\\
&\qquad terms == \langle const(Customer2)\rangle\ |\!\rangle\\
&atConfirmedOrderC2 == atom\langle\!|\ head == ConfirmedOrder,\\
&\qquad terms == \langle const(Customer2)\rangle\ |\!\rangle\\
&atServedC2 == atom\langle\!|\ head == Served,\\
&\qquad terms == \langle const(Customer2), const(Order2)\rangle\ |\!\rangle\\
&atBeingServedC2 == atom\langle\!|\ head == BeingServed,\\
&\qquad terms == \langle const(Customer2)\rangle\ |\!\rangle\\
&atHasPaidC2 == atom\langle\!|\ head == HasPaid,\\
&\qquad terms == \langle const(Customer2), const(Order2)\rangle\ |\!\rangle
\end{aligned}
$$

$$
\begin{aligned}
&atPlacedOrderC2 == atom\langle\!|\ head == PlacedOrder,\\
&\qquad terms == \langle const(Customer2), const(Order2)\rangle\ |\!\rangle\\
&atWaitingForMealC2 == atom\langle\!|\ head == WaitingForMeal,\\
&\qquad terms == \langle const(Customer2), const(Order2)\rangle\ |\!\rangle\\
&atDeliveredOrderToKitchenO2 == atom\langle\!|\ head ==\\
&\qquad DeliveredOrderToKitchen, terms == \langle const(Order2)\rangle\ |\!\rangle\\
&atOrderReadyFromKitchenO2 == atom\langle\!|\ head ==\\
&\qquad OrderReadyFromKitchen, terms == \langle const(Order2)\rangle\ |\!\rangle\\
&atAnalyzingOrderO1 == atom\langle\!|\ head == AnalyzingOrder, terms == \langle const(Order1)\rangle\ |\!\rangle\\
&atCookingO1 == atom\langle\!|\ head == Cooking, terms == \langle const(Order1)\rangle\ |\!\rangle\\
&atShopOpenS1 == atom\langle\!|\ head == ShopOpen, terms == \langle const(Shop)\rangle\ |\!\rangle\\
&atShopClosedS1 == atom\langle\!|\ head == ShopClosed, terms == \langle const(Shop)\rangle\ |\!\rangle\\
&atDefinedQualityStandardsForServices == atom\langle\!|\ head ==\\
&\qquad DefinedQualityStandardsForServices, terms == \langle const(Shop)\rangle\ |\!\rangle\\
&atDefinedQualityStandardsForMeals == atom\langle\!|\ head ==\\
&\qquad DefinedQualityStandardsForMeals, terms == \langle const(Shop)\rangle\ |\!\rangle\\
&atDefinedRewardsForWellPerformedWork == atom\langle\!|\ head ==\\
&\qquad DefinedRewardsForWellPerformedWork, terms == \langle const(Shop)\rangle\ |\!\rangle\\
&atEvaluatedCustomerService == atom\langle\!|\ head ==\\
&\qquad EvaluatedCustomerService, terms == \langle const(Shop)\rangle\ |\!\rangle\\
&atEvaluatedCustomerSatisfaction == atom\langle\!|\ head ==\\
&\qquad EvaluatedCustomerSatisfaction, terms == \langle const(Shop)\rangle\ |\!\rangle
\end{aligned}
$$

Em seguida, são declaradas algumas valorações, baseadas em átomos constituídos por constantes genéricas, que servem apenas para exemplificar como podem ser definidos os princípios das disciplinas de Senge. Neste estudo de caso, não há a necessidade de detalhamento de tais princípios.

$$
\begin{aligned}
&PersonalMasteryPrinciples == atom\langle\!|\ head == Pred2,\\
&\qquad terms == \langle const(Const1), const(Const2)\rangle\ |\!\rangle\\
&SystemsThinkingPrinciples == atom\langle\!|\ head == Pred2,\\
&\qquad terms == \langle const(Const1), const(Const2)\rangle\ |\!\rangle\\
&MentalModelsPrinciples == atom\langle\!|\ head == Pred2,\\
&\qquad terms == \langle const(Const1), const(Const2)\rangle\ |\!\rangle\\
&SharedVisionPrinciples == atom\langle\!|\ head == Pred2,\\
&\qquad terms == \langle const(Const1), const(Const2)\rangle\ |\!\rangle\\
&TeamLearningPrinciples == atom\langle\!|\ head == Pred2,\\
&\qquad terms == \langle const(Const1), const(Const2)\rangle\ |\!\rangle
\end{aligned}
$$

Com base nos átomos definidos acima é possível estabelecer então alguns possíveis estados de ambiente.

As valorações a seguir referem-se a descrições de ambientes em que o atendente está desocupado, cliente *Customer*1 está esperando para ser atendido, cliente *Customer*2 está esperando para ser atendido, ambos os clientes esperam para serem atendidos, e, finalmente, cliente *Customer*1 em processo de atendimento, e cliente *Customer*2 em processo de atendimento.

$$
\begin{array}{l}
envIdle == \{\{atIdle\}\} \\
envCust1\,WaitServ == \{\{atWaitingServiceC1\}\} \\
envCust2\,WaitServ == \{\{atWaitingServiceC2\}\} \\
envCusts\,WaitServ == \{\{atWaitingServiceC1, atWaitingServiceC2\}\} \\
envCust1BeingServed == \{\{atBeingServedC1\}\} \\
envCust2BeingServed == \{\{atBeingServedC2\}\}
\end{array}
$$

Além disso, os clientes podem estar selecionando uma refeição, aguardando por informações, ou podem ter apresentado seus pedidos.

$$
\begin{array}{l}
envCust1SelectMeal == \{\{atBeingServedC1, atSelectingMealC1\}\} \\
envCust2SelectMeal == \{\{atBeingServedC2, atSelectingMealC2\}\} \\
envCust1\,WaitForInfo == \{\{atBeingServedC1, atSelectingMealC1, atWaitingInfoC1\}\} \\
envCust2\,WaitForInfo == \{\{atBeingServedC2, atSelectingMealC2, atWaitingInfoC2\}\} \\
envCust1PlacedOrder == \{\{atBeingServedC1, atPlacedOrderC1\}\} \\
envCust2PlacedOrder == \{\{atBeingServedC2, atPlacedOrderC2\}\}
\end{array}
$$

O ambiente tem estado de "cliente está confirmando seu pedido" quando este cliente está sendo servido, já apresentou seu pedido e está em processo de confirmação desse pedido. A descrição é similar para o estado "cliente confirmou seu pedido".

$$
\begin{array}{l}
envCust1\,ConfirmingOrder == \{\{atBeingServedC1, \\
\qquad\qquad atPlacedOrderC1, atConfirmingOrderC1\}\} \\
envCust2\,ConfirmingOrder == \{\{atBeingServedC2, \\
\qquad\qquad atPlacedOrderC2, atConfirmingOrderC2\}\}
\end{array}
$$

$$
\begin{array}{l}
envCust1ConfirmedOrder == \{\{atBeingServedC1, \\
\qquad atPlacedOrderC1, atConfirmedOrderC1\}\} \\
envCust2ConfirmedOrder == \{\{atBeingServedC2, \\
\qquad atPlacedOrderC2, atConfirmedOrderC2\}\}
\end{array}
$$

Há também descrições para estados que envolvam o aguardo pelo pagamento e o pagamento efetuado por um dado cliente.

$$
\begin{array}{l}
envWaitForCust1Payment == \{\{atBeingServedC1, atPlacedOrderC1, \\
\qquad atConfirmedOrderC1, atWaitingCustomerPaymentC1\}\} \\
envWaitForCust2Payment == \{\{atBeingServedC2, atPlacedOrderC2, \\
\qquad atConfirmedOrderC2, atWaitingCustomerPaymentC2\}\} \\
envCust1HasPaid == \{\{atBeingServedC1, atPlacedOrderC1, atHasPaidC1\}\} \\
envCust2HasPaid == \{\{atBeingServedC2, atPlacedOrderC2, atHasPaidC2\}\}
\end{array}
$$

Os estados seguintes descrevem: clientes aguardando por suas refeições, refeições prontas para serem servidas, clientes já servidos, pedido em preparo na cozinha, lanchonete aberta, lanchonete fechada, definição e avaliação de padrões de atendimento e refeições.

$$
\begin{array}{l}
envCust1WaitForMeal == \{\{atBeingServedC1, atPlacedOrderC1, \\
\qquad atHasPaidC1, atWaitingForMealC1\}\} \\
envCust2WaitForMeal == \{\{atBeingServedC2, atPlacedOrderC2, \\
\qquad atHasPaidC2, atWaitingForMealC2\}\} \\
envOrder1Ready == \{\{atBeingServedC1, atPlacedOrderC1, \\
\qquad atHasPaidC1, atWaitingForMealC1, \\
\qquad atOrderReadyFromKitchenO1\}\} \\
envOrder2Ready == \{\{atBeingServedC2, atPlacedOrderC2, \\
\qquad atHasPaidC2, atWaitingForMealC2, \\
\qquad atOrderReadyFromKitchenO2\}\} \\
envCust1Served == \{\{atServedC1\}\} \\
envCust2Served == \{\{atServedC2\}\} \\
envOrder1BeingAnalyzed == \{\{atAnalyzingOrderO1\}\} \\
envOrder1BeingCooked == \{\{atCookingO1\}\} \\
envShopOpen == \{\{atShopOpenS1\}\}
\end{array}
$$

$envDefinedQualityStandardsForServices == \{\{atShopOpenS1,$
$\quad atDefinedQualityStandardsForServices\}\}$
$envDefinedQualityStandardsForMeals == \{\{atShopOpenS1,$
$\quad atDefinedQualityStandardsForMeals\}\}$
$envDefinedRewardsForWellPerformedWork == \{\{atShopOpenS1,$
$\quad atDefinedRewardsForWellPerformedWork\}\}$
$envEvaluatedCustomerService == \{\{atShopOpenS1,$
$\quad atEvaluatedCustomerService\}\}$
$envEvaluatedCustomerSatisfaction == \{\{atShopOpenS1,$
$\quad atEvaluatedCustomerSatisfaction\}\}$
$envShopRunning == \{\{atShopOpenS1,$
$\quad atDefinedQualityStandardsForServices,$
$\quad atDefinedQualityStandardsForMeals,$
$\quad atDefinedRewardsForWellPerformedWork,$
$\quad atEvaluatedCustomerService,$
$\quad atEvaluatedCustomerSatisfaction\}\}$
$envShopClosed == \{\{atShopClosedS1\}\}$

Adicionalmente, também são declaradas, de um modo genérico, as percepções de cada agente.

$agentviews1 == \{\{atom1, atom2\}\};$

Nesse ponto são declaradas valorações que definem conjuntos de metas associados ao atendente, gerente e cozinheiro.

$agentgoalsServices == \{\{\{atServeLotsOfMeals, atImproveSalary,$
$\quad atCareerDevelopment\}\}\}$
$agentgoalsCook == \{\{\{atServeLotsOfMeals, atImproveSalary,$
$\quad atCareerDevelopment\}\}\};$
$agentgoalsManager == \{\{\{atServeLotsOfMeals, atImproveSalary,$
$\quad atCareerDevelopment\}\}\};$

Quanto às ações perceptivas dos agentes, estas são definidas, no nível de abstração adequado a este estudo, apenas como um único tipo de ação.

$peractions1a == \{PerceptiveActions\}$

A função *effectinteraction*, que mapeia estados do ambiente para ações e novos estados, é definida como segue. Por exemplo, a partir de um ambiente $envCust1\,WaitServ$, se o agente efetuar a ação $GreetTheCustomer$, então o estado do ambiente passará a ser $envCust1BeingServed$. Se, a partir de qualquer estado do ambiente, o agente efetuar ações perceptivas $PerceptiveActions$, o ambiente permanecerá nesse mesmo estado. É importante notar que há a necessidade de se especificar cada mapeamento válido envolvendo um dado estado, uma ação e um novo estado do ambiente.

$$
\begin{array}{|l}
effectinteraction : Environment \rightarrow Actions \rightarrow Environment \\
\hline
effectinteraction = \\
\{(envIdle, \{ \\
\quad (\{ReceiveNewCustomer\}, envCust1\,WaitServ), \\
\quad (\{ReceiveAnOrderForCooking\}, envOrder1BeingAnalyzed), \\
\quad (\{PerceptiveActions\}, envIdle)\}), \\
(envCust1\,WaitServ, \{ \\
\quad (\{GreetTheCustomer\}, envCust1BeingServed), \\
\quad (\{PerceptiveActions\}, envCust1\,WaitServ)\}), \\
(envCust2\,WaitServ, \{ \\
\quad (\{GreetTheCustomer\}, envCust2BeingServed), \\
\quad (\{PerceptiveActions\}, envCust2\,WaitServ)\}), \\
(envCustsWaitServ, \{ \\
\quad (\{GreetTheCustomer\}, envCust1BeingServed), \\
\quad (\{PerceptiveActions\}, envCustsWaitServ)\}), \\
(envCust1BeingServed, \{ \\
\quad (\{DisplayMenu\}, envCust1SelectMeal), \\
\quad (\{PerceptiveActions\}, envCust1BeingServed)\}), \\
(envCust2BeingServed, \{ \\
\quad (\{DisplayMenu\}, envCust2SelectMeal), \\
\quad (\{PerceptiveActions\}, envCust2BeingServed)\}), \\
(envCust1SelectMeal, \{ \\
\quad (\{ReceiveCustomersQuestions\}, envCust1\,WaitForInfo), \\
\quad (\{ReceiveOrder\}, envCust1PlacedOrder), \\
\quad (\{ReceiveCustomerGivesUp\}, envIdle), \\
\quad (\{PerceptiveActions\}, envCust1SelectMeal)\}),
\end{array}
$$

$(envCust2SelectMeal, \{$
$(\{ReceiveCustomersQuestions\}, envCust2WaitForInfo),$
$(\{ReceiveCustomerGivesUp\}, envIdle),$
$(\{ReceiveOrder\}, envCust2PlacedOrder),$
$(\{PerceptiveActions\}, envCust2SelectMeal)\}),$
$(envCust1WaitForInfo, \{$
$(\{AnswerCustomersQuestions\}, envCust1SelectMeal),$
$(\{InquiryDoubtsOfTheCustomer\}, envCust1SelectMeal),$
$(\{PerceptiveActions\}, envCust1WaitForInfo)\}),$
$(envCust2WaitForInfo, \{$
$(\{AnswerCustomersQuestions\}, envCust2SelectMeal),$
$(\{InquiryDoubtsOfTheCustomer\}, envCust2SelectMeal),$
$(\{PerceptiveActions\}, envCust2WaitForInfo)\}),$
$(envCust1PlacedOrder, \{$
$(\{ConfirmOrderWithPrice\}, envWaitForCust1Payment),$
$(\{PerceptiveActions\}, envCust1PlacedOrder)\}),$
$(envCust2PlacedOrder, \{$
$(\{ConfirmOrderWithPrice\}, envWaitForCust2Payment),$
$(\{PerceptiveActions\}, envCust2PlacedOrder)\}),$
$(envWaitForCust1Payment, \{$
$(\{ReceivePayment\}, envCust1HasPaid),$
$(\{ReceiveCustomerGivesUp\}, envIdle),$
$(\{PerceptiveActions\}, envWaitForCust1Payment)\}),$
$(envWaitForCust2Payment, \{$
$(\{ReceivePayment\}, envCust2HasPaid),$
$(\{ReceiveCustomerGivesUp\}, envIdle),$
$(\{PerceptiveActions\}, envWaitForCust2Payment)\}),$
$(envCust1HasPaid, \{$
$(\{ForwardOrderToKitchen\}, envCust1WaitForMeal),$
$(\{PerceptiveActions\}, envCust1HasPaid)\}),$
$(envCust2HasPaid, \{$
$(\{ForwardOrderToKitchen\}, envCust2WaitForMeal),$
$(\{PerceptiveActions\}, envCust2HasPaid)\}),$
$(envCust1WaitForMeal, \{$
$(\{CheckStateOfOrder\}, envCust1WaitForMeal),$
$(\{ReceivePreparedDish\}, envOrder1Ready),$
$(\{PerceptiveActions\}, envCust1WaitForMeal)\}),$

$(envCust2WaitForMeal, \{$
$\quad(\{CheckStateOfOrder\}, envCust2WaitForMeal),$
$\quad(\{ReceivePreparedDish\}, envOrder2Ready),$
$\quad(\{PerceptiveActions\}, envCust2WaitForMeal)\}),$
$(envOrder1Ready, \{$
$\quad(\{DeliverDishToCustomer\}, envCust1Served),$
$\quad(\{PerceptiveActions\}, envOrder1Ready)\}),$
$(envOrder2Ready, \{$
$\quad(\{DeliverDishToCustomer\}, envCust2Served),$
$\quad(\{PerceptiveActions\}, envOrder2Ready)\}),$
$(envCust1Served, \{$
$\quad(\{OfferAdditionalSeasonings\}, envIdle),$
$\quad(\{GoodAppetiteSeeYouSoon\}, envIdle),$
$\quad(\{PerceptiveActions\}, envCust1Served)\}),$
$(envCust2Served, \{$
$\quad(\{OfferAdditionalSeasonings\}, envIdle),$
$\quad(\{GoodAppetiteSeeYouSoon\}, envIdle),$
$\quad(\{PerceptiveActions\}, envCust2Served)\}),$
$(envOrder1BeingAnalyzed, \{$
$\quad(\{PrepareMeal\}, envOrder1BeingCooked)\}),$
$(envOrder1BeingCooked, \{$
$\quad(\{DeliverMeal\}, envIdle)\}),$
$(envShopOpen, \{$
$\quad(\{DefineQualityStandardsForServices\},$
$\qquad envDefinedQualityStandardsForServices)\}),$
$(envDefinedQualityStandardsForServices, \{$
$\quad(\{DefineQualityStandardsForMeals\},$
$\qquad envDefinedQualityStandardsForMeals)\}),$
$(envDefinedQualityStandardsForMeals, \{$
$\quad(\{DefineRewardsForWellPerformedWork\},$
$\qquad envDefinedRewardsForWellPerformedWork)\}),$
$(envDefinedRewardsForWellPerformedWork, \{$
$\quad(\{EvaluateCustomerService\},$
$\qquad envEvaluatedCustomerService)\}),$
$(envEvaluatedCustomerService, \{$
$\quad(\{EvaluateCustomerSatisfaction\},$
$\qquad envEvaluatedCustomerSatisfaction)\}),$
$(envEvaluatedCustomerSatisfaction, \{$
$\quad(\{ManageTheShop\}, envShopRunning)\}),$

$$
\begin{array}{l}
(envShopRunning, \{ \\
\quad (\{CloseTheShop\}, envShopClosed)\}), \\
(envShopClosed, \{ \\
\quad (\{OpenTheShop\}, envShopOpen)\})\}
\end{array}
$$

São apresentadas, a seguir, valorações associadas aos propósitos, valores, e visões dos agentes que trabalham na lanchonete.

$$
\begin{array}{l}
visionsOsvaldoServs01 == vision\langle\!|\ visions == \{\{\{atImproveSalary, atWealth\}\}\}\ |\!\rangle \\
visionsCarlosCook01 == vision\langle\!|\ visions == \{\{\{atImproveSalary, atStatus\}\}\}\ |\!\rangle \\
visionsMariManager01 == vision\langle\!|\ visions == \{\{\{atImproveProfits, atImproveSalary\}\}\}\ |\!\rangle
\end{array}
$$

$$
\begin{array}{l}
purposesOsvaldoServs01 == purpose\langle\!|\ purposes == \{\{\{atCareerDevelopment\}\}\}\ |\!\rangle \\
purposesCarlosCook01 == purpose\langle\!|\ purposes == \{\{\{atDevelopInterpersonalSkills\}\}\}\ |\!\rangle \\
purposesMariManager01 == purpose\langle\!|\ purposes == \\
\qquad \{\{\{atDevelopInterpersonalSkills, atCareerDevelopment\}\}\}\ |\!\rangle
\end{array}
$$

$$
\begin{array}{l}
valuesOsvaldoServs01 == value\langle\!|\ values == \{\{agpredicate(positive(atHonesty)), \\
\qquad agpredicate(positive(atEthicalPractice))\}\}\ |\!\rangle \\
valuesCarlosCook01 == value\langle\!|\ values == \{\{agpredicate(positive(atHonesty)), \\
\qquad agpredicate(positive(atEthicalPractice))\}\}\ |\!\rangle \\
valuesMariManager01 == value\langle\!|\ values == \{\{agpredicate(positive(atHonesty)), \\
\qquad agpredicate(positive(atReputation))\}\}\ |\!\rangle
\end{array}
$$

Quanto às diretrizes dos agentes, são definidas diretrizes específicas e consistentes para cada tipo, por exemplo, o cozinheiro tem entre seus valores o comportamento ético, os propósitos do atendente incluem o desenvolvimento de uma carreira, etc.

$$
\begin{array}{l}
guidingideaOsvaldoServs01 == guidingidea\langle\!|\ visions == \{visionsOsvaldoServs01\}, \\
\qquad\qquad purposes == \{purposesOsvaldoServs01\}, \\
\qquad\qquad values == \{valuesOsvaldoServs01\}\ |\!\rangle \\
guidingideaCarlosCook01 == guidingidea\langle\!|\ visions == \{visionsCarlosCook01\}, \\
\qquad\qquad purposes == \{purposesCarlosCook01\}, \\
\qquad\qquad values == \{valuesCarlosCook01\}\ |\!\rangle \\
guidingideaMariManager01 == guidingidea\langle\!|\ visions == \{visionsMariManager01\}, \\
\qquad\qquad purposes == \{purposesMariManager01\}, \\
\qquad\qquad values == \{valuesMariManager01\}\ |\!\rangle
\end{array}
$$

6.2.4 Papéis, Agentes, Modelos de Agentes

Os papéis organizacionais usados neste estudo são definidos conforme o próximo esquema. Nessa especificação para o estudo de caso, o papel inclui apenas as capacidades requeridas e as metas associadas ao papel.

$$RoleOrgSchema \mathrel{\widehat=} [roleGoals : \mathbb{P}\, Goal;\ skillsRequired : \mathbb{P}\, Action]$$

Finalmente, *AgentSchema* é o esquema simplificado correspondente ao definido no capítulo anterior para o agente *LearningOrgAgent*. O esquema *AgentSchema* inclui em uma única camada as variáveis que, por intermédio de operações de inclusão de esquemas, estão presentes no esquema *LearningOrgAgent*. Desta forma, o processo de criação de valorações é facilitado.

AgentSchema

$$\begin{array}{l}
name : AGENTNAME; \\
motivations : \mathbb{P}\, Motivation; \\
capableof : \mathbb{P}\, Action; \\
attributes : \mathbb{P}\, Attribute; \\
goals : \mathbb{P}\, Goal; \\
ownguidingideas : GuidingIdea; \\
store : \mathbb{P}\, Attribute; \\
personalMasteryCapabilities : \mathbb{P}\, Action \\
mentalModelsCapabilities : \mathbb{P}\, Action \\
systemsThinkingCapabilities : \mathbb{P}\, Action \\
learningTeamCapabilities : \mathbb{P}\, Action \\
buildingSharedVisionCapabilities : \mathbb{P}\, Action \\
mentalmodels : MentalModel; \\
roles : \mathbb{P}\, RoleOrg; \\
plans : \mathbb{P}\, Plan; \\
\hline
motivations \neq \varnothing \land capableof \neq \varnothing \land attributes \neq \varnothing;\ goals \neq \varnothing \land store \neq \varnothing; \\
personalMasteryCapabilities \neq \varnothing \land mentalModelsCapabilities \neq \varnothing \land \\
systemsThinkingCapabilities \neq \varnothing \land learningTeamCapabilities \neq \varnothing \land \\
buildingSharedVisionCapabilities \neq \varnothing; \\
(personalMasteryCapabilities \cup mentalModelsCapabilities \cup systemsThinkingCapabilities \cup \\
learningTeamCapabilities \cup buildingSharedVisionCapabilities) \subset capableof;
\end{array}$$

Para este estudo de caso foi definido um tipo específico para o modelo associado ao agente do tipo *AgentSchema*. Tal modelo no capítulo 5, é definido usando-se uma abreviação, ou seja, naquele caso o modelo é idêntico ao tipo *LearningOrgAgent*.

AgentModelSchema

$$
\begin{array}{l}
name : AGENTNAME; \\
motivations : \mathbb{P}\, Motivation; \\
capableof : \mathbb{P}\, Action; \\
attributes : \mathbb{P}\, Attribute; \\
goals : \mathbb{P}\, Goal; \\
ownguidingideas : GuidingIdea; \\
store : \mathbb{P}\, Attribute; \\
personalMasteryCapabilities : \mathbb{P}\, Action \\
mentalModelsCapabilities : \mathbb{P}\, Action \\
systemsThinkingCapabilities : \mathbb{P}\, Action \\
learningTeamCapabilities : \mathbb{P}\, Action \\
buildingSharedVisionCapabilities : \mathbb{P}\, Action \\
roles : \mathbb{P}\, RoleOrg; \\
plans : \mathbb{P}\, Plan; \\
\hline
motivations \neq \varnothing \land \\
\quad capableof \neq \varnothing \land \\
\quad attributes \neq \varnothing; \\
goals \neq \varnothing \land \\
\quad store \neq \varnothing; \\
personalMasteryCapabilities \neq \varnothing; \\
mentalModelsCapabilities \neq \varnothing; \\
systemsThinkingCapabilities \neq \varnothing; \\
learningTeamCapabilities \neq \varnothing; \\
buildingSharedVisionCapabilities \neq \varnothing; \\
(personalMasteryCapabilities \cup \\
\quad mentalModelsCapabilities \cup \\
\quad systemsThinkingCapabilities \cup \\
\quad learningTeamCapabilities \cup \\
\quad buildingSharedVisionCapabilities) \subset capableof;
\end{array}
$$

As percepções desse agente são influenciadas por motivações, modelos mentais, diretrizes, papéis e metas. As variáveis *personalmasteryS* e *systemsthinkingskillS* foram introduzidas nesse esquema para que, posteriormente, seja possível fazer referência aos respectivos esquemas e às variáveis neles declaradas.

AgentPerceptionSchema

$$
\begin{array}{l}
AgentSchema; \\
peractions : Actions; \\
canperceive : Environment \rightarrow Actions \rightarrow \\
\qquad Environment; \\
personalmastery : PersonalMastery; \\
personalmasteryS : PersonalMasterySchema; \\
systemsthinkingskill : SystemsThinking; \\
systemsthinkingskillS : SystemsThinkingSchema; \\
agperceives : \mathbb{P}\, Motivation \rightarrow MentalModel \rightarrow \\
\qquad GuidingIdea \rightarrow \mathbb{P}\, RoleOrg \rightarrow \\
\qquad\quad \mathbb{P}\, Goal \rightarrow View \rightarrow View \\
\hline
peractions \cap capableof = peractions; \\
\mathrm{dom}\ agperceives = \{motivations\};
\end{array}
$$

As ações desse agente são função de motivações, modelos mentais, diretrizes, planos e metas.

AgentActionSchema

$$
\begin{array}{l}
AgentSchema; \\
agentact : \mathbb{P}\, Motivation \rightarrow MentalModel \rightarrow GuidingIdea \rightarrow \\
\qquad \mathbb{P}\, Goal \rightarrow \mathbb{P}\, Plan \rightarrow View \rightarrow Environment \rightarrow Actions; \\
\hline
\mathrm{dom}\ agentact = \{motivations\};
\end{array}
$$

Analogamente ao que foi apresentado no capítulo anterior, o estado deste agente inclui sua percepção e ação.

AgentStateSchema

$$
\begin{array}{l}
AgentActionSchema; \\
AgentPerceptionSchema; \\
peractions : Actions; \\
env : Environment; \\
posspercepts, actualpercepts : View; \\
willdo : Actions; \\
\hline
actualpercepts \subseteq posspercepts \\
posspercepts = canperceive\ env\ peractions \\
actualpercepts = agperceives\ motivations\ mentalmodels \\
\qquad ownguidingideas\ roles \\
\qquad goals\ posspercepts \\
peractions = \varnothing \Rightarrow posspercepts = \varnothing \\
willdo = agentact\ motivations\ mentalmodels \\
\qquad personalmasteryS.guidingidea\ goals \\
\qquad (systemsthinkingskillS.producePlans\ actualpercepts) \\
\qquad actualpercepts\ env; \\
willdo \cap capableof = willdo;
\end{array}
$$

Com o uso do ZETA, ocorreram erros na inclusão da variante Ξ diretamente no esquema *DeltaAgentStateSchema*. Portanto, houve a necessidade de se definir os dois esquemas auxiliares abaixo e incluí-los em *DeltaAgentStateSchema*.

XiAgentActionSchema

$$
\begin{array}{l}
\Xi AgentActionSchema;
\end{array}
$$

XiAgentPerceptionSchema

$$
\begin{array}{l}
\Xi AgentPerceptionSchema;
\end{array}
$$

DeltaAgentStateSchema

$$
\begin{array}{l}
AgentStateSchema; \\
AgentStateSchema'; \\
XiAgentActionSchema; \\
XiAgentPerceptionSchema;
\end{array}
$$

A seguir, são apresentadas definições relativas às interações desse agente com o meio ambiente. De modo análogo ao apresentado no capítulo anterior, as interações são definidas como operações sobre o esquema que define o estado do agente (*AgentStateSchema*). De acordo com essa operação, o desenvolvimento das disciplinas de domínio pessoal e pensamento sistêmico permanece inalterado. Conforme definido no SMART, os novos perceptos potenciais (*posspercepts'*) são obtidos por intermédio da aplicação da função *canperceive* sobre o novo estado do ambiente (*env'*). Os perceptos efetivamente captados pelo agente (*actualpercepts'*) resultam de mapeamentos da função (*agperceives*) sobre as novas motivações, modelos mentais, diretrizes, papéis, metas e os novos perceptos potenciais. Analogamente, as novas ações do agente (*willdo'*) resultam de mapeamentos da função (*agentact*) sobre as novas motivações, modelos mentais, diretrizes, metas, planos resultantes do pensamento sistêmico e dos novos perceptos efetivos. As funções *agperceives* e *agentact* definidas abaixo correspondem às funções *learnorgperceives* e *learnorgact* definidas no capítulo anterior, respectivamente, nos esquemas *LearningOrgAgentPerception* e *LearningOrgAgentAction*. A estratégia de especificação adotada neste capítulo não emprega múltiplas camadas de definições de agentes, tornando possível a opção por denominações mais simples para essas funções.

AgentInteractsSchema

$$
\begin{array}{l}
\Delta AgentStateSchema; \\
\hline
personalmastery' = personalmastery; \\
personalmasteryS' = personalmasteryS; \\
systemsthinkingskill' = systemsthinkingskill; \\
systemsthinkingskillS' = systemsthinkingskillS; \\
posspercepts' = canperceive\ env'\ peractions; \\
actualpercepts' = agperceives\ motivations'\ mentalmodels' \\
\qquad ownguidingideas'\ roles'\ goals'\ posspercepts' \\
willdo' = agentact\ motivations'\ mentalmodels' \\
\qquad personalmasteryS'.guidingidea\ goals' \\
\qquad (systemsthinkingskillS'.producePlans\ actualpercepts') \\
\qquad actualpercepts'\ env';
\end{array}
$$

A operação *AgentInteractsInputSchema* definida abaixo estende *AgentInteractsSchema* com a

inclusão de variáveis de entrada que definem um estado do ambiente e ações a serem realizadas. De acordo com esse esquema, a aplicação da função *effectinteraction* sobre as variáveis de entrada deve produzir um novo estado no ambiente. Mais adiante, na seção 6.4, ações padronizadas dos agentes (*UniformAction*) serão definidas em função desse esquema.

AgentInteractsInputSchema

$AgentInteractsSchema$;
$env? : Environment$;
$willdo? : Actions$;

$env' = effectinteraction\ env?\ willdo?$;

6.2.5 Equipe e Organização

A seguir, são definidos os esquemas referentes à equipe e organização aprendiz.

Para a definição da equipe são declaradas diretrizes (*teamguidingideas*), planos (*commonplans*) e metas (*commongoals*) comuns aos seus agentes membros e as funções de desenvolvimento de tais planos (*developcommonplans*) e metas (*developCommonGoals*).

LTeamSchema

$commongoals : \mathbb{P}\ Goal$
$commonplans : \mathbb{P}\ Plan$
$developCommonGoals : \mathbb{P}\ Agent \rightarrow \mathbb{P}\ Goal$
$developcommonplans : \mathbb{P}\ Agent \rightarrow \mathbb{P}\ Plan$
$teamguidingideas : GuidingIdea$
$learningmembers : \mathbb{P}\ LearningOrgAgent$

$commongoals \neq \varnothing$
$\mathsf{dom}\ developCommonGoals = \{learningmembers\}$
$\mathsf{dom}\ developcommonplans = \{learningmembers\}$
$\mathsf{ran}\ developCommonGoals = \{commongoals\}$
$\mathsf{ran}\ developcommonplans = \{commonplans\}$

O esquema *LOrgSchema* especifica a Organização Aprendiz neste estudo de caso. Nesse esquema são incluídas apenas as variáveis que fazem referência às várias equipes que fazem parte

da organização, e à visão compartilhada organizacional. Para este estudo de caso não foram consideradas normas, estrutura, ou outras características que obrigassem a inclusão de definições associadas a uma organização formal (*FormalOrg*), como efetuado no capítulo anterior na definição da Organização Aprendiz (*LearningOrg*).

LOrgSchema

$learningteams : \mathbb{P}\, LTeam$
$sharedvision : SharedVision$

6.2.6 Valorações dos Agentes, Modelos e Estados

A valoração referente ao agente *agentOsvaldo* corresponde a um modelo de um atendente da lanchonete. Possui, portanto, objetivos e capacidades específicas associadas à atividade de atendimento.

$$
\begin{aligned}
&agentOsvaldo == agent\langle\!| \; name == OsvaldoServs,\\
&\quad motivations == motivationServices,\\
&\quad capableof == capabilitiesServices,\\
&\quad ownguidingideas == guidingideaOsvaldoServs01,\\
&\quad goals == agentgoalsServices,\\
&\quad attributes == \{\{atom1, atom2\}\},\\
&\quad personalMasteryCapabilities == \{DevelopGuidingIdeasActions,\\
&\qquad EnhanceRealityVisionsActions,\\
&\qquad ClarifyPersonalVisionsActions\},\\
&\quad mentalModelsCapabilities == \{MentalModelsActions\},\\
&\quad systemsThinkingCapabilities == \{SystemsThinkingActions\},\\
&\quad learningTeamCapabilities == \{TeamLearningActions\},\\
&\quad buildingSharedVisionCapabilities == \{SharedVisionActions\},\\
&\quad mentalmodels == mentalmodelService,\\
&\quad roles == \{roleAttendant\}, plans == \{planService01\}, store == \{\{atom1, atom2\}\} \; |\!\rangle
\end{aligned}
$$

A valoração referente ao agente *agentCarlos* corresponde a um modelo de um cozinheiro da lanchonete. Analogamente, a valoração referente ao agente *agentMari* corresponde a um modelo de um gerente da lanchonete.

$$
\begin{aligned}
&agentCarlos == agent\langle\!| \; name == CarlosCook,\\
&\quad motivations == motivationCook,\\
&\quad capableof == capabilitiesCook01,\\
&\quad ownguidingideas == guidingideaCarlosCook01,\\
&\quad goals == agentgoalsCook,\\
&\quad attributes == \{\{atom1, atom2\}\},\\
&\quad personalMasteryCapabilities == \{DevelopGuidingIdeasActions,\\
&\qquad EnhanceRealityVisionsActions,\\
&\qquad ClarifyPersonalVisionsActions\},\\
&\quad mentalModelsCapabilities == \{MentalModelsActions\},\\
&\quad systemsThinkingCapabilities == \{SystemsThinkingActions\},\\
&\quad learningTeamCapabilities == \{TeamLearningActions\},\\
&\quad buildingSharedVisionCapabilities == \{SharedVisionActions\},\\
&\quad mentalmodels == mentalmodelCook,\\
&\quad roles == \{roleCook\}, plans == \{planCook01\}, store == \{\{atom1, atom2\}\} \; |\!\rangle
\end{aligned}
$$

$$
\begin{aligned}
&agentMari == agent\langle\!| \; name == MariManager,\\
&\quad motivations == motivationManager,\\
&\quad capableof == capabilitiesManager01,\\
&\quad ownguidingideas == guidingideaMariManager01,\\
&\quad goals == agentgoalsManager,\\
&\quad attributes == \{\{atom1, atom2\}\},\\
&\quad personalMasteryCapabilities == \{DevelopGuidingIdeasActions,\\
&\qquad EnhanceRealityVisionsActions,\\
&\qquad ClarifyPersonalVisionsActions\},\\
&\quad mentalModelsCapabilities == \{MentalModelsActions\},\\
&\quad systemsThinkingCapabilities == \{SystemsThinkingActions\},\\
&\quad learningTeamCapabilities == \{TeamLearningActions\},\\
&\quad buildingSharedVisionCapabilities == \{SharedVisionActions\},\\
&\quad mentalmodels == mentalmodelManager,\\
&\quad roles == \{roleManager\}, plans == \{planManagement01\},\\
&\quad store == \{\{atom1, atom2\}\} \; |\!\rangle
\end{aligned}
$$

As valorações referentes aos modelos dos agentes, neste estudo de caso, são similares às instâncias dos agentes. A exceção é a ausência de modelos mentais nos modelos. Como conseqüência, um determinado agente não pode ter acesso aos modelos mentais de um outro agente a partir de seu próprio modelo daquele agente. Desta forma, não ocorre circularidade nas definições de

agentes, modelos e modelos mentais: caso o agente A tivesse um modelo do agente B em seus modelos mentais e B tivesse um modelo de A em seus modelos, então A poderia ter acesso a um modelo de si próprio a partir dos modelos de B.

$$
\begin{aligned}
&agentOsvaldomodel == agentmodel\langle\!| \; name == OsvaldoServs,\\
&\quad motivations == motivationServices,\\
&\quad capableof == capabilitiesServices,\\
&\quad ownguidingideas == guidingideaOsvaldoServs01,\\
&\quad goals == agentgoalsServices,\\
&\quad attributes == \{\{atom1, atom2\}\},\\
&\quad personalMasteryCapabilities == \{DevelopGuidingIdeasActions,\\
&\qquad EnhanceRealityVisionsActions,\\
&\qquad ClarifyPersonalVisionsActions\},\\
&\quad mentalModelsCapabilities == \{MentalModelsActions\},\\
&\quad systemsThinkingCapabilities == \{SystemsThinkingActions\},\\
&\quad learningTeamCapabilities == \{TeamLearningActions\},\\
&\quad buildingSharedVisionCapabilities == \{SharedVisionActions\},\\
&\quad roles == \{roleAttendant\}, plans == \{planService01\}, store == \{\{atom1, atom2\}\} \; |\!\rangle
\end{aligned}
$$

$$
\begin{aligned}
&agentCarlosmodel == agentmodel\langle\!| \; name == CarlosCook,\\
&\quad motivations == motivationCook,\\
&\quad capableof == capabilitiesCook01,\\
&\quad ownguidingideas == guidingideaCarlosCook01,\\
&\quad goals == agentgoalsCook,\\
&\quad attributes == \{\{atom1, atom2\}\},\\
&\quad personalMasteryCapabilities == \{DevelopGuidingIdeasActions,\\
&\qquad EnhanceRealityVisionsActions,\\
&\qquad ClarifyPersonalVisionsActions\},\\
&\quad mentalModelsCapabilities == \{MentalModelsActions\},\\
&\quad systemsThinkingCapabilities == \{SystemsThinkingActions\},\\
&\quad learningTeamCapabilities == \{TeamLearningActions\},\\
&\quad buildingSharedVisionCapabilities == \{SharedVisionActions\},\\
&\quad roles == \{roleCook\}, plans == \{planCook01\}, store == \{\{atom1, atom2\}\} \; |\!\rangle
\end{aligned}
$$

$agentMarimodel == agentmodel\langle\!|\ name == MariManager,$
$motivations == motivationManager,$
$capableof == capabilitiesManager01,$
$ownguidingideas == guidingideaMariManager01,$
$goals == agentgoalsManager,$
$attributes == \{\{atom1, atom2\}\},$
$personalMasteryCapabilities == \{DevelopGuidingIdeasActions,$
$EnhanceRealityVisionsActions,$
$ClarifyPersonalVisionsActions\},$
$mentalModelsCapabilities == \{MentalModelsActions\},$
$systemsThinkingCapabilities == \{SystemsThinkingActions\},$
$learningTeamCapabilities == \{TeamLearningActions\},$
$buildingSharedVisionCapabilities == \{SharedVisionActions\},$
$roles == \{roleManager\},$
$plans == \{planManagement01\},$
$store == \{\{atom1, atom2\}\}\ |\!\rangle$

A seguir, são apresentadas as valorações criadas para os modelos mentais dos agentes que compõem a organização: atendente, gerente e cozinheiro.

$mentalmodelService == mentalmodel\langle\!|$
$developReflectionSkillsActions ==$
$\{MentalModelsActions, ReflectionActions\},$
$developAdvocacySkillsActions ==$
$\{MentalModelsActions, ArgumentationActions\},$
$intraPersonalMentalModelPrinciples ==$
$\{\{MentalModelsPrinciples\}\},$
$models == \{agentCarlosmodel, agentMarimodel\},$
$modelentities == \{agentCarlosmodel, agentMarimodel\},$
$modelagents == \{agentCarlosmodel, agentMarimodel\},$
$modellearningorgagents == \{agentCarlosmodel, agentMarimodel\},$
$modelbelmodel == \{(agentOsvaldomodel, agentOsvaldomodel)\},$
$modelLOrgAgInts == \{$
$(agentOsvaldomodel, agentCarlosmodel),$
$(agentOsvaldomodel, agentMarimodel)\},$

$$
\begin{array}{l}
\quad modelTrustLOrgAgInts == \{ \\
\qquad (agentOsvaldomodel, agentCarlosmodel), \\
\qquad (agentOsvaldomodel, agentMarimodel)\}, \\
\quad modelNonTrustLOrgAgInts == \{\}, \\
\quad agentbeliefs == \{agpredicate(positive(atom1))\} \rangle
\end{array}
$$

$$
\begin{array}{l}
mentalmodelCook == mentalmodel\langle \\
\quad developReflectionSkillsActions == \\
\qquad \{MentalModelsActions, ReflectionActions\}, \\
\quad developAdvocacySkillsActions == \\
\qquad \{MentalModelsActions, ArgumentationActions\}, \\
\quad intraPersonalMentalModelPrinciples == \{\{MentalModelsPrinciples\}\}, \\
\quad models == \{agentOsvaldomodel, agentMarimodel\}, \\
\quad modelentities == \{agentOsvaldomodel, agentMarimodel\}, \\
\quad modelagents == \{agentOsvaldomodel, agentMarimodel\}, \\
\quad modellearningorgagents == \{agentOsvaldomodel, agentMarimodel\}, \\
\quad modelbelmodel == \{(agentCarlosmodel, agentCarlosmodel)\}, \\
\quad modelLOrgAgInts == \{ \\
\qquad (agentCarlosmodel, agentOsvaldomodel), \\
\qquad (agentCarlosmodel, agentMarimodel)\}, \\
\quad modelTrustLOrgAgInts == \{ \\
\qquad (agentCarlosmodel, agentOsvaldomodel), \\
\qquad (agentCarlosmodel, agentMarimodel)\}, \\
\quad modelNonTrustLOrgAgInts == \{\}, \\
\quad agentbeliefs == \{agpredicate(positive(atom1))\} \rangle
\end{array}
$$

$$
\begin{array}{l}
mentalmodelManager == mentalmodel\langle \\
\quad developReflectionSkillsActions == \\
\qquad \{MentalModelsActions, ReflectionActions\}, \\
\quad developAdvocacySkillsActions == \\
\qquad \{MentalModelsActions, ArgumentationActions\}, \\
\quad intraPersonalMentalModelPrinciples == \{\{MentalModelsPrinciples\}\}, \\
\quad models == \{agentCarlosmodel, agentOsvaldomodel\}, \\
\quad modelentities == \{agentCarlosmodel, agentOsvaldomodel\},
\end{array}
$$

$$
\begin{array}{l}
\quad modelagents == \{agentCarlosmodel, agentOsvaldomodel\}, \\
\quad modellearningorgagents == \{agentCarlosmodel, agentOsvaldomodel\}, \\
\quad modelbelmodel == \{(agentMarimodel, agentMarimodel)\}, \\
\quad modelLOrgAgInts == \{ \\
\qquad (agentMarimodel, agentCarlosmodel), \\
\qquad (agentMarimodel, agentOsvaldomodel)\}, \\
\quad modelTrustLOrgAgInts == \{ \\
\qquad (agentMarimodel, agentCarlosmodel), \\
\qquad (agentMarimodel, agentOsvaldomodel)\}, \\
\quad modelNonTrustLOrgAgInts == \{\}, \\
\quad agentbeliefs == \{agpredicate(positive(atom1))\} \rangle
\end{array}
$$

Nesse ponto, são introduzidas as valorações necessárias para a definição de domínio pessoal.

Inicialmente, são definidas valorações para a visão pessoal de cada agente.

$$
\begin{array}{l}
personalvisionServices01 == personalvisiongen\langle \\
\quad personalvision == personalvisiontemplateServices01 \rangle \\
personalvisionCook01 == personalvisiongen\langle \\
\quad personalvision == personalvisiontemplateCook01 \rangle \\
personalvisionManager01 == personalvisiongen\langle \\
\quad personalvision == personalvisiontemplateManager01 \rangle
\end{array}
$$

São definidas, a seguir, as valorações para as funções de visão pessoal de cada agente.

$$
\begin{array}{l}
personalvisiontemplateServices01 == \\
\{(motivationServices, \\
\qquad \{(mentalmodelService, \\
\qquad\quad \{(guidingideaOsvaldoServs01, \\
\qquad\qquad \{\{\{atCareerDevelopment\}\}\}) \\
\qquad\quad \}) \\
\qquad \})\}
\end{array}
$$

$$
\begin{array}{l}
personalvisiontemplateCook01 == \\
\{(motivationCook, \\
\quad \{(mentalmodelCook, \\
\quad\quad \{(guidingideaCarlosCook01, \\
\quad\quad\quad \{\{\{atCareerDevelopment\}\}\}) \\
\quad\quad \}) \\
\quad \})\}
\end{array}
$$

$$
\begin{array}{l}
personalvisiontemplateManager01 == \\
\{(motivationManager, \\
\quad \{(mentalmodelManager, \\
\quad\quad \{(guidingideaMariManager01, \\
\quad\quad\quad \{\{\{atCareerDevelopment\}\}\}) \\
\quad\quad \}) \\
\quad \})\}
\end{array}
$$

São declaradas, então, as valorações para o domínio pessoal de cada agente.

$$
\begin{array}{l}
personalmasteryServices01 == \\
\quad personalmasterygen\langle\!| \ developguidingideasactions == \\
\quad \{DevelopGuidingIdeasActions\}, \\
\quad enhancerealityvisionsactions == \{EnhanceRealityVisionsActions\}, \\
\quad clarifypersonalvisionsactions == \{ClarifyPersonalVisionsActions\}, \\
\quad personalMasteryPrinciples == \{\{PersonalMasteryPrinciples\}\}, \\
\quad personalvisions == \{personalvisionServices01\}, \\
\quad guidingidea == guidingideaOsvaldoServs01, \\
\quad developguidingideas == developguidingideastemplateServices01, \\
\quad enhancerealityvisions == enhancerealityvisionstemplateServices01, \\
\quad clarifypersonalvisions == clarifypersonalvisionstemplateServices01, \\
\quad creativetension == creativetensiontemplateServices01 \ |\!\rangle
\end{array}
$$

$personalmasteryCook01 ==$
$personalmasterygen\langle\!|\ developguidingideasactions ==$
$\{DevelopGuidingIdeasActions\},$
$enhancerealityvisionsactions == \{EnhanceRealityVisionsActions\},$
$clarifypersonalvisionsactions == \{ClarifyPersonalVisionsActions\},$
$personalMasteryPrinciples == \{\{PersonalMasteryPrinciples\}\},$
$personalvisions == \{personalvisionCook01\},$
$guidingidea == guidingideaCarlosCook01,$
$developguidingideas == developguidingideastemplateCook01,$
$enhancerealityvisions == enhancerealityvisionstemplateCook01,$
$clarifypersonalvisions == clarifypersonalvisionstemplateCook01,$
$creativetension == creativetensiontemplateCook01\ |\!\rangle$

$personalmasteryManager01 ==$
$personalmasterygen\langle\!|\ developguidingideasactions ==$
$\{DevelopGuidingIdeasActions\},$
$enhancerealityvisionsactions == \{EnhanceRealityVisionsActions\},$
$clarifypersonalvisionsactions == \{ClarifyPersonalVisionsActions\},$
$personalMasteryPrinciples == \{\{PersonalMasteryPrinciples\}\},$
$personalvisions == \{personalvisionManager01\},$
$guidingidea == guidingideaMariManager01,$
$developguidingideas == developguidingideastemplateManager01,$
$enhancerealityvisions == enhancerealityvisionstemplateManager01,$
$clarifypersonalvisions == clarifypersonalvisionstemplateManager01,$
$creativetension == creativetensiontemplateManager01\ |\!\rangle$

As definições de valorações para a visão da realidade usadas neste estudo são simplificadas, apenas indicando que alguns átomos devem compor tal realidade.

$realityVisionServices01 == \{\{\{atom1, atom2\}\}\}$
$realityVisionCook01 == \{\{\{atom1, atom2\}\}\}$
$realityVisionManager01 == \{\{\{atom1, atom2\}\}\}$

As definições referentes ao desenvolvimento de diretrizes, tensão criativa e visão clara da realidade usadas acima nas valorações de domínio pessoal dos agentes, são apresentadas a seguir.

$$
\begin{array}{l}
developguidingideastemplateServices01 == \\
\{((motivationServices, \\
\quad \{DevelopGuidingIdeasActions\}), guidingideaOsvaldoServs01)\}
\end{array}
$$

$$
\begin{array}{l}
developguidingideastemplateCook01 == \\
\{((motivationCook, \\
\quad \{DevelopGuidingIdeasActions\}), guidingideaCarlosCook01)\}
\end{array}
$$

$$
\begin{array}{l}
developguidingideastemplateManager01 == \\
\{((motivationManager, \\
\quad \{DevelopGuidingIdeasActions\}), guidingideaMariManager01)\}
\end{array}
$$

$$
\begin{array}{l}
enhancerealityvisionstemplateServices01 == \\
\{((motivationServices, \\
\quad \{EnhanceRealityVisionsActions\}), \{realityVisionServices01\})\}
\end{array}
$$

$$
\begin{array}{l}
enhancerealityvisionstemplateCook01 == \\
\{((motivationCook, \\
\quad \{EnhanceRealityVisionsActions\}), \{realityVisionCook01\})\}
\end{array}
$$

$$
\begin{array}{l}
enhancerealityvisionstemplateManager01 == \\
\{((motivationManager, \\
\quad \{EnhanceRealityVisionsActions\}), \{realityVisionManager01\})\}
\end{array}
$$

$$
\begin{array}{l}
clarifypersonalvisionstemplateServices01 == \\
\{((motivationServices, \\
\quad \{ClarifyPersonalVisionsActions\}), \{personalvisionServices01\})\}
\end{array}
$$

$$
\begin{array}{l}
clarifypersonalvisionstemplateCook01 == \\
\{((motivationCook, \\
\quad \{ClarifyPersonalVisionsActions\}), \{personalvisionCook01\})\}
\end{array}
$$

$$
\begin{array}{l}
clarifypersonalvisionstemplateManager01 == \\
\{((motivationManager, \\
\quad \{ClarifyPersonalVisionsActions\}), \{personalvisionManager01\})\}
\end{array}
$$

$creativetensiontemplateServices01 ==$
$\{((realityVisionServices01,$
$\quad \{personalvisionServices01\}), \{\{\{\{atCareerDevelopment\}\}\}\})\}$

$creativetensiontemplateCook01 ==$
$\{((realityVisionCook01,$
$\quad \{personalvisionCook01\}), \{\{\{\{atCareerDevelopment\}\}\}\})\}$

$creativetensiontemplateManager01 ==$
$\{((realityVisionManager01,$
$\quad \{personalvisionManager01\}), \{\{\{\{atCareerDevelopment\}\}\}\})\}$

As valorações de pensamento sistêmico associadas ao atendente, cozinheiro e gerente são definidas em um nível de abstração menos detalhado que associa percepções a planos. Neste estudo de caso, foi adotada a simplificação de que para todo estado do ambiente é mapeado o mesmo conjunto de planos. No entanto, claramente podem ser definidos diferentes estados e planos que satisfaçam as necessidades de outros tipos de exemplos e estudos.

$systemsthinkingServices01 == systemsthinkinggen\langle$
$\quad producePlans == producePlanstemplateServices01 \rangle$
$systemsthinkingCook01 == systemsthinkinggen\langle$
$\quad producePlans == producePlanstemplateCook01 \rangle$
$systemsthinkingManagement01 == systemsthinkinggen\langle$
$\quad producePlans == producePlanstemplateManagement01 \rangle$

$producePlanstemplateServices01 ==$
$\{(envIdle, \{planService01\}),$
$\quad (envCust1WaitServ, \{planService01\}),$
$\quad (envCust2WaitServ, \{planService01\}),$
$\quad (envCustsWaitServ, \{planService01\}),$
$\quad (envCust1BeingServed, \{planService01\}),$
$\quad (envCust2BeingServed, \{planService01\}),$

$(envCust1SelectMeal, \{planService01\}),$
$(envCust2SelectMeal, \{planService01\}),$
$(envCust1WaitForInfo, \{planService01\}),$
$(envCust2WaitForInfo, \{planService01\}),$
$(envCust1PlacedOrder, \{planService01\}),$
$(envCust2PlacedOrder, \{planService01\}),$
$(envCust1ConfirmingOrder, \{planService01\}),$
$(envCust2ConfirmingOrder, \{planService01\}),$
$(envCust1ConfirmedOrder, \{planService01\}),$
$(envCust2ConfirmedOrder, \{planService01\}),$
$(envWaitForCust1Payment, \{planService01\}),$
$(envWaitForCust2Payment, \{planService01\}),$
$(envCust1HasPaid, \{planService01\}),$
$(envCust2HasPaid, \{planService01\}),$
$(envCust1WaitForMeal, \{planService01\}),$
$(envCust2WaitForMeal, \{planService01\}),$
$(envOrder1Ready, \{planService01\}),$
$(envOrder2Ready, \{planService01\}),$
$(envCust1Served, \{planService01\}),$
$(envCust2Served, \{planService01\}),$
$(envOrder1BeingAnalyzed, \{planService01\}),$
$(envOrder1BeingCooked, \{planService01\}),$
$(envShopOpen, \{planService01\}),$
$(envShopClosed, \{planService01\})\}$

$producePlanstemplateCook01 ==$
$\{(envIdle, \{planCook01\}),$
$(envOrder1BeingAnalyzed, \{planCook01\}),$
$(envOrder1BeingCooked, \{planCook01\}),$
$(envShopOpen, \{planCook01\}),$
$(envShopClosed, \{planCook01\})\}$

$$
\begin{array}{l}
produceEPlanstemplateManagement01 == \\
\{(envIdle, \{planManagement01\}), \\
\quad (envOrder1BeingAnalyzed, \{planManagement01\}), \\
\quad (envOrder1BeingCooked, \{planManagement01\}), \\
\quad (envShopOpen, \{planManagement01\}), \\
\quad (envDefinedQualityStandardsForServices, \\
\qquad \{planManagement01\}), \\
\quad (envDefinedQualityStandardsForMeals, \{planManagement01\}), \\
\quad (envDefinedRewardsForWellPerformedWork, \\
\qquad \{planManagement01\}), \\
\quad (envEvaluatedCustomerService, \{planManagement01\}), \\
\quad (envEvaluatedCustomerSatisfaction, \{planManagement01\}), \\
\quad (envShopRunning, \{planManagement01\}), \\
\quad (envShopClosed, \{planManagement01\})\}
\end{array}
$$

Uma vez declaradas as valorações referentes ao estado do agente, seu domínio pessoal e pensamento sistêmico, é possível, a partir de uma dada valoração obter-se um esquema específico.

Estas funções auxiliares permitem tal operação.

$$
\begin{array}{l}
getAgentStateSchemaFromBinding == agentstate^{\sim} \\
getPersonalMasterySchemaFromBinding == personalmasterygen^{\sim} \\
getSystemsThinkingSchemaFromBinding == systemsthinkinggen^{\sim}
\end{array}
$$

Em seguida, são definidos os ambientes, capacidades e a instância de estado associados ao atendente. Os ambientes relacionados ao atendente podem incluir: espera por um novo cliente, cliente selecionando um lanche, cliente aguardando lanche, pedido pronto para ser entregue ao cliente, lanche pago, entre outros.

$$
\begin{array}{l}
env0Service == envIdle \\
action0Service == \{ReceiveNewCustomer\} \\
env1Service == envCust1WaitServ \\
action1Service == \{GreetTheCustomer\} \\
env2Service == envCust1BeingServed \\
action2Service == \{DisplayMenu\}
\end{array}
$$

$$
\begin{aligned}
&env3Service == envCust1SelectMeal \\
&action3Service == \{ReceiveOrder\} \\
&env4Service == envCust1PlacedOrder \\
&action4Service == \{ConfirmOrderWithPrice\} \\
&env5Service == envWaitForCust1Payment \\
&action5Service == \{ReceivePayment\} \\
&env6Service == envCust1HasPaid \\
&action6Service == \{ForwardOrderToKitchen\} \\
&env7Service == envCust1WaitForMeal \\
&action7Service == \{ReceivePreparedDish\} \\
&env8Service == envOrder1Ready \\
&action8Service == \{DeliverDishToCustomer\} \\
&env9Service == envCust1Served \\
&action9Service == \{GoodAppetiteSeeYouSoon\}
\end{aligned}
$$

São definidas também as capacidades do agente atendente, que incluem capacidades associadas ao desenvolvimento das disciplinas de Senge e capacidades específicas necessárias para o desempenho da função de atendente.

$$
\begin{aligned}
capabilitiesServices == \{&DevelopGuidingIdeasActions, \\
&EnhanceRealityVisionsActions, \\
&ClarifyPersonalVisionsActions, MentalModelsActions, \\
&TeamLearningActions, SystemsThinkingActions, \\
&SharedVisionActions, PerceptiveActions, \\
&ReceiveNewCustomer, GreetTheCustomer, \\
&DisplayMenu, ReceiveCustomersQuestions, \\
&AnswerCustomersQuestions, InquiryDoubtsOfTheCustomer, \\
&ReceiveOrder, SuggestSideDishes, InformInvalidOrder, \\
&ReceiveOrderConfirmation, ReceiveCustomerGivesUp, \\
&ConfirmOrder, InformAmountOfTheBill, \\
&ConfirmOrderWithPrice, ForwardOrderToKitchen, \\
&CheckStateOfOrder, ReceivePayment, \\
&CheckPayment, ComputeChange, ReturnChange, \\
&ReceivePreparedDish, DeliverDishToCustomer, \\
&OfferAdditionalSeasonings, GoodAppetiteSeeYouSoon\}
\end{aligned}
$$

Os esquemas declarados a seguir são obtidos a partir de valorações. Desta forma, são obtidos esquemas referentes ao domínio pessoal e ao pensamento sistêmico do agente.

$personalMasterySchemaServices01 ==$
$\quad getPersonalMasterySchemaFromBinding \ \ personalmasteryServices01$
$systemsThinkingSchemaServices01 ==$
$\quad getSystemsThinkingSchemaFromBinding \ \ systemsthinkingServices01$

Finalmente, a valoração associada ao agente de atendimento é declarada a seguir.

$agentOsvaldostate == agentstate(\!|\ name == OsvaldoServs,$
$\quad motivations == motivationServices,$
$\quad capableof == capabilitiesServices,$
$\quad ownguidingideas == guidingideaOsvaldoServs01,$
$\quad goals == agentgoalsServices,$
$\quad attributes == \{\{atom1, atom2\}\},$
$\quad personalMasteryCapabilities ==$
$\quad\quad \{DevelopGuidingIdeasActions,$
$\quad\quad EnhanceRealityVisionsActions,$
$\quad\quad ClarifyPersonalVisionsActions\},$
$\quad mentalModelsCapabilities == \{MentalModelsActions\},$
$\quad systemsThinkingCapabilities == \{SystemsThinkingActions\},$
$\quad learningTeamCapabilities == \{TeamLearningActions\},$
$\quad buildingSharedVisionCapabilities == \{SharedVisionActions\},$
$\quad mentalmodels == mentalmodelService,$
$\quad roles == \{roleAttendant\},$
$\quad plans == \{planService01\},$
$\quad personalmastery == personalmasteryServices01,$
$\quad personalmasteryS == personalMasterySchemaServices01,$
$\quad systemsthinkingskill == systemsthinkingServices01,$
$\quad systemsthinkingskillS == systemsThinkingSchemaServices01,$
$\quad store == \{\{atom1, atom2\}\},$
$\quad peractions == peractions1a,$
$\quad canperceive == canperceive1,$
$\quad agperceives == lagperceivesService,$
$\quad agentact == lagentactService,$
$\quad env == env0Service,$
$\quad posspercepts == env0Service,$
$\quad actualpercepts == env0Service,$
$\quad willdo == \{ReceiveNewCustomer\}\ |\!)$

São apresentadas abaixo, as definições das valorações associadas aos ambientes, capacidades e ao estado do agente cozinheiro.

$$
\begin{array}{l}
env0Cook == envIdle \\
action0Cook == \{ReceiveAnOrderForCooking\} \\
env1Cook == envOrder1BeingAnalyzed \\
action1Cook == \{PrepareMeal\} \\
env2Cook == envOrder1BeingCooked \\
action2Cook == \{DeliverMeal\}
\end{array}
$$

As capacidades do agente cozinheiro incluem aquelas relativas às disciplinas e as associadas à atividade de cozinha.

$$
\begin{array}{l}
capabilitiesCook01 == \{DevelopGuidingIdeasActions, \\
\quad EnhanceRealityVisionsActions, ClarifyPersonalVisionsActions, \\
\quad MentalModelsActions, TeamLearningActions, \\
\quad SystemsThinkingActions, SharedVisionActions, PerceptiveActions, \\
\quad ReceiveAnOrderForCooking, PrepareMeal, DeliverMeal\}
\end{array}
$$

Esquemas relativos ao domínio pessoal e pensamento sistêmico são obtidos a partir de valorações associadas ao cozinheiro.

$$
\begin{array}{l}
personalMasterySchemaCook01 == \\
\quad getPersonalMasterySchemaFromBinding\ personalmasteryCook01 \\
systemsThinkingSchemaCook01 == \\
\quad getSystemsThinkingSchemaFromBinding\ systemsthinkingCook01
\end{array}
$$

Concluindo, a valoração associada ao estado do agente cozinheiro é apresentada a seguir.

$$
\begin{aligned}
&agentCarlosstate == agentstate(\!| \; name == CarlosCook,\\
&\quad motivations == motivationCook,\\
&\quad capableof == capabilitiesCook01,\\
&\quad ownguidingideas == guidingideaCarlosCook01,\\
&\quad goals == agentgoalsCook,\\
&\quad attributes == \{\{atom1, atom2\}\},\\
&\quad personalMasteryCapabilities == \{DevelopGuidingIdeasActions,\\
&\qquad EnhanceRealityVisionsActions,\\
&\qquad ClarifyPersonalVisionsActions\},\\
&\quad mentalModelsCapabilities == \{MentalModelsActions\},\\
&\quad systemsThinkingCapabilities == \{SystemsThinkingActions\},\\
&\quad learningTeamCapabilities == \{TeamLearningActions\},\\
&\quad buildingSharedVisionCapabilities == \{SharedVisionActions\},\\
&\quad mentalmodels == mentalmodelCook,\\
&\quad roles == \{roleCook\},\\
&\quad plans == \{planCook01\},\\
&\quad personalmastery == personalmasteryCook01,\\
&\quad personalmasteryS == personalMasterySchemaCook01,\\
&\quad systemsthinkingskill == systemsthinkingCook01,\\
&\quad systemsthinkingskillS == systemsThinkingSchemaCook01,\\
&\quad store == \{\{atom1, atom2\}\},\\
&\quad peractions == peractions1a,\\
&\quad canperceive == canperceive1,\\
&\quad agperceives == lagperceivesCook,\\
&\quad agentact == lagentactCook,\\
&\quad env == env0Cook,\\
&\quad posspercepts == env0Cook,\\
&\quad actualpercepts == env0Cook,\\
&\quad willdo == \{ReceiveAnOrderForCooking\} \; |\!)
\end{aligned}
$$

Neste ponto são definidas valorações associadas aos ambientes, capacidades e ao estado do agente gerente.

$$
\begin{aligned}
&env0Manage == envShopClosed\\
&action0Manage == \{OpenTheShop\}\\
&env1Manage == envShopOpen\\
&action1Manage == \{DefineQualityStandardsForServices\}
\end{aligned}
$$

$env2Manage == envDefinedQualityStandardsForServices$
$action2Manage == \{DefineQualityStandardsForMeals\}$
$env3Manage == envDefinedQualityStandardsForMeals$
$action3Manage == \{DefineRewardsForWellPerformedWork\}$
$env4Manage == envDefinedRewardsForWellPerformedWork$
$action4Manage == \{EvaluateCustomerService\}$
$env5Manage == envEvaluatedCustomerService$
$action5Manage == \{EvaluateCustomerSatisfaction\}$
$env6Manage == envEvaluatedCustomerSatisfaction$
$action6Manage == \{ManageTheShop\}$
$env7Manage == envShopRunning$
$action7Manage == \{CloseTheShop\}$

As capacidades do agente gerente incluem aquelas relativas às disciplinas e as associadas à atividade de gerência.

$capabilitiesManager01 == \{DevelopGuidingIdeasActions,$
$EnhanceRealityVisionsActions, ClarifyPersonalVisionsActions,$
$MentalModelsActions, TeamLearningActions,$
$SystemsThinkingActions, SharedVisionActions,$
$PerceptiveActions, OpenTheShop,$
$DefineQualityStandardsForServices,$
$DefineQualityStandardsForMeals,$
$DefineRewardsForWellPerformedWork,$
$EvaluateCustomerService, EvaluateCustomerSatisfaction,$
$ManageTheShop, CloseTheShop\}$

Esquemas relativos ao domínio pessoal e pensamento sistêmico são obtidos a partir de valorações associadas ao gerente.

$personalMasterySchemaManager01 ==$
$getPersonalMasterySchemaFromBinding\ personalmasteryManager01$
$systemsThinkingSchemaManager01 ==$
$getSystemsThinkingSchemaFromBinding\ systemsthinkingManagement01$

Finalmente, a valoração associada ao estado do agente gerente é apresentada como segue.

$$
\begin{array}{l}
agentMaristate == agentstate(\!| \; name == MariManager, \\
\quad motivations == motivationManager, \\
\quad capableof == capabilitiesManager01, \\
\quad ownguidingideas == guidingideaMariManager01, \\
\quad goals == agentgoalsManager, \\
\quad attributes == \{\{atom1, atom2\}\}, \\
\quad personalMasteryCapabilities == \{DevelopGuidingIdeasActions, \\
\qquad EnhanceRealityVisionsActions, \\
\qquad ClarifyPersonalVisionsActions\}, \\
\quad mentalModelsCapabilities == \{MentalModelsActions\}, \\
\quad systemsThinkingCapabilities == \{SystemsThinkingActions\}, \\
\quad learningTeamCapabilities == \{TeamLearningActions\}, \\
\quad buildingSharedVisionCapabilities == \{SharedVisionActions\}, \\
\quad mentalmodels == mentalmodelManager, \\
\quad roles == \{roleManager\}, \\
\quad plans == \{planManagement01\}, \\
\quad personalmastery == personalmasteryManager01, \\
\quad personalmasteryS == personalMasterySchemaManager01, \\
\quad systemsthinkingskill == systemsthinkingManagement01, \\
\quad systemsthinkingskillS == systemsThinkingSchemaManager01, \\
\quad store == \{\{atom1, atom2\}\}, \\
\quad peractions == peractions1a, \\
\quad canperceive == canperceive1, \\
\quad agperceives == lagperceivesManager, \\
\quad agentact == lagentactManage, \\
\quad env == env0Manage, \\
\quad posspercepts == env0Manage, \\
\quad actualpercepts == env0Manage, \\
\quad willdo == \{OpenTheShop\} \; |\!)
\end{array}
$$

As valorações associadas aos estados dos agentes apresentadas acima usam valorações específicas para os seus planos. Tais definições dos planos são apresentadas a seguir e incluem valorações para os planos associados às funções de atendimento, cozinha e gerência. Novamente, vale ressaltar que foram definidos, neste exemplo, apenas um plano para cada agente. Entretanto, é possível

a definição de diversos planos para cada tipo de agente de acordo com as necessidades de outros casos e exemplos.

$$
\begin{aligned}
&planService01 == \langle ReceiveNewCustomer, GreetTheCustomer,\\
&\quad DisplayMenu, ReceiveOrder, ConfirmOrderWithPrice,\\
&\quad ReceivePayment,\\
&\quad ForwardOrderToKitchen,\\
&\quad CheckStateOfOrder,\\
&\quad ReceivePreparedDish,\\
&\quad DeliverDishToCustomer,\\
&\quad OfferAdditionalSeasonings,\\
&\quad GoodAppetiteSeeYouSoon\rangle
\end{aligned}
$$

$$
\begin{aligned}
&planCook01 == \langle ReceiveAnOrderForCooking,\\
&\quad PrepareMeal,\\
&\quad DeliverMeal\rangle
\end{aligned}
$$

$$
\begin{aligned}
&planManagement01 == \langle OpenTheShop,\\
&\quad DefineMinimumStockLevelOfPreviouslyCookedSideDishes,\\
&\quad DefineQualityStandardsForServices,\\
&\quad DefineQualityStandardsForMeals,\\
&\quad DefineRewardsForWellPerformedWork,\\
&\quad EvaluateCustomerSatisfaction,\\
&\quad EvaluateCustomerService,\\
&\quad EvaluateCookService,\\
&\quad CloseTheShop\rangle
\end{aligned}
$$

As definições de valorações para os domínios e imagens das funções de percepção dos diversos agentes são apresentadas a seguir. Como uma simplificação adotada neste estudo, definiu-se que o ambiente captado através da percepção corresponde exatamente ao ambiente observado.

$lagperceivesService ==$
$\{(motivationServices,$
$\{(mentalmodelService,$
$\{(guidingideaOsvaldoServs01,$
$\{(\{roleAttendant\},$
$\{(agentgoalsServices,$
$\{(envIdle, envIdle),$
$(envCust1WaitServ, envCust1WaitServ),$
$(envCust2WaitServ, envCust2WaitServ),$
$(envCustsWaitServ, envCustsWaitServ),$
$(envCust1BeingServed, envCust1BeingServed),$
$(envCust2BeingServed, envCust2BeingServed),$
$(envCust1SelectMeal, envCust1SelectMeal),$
$(envCust2SelectMeal, envCust2SelectMeal),$
$(envCust1WaitForInfo, envCust1WaitForInfo),$
$(envCust2WaitForInfo, envCust2WaitForInfo),$
$(envCust1PlacedOrder, envCust1PlacedOrder),$
$(envCust2PlacedOrder, envCust2PlacedOrder),$
$(envCust1ConfirmingOrder, envCust1ConfirmingOrder),$
$(envCust2ConfirmingOrder, envCust2ConfirmingOrder),$
$(envCust1ConfirmedOrder, envCust1ConfirmedOrder),$
$(envCust2ConfirmedOrder, envCust2ConfirmedOrder),$
$(envWaitForCust1Payment, envWaitForCust1Payment),$
$(envWaitForCust2Payment, envWaitForCust2Payment),$
$(envCust1HasPaid, envCust1HasPaid),$
$(envCust2HasPaid, envCust2HasPaid),$
$(envCust1WaitForMeal, envCust1WaitForMeal),$
$(envCust2WaitForMeal, envCust2WaitForMeal),$
$(envOrder1Ready, envOrder1Ready),$
$(envOrder2Ready, envOrder2Ready),$
$(envCust1Served, envCust1Served),$
$(envCust2Served, envCust2Served)$
$\})$
$\})$
$\})$
$\})$
$\})$
$\}$

$$
\begin{aligned}
&lagperceivesManager == \{(motivationManager,\\
&\quad \{(mentalmodelManager,\\
&\qquad \{(guidingideaMariManager01,\\
&\qquad\quad \{(\{roleManager\},\\
&\qquad\qquad \{(agentgoalsManager, \{(envIdle, envIdle),\\
&\quad (envCust1BeingServed, envCust1BeingServed),\\
&\quad (envCust2BeingServed, envCust2BeingServed),\\
&\quad (envOrder1BeingAnalyzed, envOrder1BeingAnalyzed),\\
&\quad (envOrder1BeingCooked, envOrder1BeingCooked),\\
&\quad (envCust1Served, envCust1Served),\\
&\quad (envCust2Served, envCust2Served),\\
&\quad (envShopOpen, envShopOpen),\\
&\quad (envDefinedQualityStandardsForServices,\\
&\qquad envDefinedQualityStandardsForServices),\\
&\quad (envDefinedQualityStandardsForMeals,\\
&\qquad envDefinedQualityStandardsForMeals),\\
&\quad (envDefinedRewardsForWellPerformedWork,\\
&\qquad envDefinedRewardsForWellPerformedWork),\\
&\quad (envEvaluatedCustomerService, envEvaluatedCustomerService),\\
&\quad (envEvaluatedCustomerSatisfaction,\\
&\qquad envEvaluatedCustomerSatisfaction),\\
&\quad (envShopRunning, envShopRunning),\\
&\quad (envShopClosed, envShopClosed)\\
&\quad \})\ \})\ \})\ \})\ \})\}
\end{aligned}
$$

$$
\begin{aligned}
&lagperceivesCook ==\\
&\{(motivationCook,\\
&\quad \{(mentalmodelCook,\\
&\qquad \{(guidingideaCarlosCook01,\\
&\qquad\quad \{(\{roleCook\},\\
&\qquad\qquad \{(agentgoalsCook, \{(envIdle, envIdle),\\
&\quad (envOrder1BeingAnalyzed, envOrder1BeingAnalyzed),\\
&\quad (envOrder1BeingCooked, envOrder1BeingCooked),\\
&\quad (envShopOpen, envShopOpen),\\
&\quad (envShopClosed, envShopClosed)\\
&\quad \})\ \})\ \})\ \})\ \})\}
\end{aligned}
$$

As definições de valorações para os domínios e imagens das funções de seleção de ações dos

diversos agentes são apresentadas a seguir. Como uma simplificação adotada neste estudo, definiu-se que o ambiente efetivamente percebido corresponde exatamente ao ambiente observado.

$$
\begin{array}{l}
lagentactService == \\
\{(motivationServices, \\
\quad \{(mentalmodelService, \\
\quad\quad \{(guidingideaOsvaldoServs01, \\
\quad\quad\quad \{(agentgoalsServices, \\
\quad\quad\quad\quad \{(\{planService01\}, \\
\quad\quad\quad\quad \{(envIdle, \\
\quad\quad\quad\quad\quad \{(envIdle, \{ReceiveNewCustomer\})\}), \\
\quad\quad\quad\quad (envCust1WaitServ, \\
\quad\quad\quad\quad\quad \{(envCust1WaitServ, \\
\quad\quad\quad\quad\quad\quad \{GreetTheCustomer\})\}), \\
\quad\quad\quad\quad (envCust1BeingServed, \\
\quad\quad\quad\quad\quad \{(envCust1BeingServed, \{DisplayMenu\})\}), \\
\quad\quad\quad\quad (envCust1SelectMeal, \\
\quad\quad\quad\quad\quad \{(envCust1SelectMeal, \{ReceiveOrder\})\}), \\
\quad\quad\quad\quad (envCust1WaitForInfo, \\
\quad\quad\quad\quad\quad \{(envCust1WaitForInfo, \\
\quad\quad\quad\quad\quad\quad \{AnswerCustomersQuestions\})\}), \\
\quad\quad\quad\quad (envCust1PlacedOrder, \\
\quad\quad\quad\quad\quad \{(envCust1PlacedOrder, \\
\quad\quad\quad\quad\quad\quad \{ConfirmOrderWithPrice\})\}), \\
\quad\quad\quad\quad (envWaitForCust1Payment, \\
\quad\quad\quad\quad\quad \{(envWaitForCust1Payment, \\
\quad\quad\quad\quad\quad\quad \{ReceivePayment\})\}), \\
\quad\quad\quad\quad (envCust1HasPaid, \\
\quad\quad\quad\quad\quad \{(envCust1HasPaid, \\
\quad\quad\quad\quad\quad\quad \{ForwardOrderToKitchen\})\}), \\
\quad\quad\quad\quad (envCust1WaitForMeal, \\
\quad\quad\quad\quad\quad \{(envCust1WaitForMeal, \\
\quad\quad\quad\quad\quad\quad \{ReceivePreparedDish\})\}), \\
\quad\quad\quad\quad (envOrder1Ready, \\
\quad\quad\quad\quad\quad \{(envOrder1Ready, \\
\quad\quad\quad\quad\quad\quad \{DeliverDishToCustomer\})\}), \\
\quad\quad\quad\quad (envCust1Served, \\
\quad\quad\quad\quad\quad \{(envCust1Served, \\
\quad\quad\quad\quad\quad\quad \{GoodAppetiteSeeYouSoon\})\})\ \}\ )\}\ )\}\ )\}\ )\}\ )\}
\end{array}
$$

$$
\begin{array}{l}
lagentactCook == \\
\{(motivationCook, \\
\quad \{(mentalmodelCook, \\
\quad\quad \{(guidingideaCarlosCook01, \\
\quad\quad\quad \{(agentgoalsCook, \\
\quad\quad\quad\quad \{(\{planCook01\}, \\
\quad\quad\quad\quad \{(envIdle, \{(envIdle, \{ReceiveAnOrderForCooking\})\}), \\
\quad\quad\quad\quad (envOrder1BeingAnalyzed, \\
\quad\quad\quad\quad\quad \{(envOrder1BeingAnalyzed, \{PrepareMeal\})\}), \\
\quad\quad\quad\quad (envOrder1BeingCooked, \\
\quad\quad\quad\quad\quad \{(envOrder1BeingCooked, \{DeliverMeal\})\})\ \}\ )\}\ )\}\ )\}\ )\}\ )\}
\end{array}
$$

$$
\begin{array}{l}
lagentactManage == \\
\{(motivationManager, \\
\quad \{(mentalmodelManager, \\
\quad\quad \{(guidingideaMariManager01, \\
\quad\quad\quad \{(agentgoalsManager, \\
\quad\quad\quad\quad \{(\{planManagement01\}, \\
\quad\quad\quad\quad \{(envShopClosed, \{(envShopClosed, \{OpenTheShop\})\}), \\
\quad\quad\quad\quad (envShopOpen, \\
\quad\quad\quad\quad\quad \{(envShopOpen, \\
\quad\quad\quad\quad\quad\quad \{DefineQualityStandardsForServices\})\}), \\
\quad\quad\quad\quad (envDefinedQualityStandardsForServices, \\
\quad\quad\quad\quad\quad \{(envDefinedQualityStandardsForServices, \\
\quad\quad\quad\quad\quad\quad \{DefineQualityStandardsForMeals\})\}), \\
\quad\quad\quad\quad (envDefinedQualityStandardsForMeals, \\
\quad\quad\quad\quad\quad \{(envDefinedQualityStandardsForMeals, \\
\quad\quad\quad\quad\quad \{DefineRewardsForWellPerformedWork\})\}), \\
\quad\quad\quad\quad (envDefinedRewardsForWellPerformedWork, \\
\quad\quad\quad\quad\quad \{(envDefinedRewardsForWellPerformedWork, \\
\quad\quad\quad\quad\quad\quad \{EvaluateCustomerService\})\}), \\
\quad\quad\quad\quad (envEvaluatedCustomerService, \\
\quad\quad\quad\quad\quad \{(envEvaluatedCustomerService, \\
\quad\quad\quad\quad\quad\quad \{EvaluateCustomerSatisfaction\})\}), \\
\quad\quad\quad\quad (envEvaluatedCustomerSatisfaction, \\
\quad\quad\quad\quad\quad \{(envEvaluatedCustomerSatisfaction, \\
\quad\quad\quad\quad\quad\quad \{ManageTheShop\})\}), \\
\quad\quad\quad\quad (envShopRunning, \\
\quad\quad\quad\quad\quad \{(envShopRunning, \{CloseTheShop\})\})\ \}\ )\}\ )\}\ )\}\ )\}\ )\}
\end{array}
$$

A seguir, são apresentadas as valorações associadas aos papéis organizacionais. Estes papéis se referem às atividades de atendimento, cozinha e gerência. Neste estudo, não foram definidas valorações específicas para as metas associadas aos papéis.

$$
\begin{array}{l}
roleAttendant == roleorg\langle\!| \; roleGoals == \{\{\{atom1, atom2\}\}\}, \\
\qquad\qquad skillsRequired == \{Attend\} \; |\!\rangle \\
roleCook == roleorg\langle\!| \; roleGoals == \{\{\{atom1, atom2\}\}\}, \\
\qquad\qquad skillsRequired == \{Cook\} \; |\!\rangle \\
roleManager == roleorg\langle\!| \; roleGoals == \{\{\{atom1, atom2\}\}\}, \\
\qquad\qquad skillsRequired == \{Manage\} \; |\!\rangle
\end{array}
$$

6.2.7 Valorações de Equipes, Visões Compartilhadas e Organização

Os agentes são agrupados para formar uma valoração de uma equipe (*lteamfastfood*). São também definidas as metas e planos dessa equipe.

$$
\begin{array}{l}
fastfoodmembers01 == \{agentOsvaldo, agentCarlos, agentMari\} \\
commongoalsFastFood == \{\{\{atServeBestQualityMeals, \\
\qquad\qquad atOfferBestFastFoodService, atCareerDevelopment\}\}\} \\
commonplansFastFood == \{planService01, planCook01, \\
\qquad\qquad planManagement01\}
\end{array}
$$

$$
\begin{array}{l}
developCommonGoalsFastFood == \{(fastfoodmembers01, \\
\qquad\qquad commongoalsFastFood)\} \\
developcommonplansFastFood == \{(fastfoodmembers01, \\
\qquad\qquad commonplansFastFood)\}
\end{array}
$$

$$
\begin{array}{l}
lteamfastfood == lteamgen\langle\!| \\
\qquad commongoals == commongoalsFastFood, \\
\qquad commonplans == commonplansFastFood, \\
\qquad learningmembers == fastfoodmembers01, \\
\qquad developCommonGoals == developCommonGoalsFastFood, \\
\qquad developcommonplans == developcommonplansFastFood, \\
\qquad teamguidingideas == guidingideaMariManager01 \; |\!\rangle
\end{array}
$$

São descritas, inicialmente, valorações que servem de apoio para a definição da visão compartilhada. Estas valorações são referentes às diretrizes de equipes e à obtenção de visões a partir de um conjunto de diretrizes.

$$
\begin{array}{l}
interteamsgifastfood == \{(\{lteamfastfood\}, \\
\qquad guidingideaMariManager01)\} \\
sharedvisionsfastfood == \{visionsMariManager01\} \\
visionsfromgifastfood == \{(\{guidingideaMariManager01\}, \\
\qquad \{visionsMariManager01\})\}
\end{array}
$$

Desta forma, a valoração associada à visão compartilhada pode ser definida como segue. Como uma simplificação, neste estudo foi definido que a visão compartilhada corresponde exatamente à visão pessoal do gerente. Porém, outro tipo de valoração poderia ter sido adotado.

$$
\begin{array}{l}
svisionfastfood == sharedvisiongen(\!| \\
\qquad learningteams == \{lteamfastfood\}, \\
\qquad interTeamsGuidingIdeas == interteamsgifastfood, \\
\qquad sharedvisions == sharedvisionsfastfood, \\
\qquad visionsFromGuidingIdea == visionsfromgifastfood \ |\!)
\end{array}
$$

Finalmente, uma valoração para a organização, referente a este estudo de caso, pode ser definida. Tal valoração inclui a equipe *lteamfastfood*, que tem a visão compartilhada *svisionfastfood*.

$$
\begin{array}{l}
lorgfastfood == lorggen(\!| \ learningteams == \{lteamfastfood\}, \\
\qquad sharedvision == svisionfastfood \ |\!)
\end{array}
$$

6.3 Operações

Para permitir a realização das diversas operações do estudo de caso, os dados de testes são especificados como seqüências de entrada e de saída. As seqüências de entrada são definidas como pares, que definem um estado do ambiente e uma ação. O objetivo neste exemplo, é definir um arcabouço que permita validar se seqüências de entradas e saídas estão de acordo com a especificação.

Inicialmente, parâmetros genéricos de entrada e de saída são definidos. O parâmetro de entrada é representado por um par contendo um estado do ambiente e uma seqüência de ações. O parâmetro de saída não precisa ser definido nesse exemplo, ou seja, é suficiente que toda operação devolva o mesmo parâmetro de saída.

$$
\begin{array}{l}
INPUTACT ::= \\
\quad actionI \langle\langle Environment \times Actions \rangle\rangle \\
OUTPUTACT ::= \\
\quad okayActionO
\end{array}
$$

Uma ação padronizada é definida, envolvendo pares de entradas e saídas. Estes parâmetros são utilizados em operações do tipo $\Delta AgentStateSchema$. Além disso, esses parâmetros correspondem exatamente àqueles definidos na operação *AgentInteractsInputSchema*.

$$
\begin{array}{l}
UniformAction == \\
\quad [\Delta AgentStateSchema;\ in : INPUTACT;\ out : OUTPUTACT \mid \\
\qquad \exists\, env? : Environment;\ willdo? : Actions \bullet \\
\qquad\quad (env?, willdo?) \mapsto in \in actionI \land \\
\qquad\qquad AgentInteractsInputSchema \land \\
\qquad\qquad out = okayActionO]
\end{array}
$$

A função *stepAction* recebe um par de entradas e saídas e devolve uma relação entre estados de *AgentStateSchema*, antes e após a operação e definidos em relação às entradas e saídas.

$$
\begin{array}{l}
stepAction == \lambda\, in : INPUTACT;\ out : OUTPUTACT \bullet \\
\qquad\qquad \{\Delta AgentStateSchema \mid UniformAction \bullet \\
\qquad\qquad\qquad (\theta AgentStateSchema, \theta AgentStateSchema')\}
\end{array}
$$

Finalizando, define-se também a função recursiva *testAction*. Esta função calcula os estados resultantes em seqüências de dados de entrada e saída. A função recebe como parâmetros um conjunto finito de estados iniciais.

$$
\begin{array}{|l}
testAction : \mathbb{P}\, AgentStateSchema \rightarrow \\
\qquad \text{seq}\, INPUTACT \times \text{seq}\, OUTPUTACT \rightarrow \\
\qquad \mathbb{P}\, AgentStateSchema \\
\hline
testAction = \lambda\, init : \mathbb{P}\, AgentStateSchema \bullet \\
\quad \lambda\, is : \text{seq}\, INPUTACT;\ os : \text{seq}\, OUTPUTACT \mid \\
\qquad \#is = \#os \bullet \\
\qquad \textbf{if}\ \#is = 0\ \textbf{then}\ init \\
\qquad \textbf{else}\ stepAction(last\ is, last\ os) \\
\qquad\quad (\!|testAction(init)(front\ is, front\ os)|\!)
\end{array}
$$

Para a realização dos testes, são definidos alguns dados de entrada. Neste exemplo, apenas pares de estado do ambiente e ação realizada são utilizados.

O primeiro conjunto de dados de teste ($iact0$ a $iact9$) define estados do ambiente e ações de atendimento, o segundo ($mact0$ a $mact7$) refere-se à gerência e o terceiro ($cact0$ a $cact2$) a atividades de cozinha.

$$
\begin{array}{|l}
iact0 == \langle actionI(env0Service, action0Service)\rangle \\
iact1 == \langle actionI(env1Service, action1Service)\rangle \\
iact2 == \langle actionI(env2Service, action2Service)\rangle \\
iact3 == \langle actionI(env3Service, action3Service)\rangle \\
iact4 == \langle actionI(env4Service, action4Service)\rangle \\
iact5 == \langle actionI(env5Service, action5Service)\rangle \\
iact6 == \langle actionI(env6Service, action6Service)\rangle \\
iact7 == \langle actionI(env7Service, action7Service)\rangle \\
iact8 == \langle actionI(env8Service, action8Service)\rangle \\
iact9 == \langle actionI(env9Service, action9Service)\rangle
\end{array}
$$

$$
\begin{array}{|l}
mact0 == \langle actionI(env0Manage, action0Manage)\rangle \\
mact1 == \langle actionI(env1Manage, action1Manage)\rangle \\
mact2 == \langle actionI(env2Manage, action2Manage)\rangle \\
mact3 == \langle actionI(env3Manage, action3Manage)\rangle \\
mact4 == \langle actionI(env4Manage, action4Manage)\rangle \\
mact5 == \langle actionI(env5Manage, action5Manage)\rangle \\
mact6 == \langle actionI(env6Manage, action6Manage)\rangle \\
mact7 == \langle actionI(env7Manage, action7Manage)\rangle
\end{array}
$$

$$
\begin{array}{l}
cact0 == \langle actionI(env0Cook, action0Cook)\rangle \\
cact1 == \langle actionI(env1Cook, action1Cook)\rangle \\
cact2 == \langle actionI(env2Cook, action2Cook)\rangle
\end{array}
$$

Há apenas um tipo de dado de saída para esses testes, pois nesse exemplo, será efetuada apenas uma exposição de seqüências de ações de atendimento, cozinha e gerência.

Podem ser definidas também seqüências maiores de dados de entrada. Por exemplo, as seqüências $iseqact0$, $mseqact0$ e $cseqact0$ representam ciclos completos de ações e estados envolvendo atividades de atendimento, gerência e cozinha, respectivamente.

$$
\begin{array}{l}
iseqact0 == \langle actionI(env0Service, action0Service), \\
\qquad\qquad actionI(env1Service, action1Service), \\
\qquad\qquad actionI(env2Service, action2Service), \\
\qquad\qquad actionI(env3Service, action3Service), \\
\qquad\qquad actionI(env4Service, action4Service), \\
\qquad\qquad actionI(env5Service, action5Service), \\
\qquad\qquad actionI(env6Service, action6Service), \\
\qquad\qquad actionI(env7Service, action7Service), \\
\qquad\qquad actionI(env8Service, action8Service), \\
\qquad\qquad actionI(env9Service, action9Service)\rangle \\
mseqact0 == \langle actionI(env0Manage, action0Manage), \\
\qquad\qquad actionI(env1Manage, action1Manage), \\
\qquad\qquad actionI(env2Manage, action2Manage), \\
\qquad\qquad actionI(env3Manage, action3Manage), \\
\qquad\qquad actionI(env4Manage, action4Manage), \\
\qquad\qquad actionI(env5Manage, action5Manage), \\
\qquad\qquad actionI(env6Manage, action6Manage), \\
\qquad\qquad actionI(env7Manage, action7Manage)\rangle \\
cseqact0 == \langle actionI(env0Cook, action0Cook), \\
\qquad\qquad actionI(env1Cook, action1Cook), \\
\qquad\qquad actionI(env2Cook, action2Cook)\rangle
\end{array}
$$

As seqüências $iseqact1$ e $mseqact1$ representam ciclos parciais de ações e estados envolvendo atividades de atendimento e gerência, respectivamente.

$$
\begin{aligned}
& iseqact1 == \langle actionI(env0Service, action0Service), \\
& \qquad actionI(env1Service, action1Service), \\
& \qquad actionI(env2Service, action2Service), \\
& \qquad actionI(env3Service, action3Service), \\
& \qquad actionI(env4Service, action4Service), \\
& \qquad actionI(env5Service, action5Service), \\
& \qquad actionI(env6Service, action6Service), \\
& \qquad actionI(env7Service, action7Service)\rangle \\
& mseqact1 == \langle actionI(env0Manage, action0Manage), \\
& \qquad actionI(env1Manage, action1Manage), \\
& \qquad actionI(env2Manage, action2Manage), \\
& \qquad actionI(env3Manage, action3Manage), \\
& \qquad actionI(env4Manage, action4Manage), \\
& \qquad actionI(env5Manage, action5Manage)\rangle
\end{aligned}
$$

Os estados iniciais a serem utilizados correspondem exatamente às definições das instâncias de *AgentStateSchema* apresentadas na seção 6.2.

$$
\begin{aligned}
& oact1 == \langle okayActionO\rangle \\
& serviceistate == \{getAgentStateSchemaFromBinding(agentOsvaldostate)\} \\
& manageristate == \{getAgentStateSchemaFromBinding(agentMaristate)\} \\
& cookistate == \{getAgentStateSchemaFromBinding(agentCarlosstate)\}
\end{aligned}
$$

6.4 Testes e Resultados

Nesta seção são descritos alguns dos testes realizados com os dados do exemplo especificado acima. Os testes representam seqüências de ações de serviços de atendimento, gerência e cozinha. São também apresentados os resultados desses testes.

As informações são apresentadas no formato descrito abaixo, onde a cada execução de uma determinada expressão obtém-se um resultado. Uma expressão corresponde a um elemento contido na especificação, como por exemplo: função, predicado, esquema de estado ou de operação. Os resultados apresentados pelo produto ZETA são bem detalhados e prolixos. Quando é solicitada a execução de uma expressão, esses resultados incluem informações sobre as diversas variáveis

declaradas nos tipos associados àquela expressão. Neste trabalho, devido ao grande volume de informações devolvido pelo ZETA a cada execução de uma expressão, serão apresentadas apenas as informações relevantes para este exemplo e que são relativas às percepções, estado do ambiente e ação selecionada pelo agente.

expressao
⇛ {resultado}

Um primeiro exemplo, envolve o atendente, que está inicialmente inativo, mas ao perceber um cliente esperando atendimento, reconhece este estado no ambiente e seleciona a ação de recebimento do cliente (*GreetTheCustomer*).

testAction(*serviceistate*)(*iact*0, *oact*1)
⇛ {<actualpercepts== {{atom <head==WaitingForService,
terms==<const Customer1>>}}, ...
env== {{atom <head==WaitingForService,
terms==<const Customer1>>}}, ...
posspercepts== {{atom <head==WaitingForService,
terms==<const Customer1>>}}, ...
,willdo=={GreetTheCustomer}>,...}

No exemplo seguinte, o gerente encontra a lanchonete aberta e seleciona a ação de definir o padrão de qualidade para os serviços da lanchonete.

testAction(*manageristate*)(*mact*0, *oact*1)

```
⇛ {<actualpercepts== {{atom <head==ShopOpen,
   terms==<const Shop>>}}, ...
     env=={{atom <head==ShopOpen,
   terms==<const Shop>>}}, ...
     posspercepts=={{atom <head==ShopOpen,
   terms==<const Shop>>}}, ...
     willdo=={DefineQualityStandardsForServices}>,...}
```

O cozinheiro inicialmente se encontra inativo, percebe a chegada de um pedido de lanche, o novo ambiente indica que um pedido está sendo analisado, e a ação selecionada por esse agente é preparar o lanche.

testAction(*cookistate*)(*cact*0, *oact*1)

```
⇛ {<actualpercepts== {{atom <head==AnalyzingOrder,
   terms==<const Order1>>}}, ...
     env=={{atom <head==AnalyzingOrder,
   terms==<const Order1>>}}, ...
     posspercepts== {{atom <head==AnalyzingOrder,
   terms==<const Order1>>}}, ...
     willdo=={PrepareMeal}>,...}
```

Os testes a seguir envolvem seqüências maiores de pares de ações e ambientes.

O primeiro caso apresenta testes envolvendo toda a seqüência de ações e ambientes do atendente prevista no exemplo[1]. Desta forma, o agente tem um estado inicial de desocupado, realiza todas as operações associadas às suas atividades, e enquanto isso, altera o ambiente de acordo com essas ações. Ao final do processo, o agente volta ao estado de desocupado, e sua próxima ação é esperar por um novo cliente.

testAction(*serviceistate*)(*iseqact*0, *oseqact*10)

```
⇛ {<actualpercepts=={{atom <head==Idle,
   terms==<const Self>>}},  ...
     env=={{atom <head==Idle,
   terms==<const Self>>}},  ...
     posspercepts=={{atom <head==Idle,
   terms==<const Self>>}}, ...
     willdo=={ReceiveNewCustomer}>,...}
```

[1] As definições das seqüências de saída (*oseqact*10, *oseqact*8, etc.) não são relevantes e foram omitidas.

São apresentados, agora, os testes envolvendo toda a seqüência de ações e ambientes associados ao gerente. Novamente, os resultados mostram que o agente realiza todas as operações previstas. Ao final desse processo, a lanchonete se encontra fechada e a próxima ação do agente é abrí-la.

testAction(*manageristate*)(*mseqact*0, *oseqact*8)
⇛ {<actualpercepts== {{atom <head==ShopClosed,
terms==<const Shop>>}}, ...
env=={{atom <head==ShopClosed,
terms==<const Shop>>}}, ...
posspercepts=={{atom <head==ShopClosed,
terms==<const Shop>>}}, ...
willdo=={OpenTheShop}>,...}

Finalmente, é apresentado o teste envolvendo toda a seqüência de ações e ambientes do cozinheiro. Também nesse caso, todas as operações são bem sucedidas e o cozinheiro, ao final do processo, está desocupado. Sua próxima ação é receber um novo pedido de refeição.

testAction(*cookistate*)(*cseqact*0, *oseqact*3)
⇛ {<actualpercepts=={{atom <head==Idle,
terms==<const Self>>}}, ...
env=={{atom <head==Idle,
terms==<const Self>>}}, ...
posspercepts=={{atom <head==Idle,
terms==<const Self>>}}, ...
willdo=={ReceiveAnOrderForCooking}>,...}

Nos dois exemplos seguintes são realizados testes com transições de estados que não completam um ciclo completo das atividades previstas no estudo de caso para o atendente e para o gerente.

No primeiro exemplo, o atendente já realizou as tarefas necessárias ao recebimento de um

pedido, repasse desse pedido à cozinha, recebimento do pagamento, entre outros, e sua próxima ação é a entrega da refeição ao cliente.

testAction(*serviceistate*)(*iseqact*1, *oseqact*8)

⇒ {<actualpercepts== {{atom <head==BeingServed,

terms==<const Customer1>>, ...

env== {{atom <head==BeingServed,

terms==<const Customer1>>,

atom <head==HasPayed,

terms==<const Customer1,const Order1>>,

atom <head==PlacedOrder,

terms==<const Customer1,const Order1>>,

atom <head==WaitingForMeal,

terms==<const Customer1,const Order1>>,

atom <head==OrderReadyFromKitchen,

terms==<const Order1>>}}, ...

posspercepts== {{atom <head==BeingServed,

terms==<const Customer1>>,

atom <head==HasPayed,

terms==<const Customer1,const Order1>>,

atom <head==PlacedOrder,

terms==<const Customer1,const Order1>>,

atom <head==WaitingForMeal,

terms==<const Customer1,const Order1>>,

atom <head==OrderReadyFromKitchen,

terms==<const Order1>>}}, ...

willdo=={DeliverDishToCustomer}>,...}

No segundo exemplo, o gerente já efetuou a abertura da lanchonete, avaliou a satisfação dos clientes e sua próxima ação é gerenciar a lanchonete.

testAction(*manageristate*)(*mseqact*1, *oseqact*6)

$\Rightarrow$ {<actualpercepts== {{atom <head==ShopOpen,
terms==<const Shop>>,
atom <head==EvaluatedCustomerSatisfaction,
terms==<const Shop>>}}, ...
env== {{atom <head==ShopOpen,
terms==<const Shop>>, ...
atom <head==EvaluatedCustomerSatisfaction,
terms==<const Shop>>}}, ...
posspercepts== {{atom <head==ShopOpen,
terms==<const Shop>>,
atom <head==EvaluatedCustomerSatisfaction,
terms==<const Shop>>}}, ...
willdo=={ManageTheShop}>,...}

Adicionalmente, é possível a consulta aos estados da equipe e da organização. Os resultados são apresentados a seguir.

Por exemplo, uma consulta ao estado da equipe *lteamfastfood* mostra quais são os seus planos e metas comuns, quais agentes estão envolvidos no processo de desenvolvimento de tais planos e metas e quais são as diretrizes da equipe.

lteamfastfood
⇛ lteamgen

```
 <commongoals==  {{{atom <head==ServeBestQualityMeals,
terms==<const Meals>>, ...
 commonplans==  {<ReceiveNewCustomer,
GreetTheCustomer, ...
 developCommonGoals==  {( {agent <attributes==  ...
 developcommonplans==  {( {agent(<attributes==  ...
 teamguidingideas== guidingidea
<purposes== {purpose <purposes== ...
```

No caso de uma consulta ao estado da organização *lorgfastfood* os resultados mostram quais são as suas equipes, detalhando para cada equipe os dados apresentados no exemplo acima. Também são exibidas em detalhes as diretrizes comuns organizacionais, cuja valoração está associada à função *interTeamsGuidingIdeas*.

lorgfastfood
⇛ lorggen

```
 <learningteams== {lteamgen <commongoals== ...
 sharedvision==  sharedvisiongen
<interTeamsGuidingIdeas==
{({lteamgen <commongoals=={{{...}}}, ...
```

É importante destacar que é possível a validação da especificação com relação aos pressupostos da Quinta Disciplina, tal como especificados no modelo formal. No exemplo mostrado a seguir, inicialmente, é apresentada a valoração referente ao estado das diretrizes de um agente.

*guidingideaMariManager*01
⇛ guidingidea

```
<purposes== {purpose <purposes== {{{
   atom <head==CareerDevelopment,terms==<const Self>>,
   atom <head==DevelopInterpersonalSkills, terms==<const Self>>}}}>},
 values== {value <values== {{
   agpredicate positive atom <head==Honesty,terms==<const Self>>,
   agpredicate positive atom <head==Reputation,terms==<const Self>>}}>},
 visions== {vision  <visions== {{{
   atom <head==ImproveProfits,terms==<const Shop>>,
   atom <head==ImproveSalary,terms==<const Worker>>}}}>}>
```

Apenas para relembrar, esta diretriz está definida como segue:

$$
\begin{array}{l}
guidingideaMariManager01 == guidingidea\langle\!| \; visions == \{visionsMariManager01\}, \\
\qquad\qquad\qquad purposes == \{purposesMariManager01\}, \\
\qquad\qquad\qquad values == \{valuesMariManager01\} \; |\!\rangle
\end{array}
$$

Após alterações na definição da diretriz, de forma a tornar inconsistente o conjunto de valores, visão e propósito, passa a ser a seguinte sua definição:

$$\begin{array}{l} guidingideaMariManager01 == guidingidea\langle\!| \; visions == \{visionsMariManager01\}, \\ \qquad\qquad\qquad purposes == \{purposesMariManager01\}, \\ \qquad\qquad\qquad values == \{valuesOsvaldoServs01\} \; |\!\rangle \end{array}$$

A resposta do ZETA para a mesma consulta realizada acima indica problemas com a valoração da diretriz, uma vez que os valores presentes na diretriz não são consistentes com propósitos e visão.

```
guidingideaMariManager01
  ⇛      unresolved constraints:
          LTX:C:\aaaZetaWork\tzInstanciasGIdea01.tex(44.30-46.71) undefined:
           undefined in subgoal {
            resolving LTX:C:\aaaZetaWork\tzGIdea01.tex(19.1-27.12)
            with variables:
             purposes = {purpose(<purposes==...>)}
             values = {value(<values==...>)}
             visions = {vision(<visions==...>)}
            unresolved constraints:
             LTX:C:\aaaZetaWork\tzGIdea01.tex(25.9-25.60) undefined:
              mu-value undefined
             LTX:C:\aaaZetaWork\tzGIdea01.tex(26.9-26.60) undefined:
              mu-value undefined
```

Além disso o ZETA indica problemas na valoração dos modelos mentais:

*guidingideaMariManager*01

⇛ ...

```
LTX:C:\aaaZetaWork\tzInstanciasMentalModel01.tex(66.24-80.22)
 waiting for subgoal {
  resolving '{_S1[_x] = _S2[_x]}'
  with parameters:
   _x = guidingideaMariManager01
   _S1 = {(agentmodel(...),agentmodel(...)),(agentmodel(...),
                                              agentmodel(...))}
   _S2 = {(agentmodel(...),agentmodel(...)),(agentmodel(...),
                                              agentmodel(...))}
  unresolved constraints:
   '_S1[x] = _S2[x]' waiting for variable guidingideaMariManager01
}
```

Como conseqüência, também não é possível efetuar a valoração da instância do agente:

```
guidingideaMariManager01
  ⇛      ...
      LTX:C:\aaaZetaWork\tzInstanciasAgentStatesMari01.tex(127.20-185.22)
          waiting for subgoal {
          resolving LTX:C:\aaaZetaWork\tzAgentState01.tex(6.1-72.12)
          with variables:
           actualpercepts~~ = {{atom(<head==...,terms==...>)}}
           buildingSharedVisionCapabilities~~~~~~~~ =
                                                  {SharedVisionActions}
         ...
           store~~~~~~~~ = {{atom(<head==...,terms==...>),atom(<head==...,
                                                        terms==...>)}}
           clarifypersonalvisions~~ = {((...,...),
                                    {personalvisiongen(...)})}
          unresolved constraints:
           LTX:C:\aaaZetaWork\tzAgentState01.tex(50.12-52.75)
             waiting for subgoal {
             resolving '{_S1[_x] = _S2[_x]}'
             with parameters:
              _x = guidingideaMariManager01
              _S1 = guidingideaMariManager01
              _S2 = guidingideaMariManager01
           unresolved constraints:
           '_S1[x] = _S2[x]' waiting for variable guidingideaMariManager01
```

6.5 Considerações Finais

Neste capítulo foi apresentado um estudo de caso, baseado no modelo da Quinta Disciplina, porém usando um nível de abstração diferente: foi definido apenas um tipo de agente que incorpora as características do tipo *LearningOrgAgent*, definido no capítulo 5.

Para a produção deste estudo, alguns problemas operacionais tiveram que ser resolvidos. Por exemplo, como a especificação deste modelo inclui a definição de uma grande quantidade de tipos e instâncias, ao se tentar realizar uma operação de execução de uma expressão para validar o estado de um agente, o ZETA esporadicamente apresentou várias mensagens referentes à alocação e uso de memória, seguidas de mensagens de erro informando a não obtenção de resposta para a expressão solicitada. Entretanto, após a parada e reinício do ZETA, a mesma solicitação era atendida, apresentando os resultados esperados. Portanto, considerou-se que se trata de um problema esporádico de insuficiência de recursos que o ZETA não consegue gerenciar apropriadamente.

Além disso, como o modelo completo foi especificado e verificado sintaticamente (verificação de tipos) com a ferramenta ZTC [ZTC03], houve também nesta etapa a necessidade de adaptação de pequenas diferenças entre a notação Z do ZTC e a notação do ZETA. A principal delas refere-se à sufixação do nome de um esquema com o símbolo ' quando de sua inclusão em um esquema que define uma operação. No caso do ZTC este símbolo deve ser posicionado imediatamente após o último caracter do nome do esquema, enquanto que na notação em uso pelo ZETA, há a necessidade de se inserir um caracter de espaço em branco entre o nome e o símbolo.

Quanto ao estudo de caso, foi possível efetuar testes em que ciclos completos de operações de atendimento, gerência e cozinha foram validados. Adicionalmente, verificou-se também que a introdução de dados inconsistentes com os requisitos do modelo é reconhecida pelo sistema, causando a apresentação de mensagens de erros. Desta forma, os resultados obtidos com os testes estiveram de acordo com os comportamentos esperados do modelo e, portanto, os aspectos validados nestes testes são coerentes com os pressupostos da teoria de Senge, tal como especificados no modelo formal.

CAPÍTULO 7

Discussão e Observações Finais

Nesta tese foi apresentado um modelo formal para a teoria organizacional da Quinta Disciplina. O objetivo dessa teoria é a construção de um tipo particular de organização: a Organização Aprendiz. Além disso, este modelo foi situado no contexto do arcabouço formal SMART. Como conseqüência, o modelo formal da Organização Aprendiz corresponde a um tipo de Sistema Multi-Agentes onde cada um dos membros da organização é modelado como um agente, cuja especificação formal obedece aos requisitos expostos na teoria de Senge.

Inicialmente foi apresentada sucintamente uma visão geral da área de Teoria Organizacional, seguida de um enfoque particular na abordagem da Aprendizagem Organizacional, onde a teoria da Quinta Disciplina se insere. Após esse posicionamento da teoria de Senge no contexto da área de Teoria Organizacional, uma visão mais detalhada da teoria da Quinta Disciplina foi exposta no capítulo 2.

Em seguida, também foi descrita, de uma forma geral e resumida, a área de Sistemas Multi-Agentes e a relação do arcabouço SMART com esta área. Além disso, foi também apresentado um tutorial sucinto do formalismo adotado para a especificação do SMART, a linguagem formal Z. O objetivo da inclusão desse tutorial foi oferecer o ferramental suficiente para permitir a leitura das especificações presentes nesta tese.

Após a apresentação do contexto e da notação formal associados ao SMART, este arcabouço foi descrito, em linhas gerais, no capítulo 4. Essa descrição introduziu os conceitos básicos do

SMART: atributos, ações, metas e motivações, e suas quatro camadas fundamentais, que incluem: entidades, objetos, agentes e agentes autônomos.

No capítulo 5 foi apresentado, então, o modelo formal para a teoria da Quinta Disciplina. Esse modelo, de modo análogo ao SMART, foi construído em camadas fundamentadas nas camadas básicas do arcabouço, e que apresentam tipos de agentes com crescentes níveis de capacidades. No topo dessa estrutura, é definido o modelo formal para o agente da Organização Aprendiz: o *LearningOrgAgent*, que é um agente autônomo, capaz de desempenhar papéis em organizações, lidar com planos, e que desenvolve as cinco disciplinas de Senge. Adicionalmente, foram analisadas algumas características e propriedades desse modelo formal.

Finalmente, um estudo de caso que apresenta um exemplo de utilização desse modelo formal foi apresentado no capítulo 6. O desenvolvimento deste exemplo foi baseado na técnica de animação da especificação formal. O exemplo descrito envolve uma lanchonete fictícia e operações básicas de atendimento, cozinha e gerência. Com base nessa técnica foram realizados testes e apresentados os seus resultados.

Neste capítulo, são apresentados os trabalhos relacionados, considerações finais sobre o modelo formal e o estudo de caso, conclusões e possíveis desenvolvimentos de trabalhos futuros baseados na pesquisa reportada nesta tese.

7.1 Trabalhos Relacionados - Discussão

Em [KP99] e [MH96], são apresentadas formalizações da teoria "Organization in Action"(OA) [Tho67]. De modo similar ao apresentado nesta tese, em ambos os trabalhos os autores apresentam uma revisão de parte de uma teoria discursiva usando métodos formais.

No caso de [KP99], lógica de predicados de primeira ordem é usada para estudar a estrutura de argumentação que serve de embasamento para as proposições da teoria OA. Por outro lado, em [MH96] é usada uma lógica modal multi-agentes desenvolvida pelos próprios autores, com objetivos similares aos citados acima e com o objetivo adicional de investigar a utilidade e capacidade

expressiva desse tipo de lógica para a formalização de uma teoria discursiva como a OA. Com um enfoque diferente, o modelo formal apresentado nesta tese utilizou a notação Z para a construção de um arcabouço estruturado que pôde ser usado para o estudo da teoria de Senge e que permite que sejam efetuadas investigações envolvendo organizações híbridas, compostas por agentes humanos e artificiais.

Adicionalmente, em [PCG98] e [CP94], diversos trabalhos relacionados à simulação computadorizada de organizações são apresentados. Todos esses trabalhos envolvem a construção de modelos computacionais relacionados a organizações e teorias organizacionais. A seguir, alguns desses trabalhos são citados.

O problema da modelagem da tomada de decisão em equipe é o foco do trabalho apresentado em [KWW98]. Nesse trabalho uma equipe corresponde a um grupo que compartilha uma meta ou propósito. A equipe é composta por membros altamente diferenciados e interdependentes com liderança compartilhada, responsabilidades e atribuições individuais e coletivas. No processo de tomada de decisão em equipe são utilizadas as múltiplas inteligências de seus membros, freqüentemente envolvendo a divisão do problema em subproblemas menores que serão posteriormente integrados para a obtenção da solução da equipe.

Comparativamente, no modelo da Quinta Disciplina, há a definição de três tipos de agrupamentos de agentes: grupo, equipe e equipe aprendiz. Um grupo compartilha recursos, enquanto uma equipe corresponde a um refinamento do tipo grupo e compartilha não só recursos, mas também metas. A equipe aprendiz incorpora todas essas características e, adicionalmente, compartilha diretrizes e planos. As diretrizes incluem visão, propósito e valores compartilhados. O modelo da Quinta Disciplina não especifica processos de tomada de decisão. O que é especificado para cada tipo de agente é a função de seleção de ações. Cada agente membro da equipe aprendiz é do tipo *LearningOrgAgent* e a função *learnorgact* define a seleção de ações para esse tipo de agente. Por sua vez, essa função é influenciada pelas metas do agente. Entretanto, o processo de adoção de metas desse tipo de agente mostra que são adotadas as metas que satisfaçam suas motivações e que sejam consistentes com sua visão, propósito e valor. Além disso, quando diretrizes comparti-

lhadas são desenvolvidas na equipe, estas apresentam um conjunto de visão, propósito e valores que é consistente com a visão, propósito e valores de cada agente da equipe. Como conseqüência, em uma equipe os agentes adotam metas que sejam consistentes com a visão compartilhada e selecionam suas próximas ações com base nessas metas. Logo, de acordo com este modelo, se um dos agentes não adota a visão compartilhada, a organização não pode ser considerada uma Organização Aprendiz devido a ausência desse atributo. Em resumo, diante desse tipo de dilema, no modelo da Quinta Disciplina os agentes cooperam.

O trabalho de [Lin98] se concentra na compreensão e no projeto de organizações que exigem alta confiabilidade. A perspectiva do autor é a de que modelos de simulação são ferramentas para examinar diversas possibilidades de estruturações de organizações. O autor examina os efeitos de condições externas na performance de tomada de decisão em situações onde é importante evitar enganos severos. Esta é uma generalização da teoria da contingência clássica - o efeito do ambiente sobre a performance organizacional e suas implicações sobre sua estrutura. A análise é efetuada por meio de um modelo computacional, permitindo que sejam estudados diversos fatores que afetam o desempenho das organizações em ambientes dinâmicos.

Traçando-se um paralelo com o trabalho reportado nesta tese, deve ser observado que o estudo de estruturas organizacionais não faz parte do escopo desta pesquisa, uma vez que Senge não as discute em detalhes em sua teoria. Entretanto, este modelo formal também permite a definição de diferentes estruturas organizacionais. Adicionalmente, a técnica de animação de especificação adotada no estudo de caso apresentado no capítulo 6, representa um tipo de simulação que permite a análise da conformidade da especificação em relação aos requisitos da teoria da Quinta Disciplina e também sua consistência.

O dilema social de gerar cooperação voluntária entre indivíduos confrontados com opções conflitantes de tempo e esforço é o foco do artigo apresentado em [HG98]. O dilema, nesse caso, é que o indivíduo pode ajudar a criar um bem público, compartilhado por muitos, mas onde os custos individuais desestimulam tal esforço. Alternativamente, o indivíduo pode optar por tirar proveito dos esforços dos outros. Esta questão é fundamental para o estudo do comportamento

cooperativo em organizações.

Comparando o trabalho citado acima com o modelo da Quinta Disciplina, observa-se que a disciplina de visão compartilhada envolve exatamente um processo no qual deve emergir um objetivo de alto nível, compartilhado pelos agentes da organização. Já foi discutido acima o relacionamento entre o desenvolvimento da visão compartilhada, a adoção de metas pelo agente e sua seleção de novas ações. Entretanto, cabe ressaltar aqui que o agente sempre se baseia em suas motivações durante esses processos. Logo, no modelo da Quinta Disciplina, a adoção de uma visão compartilhada por um agente envolve a satisfação das motivações desse agente. Além disso, esta visão é incorporada às suas visões pessoais. Portanto, seguindo os preceitos da disciplina de domínio pessoal, esse agente tentará selecionar ações visando reduzir a tensão criativa entre a realidade atual e a visão compartilhada. A existência de opções conflitantes de tempo e esforço para um dado agente poderia implicar na ausência de motivação por parte desse agente para adotar a visão compartilhada. Entretanto, no modelo da Quinta Disciplina a Organização Aprendiz precisa, necessariamente, desenvolver a visão compartilhada, em um processo que envolve todos os agentes da organização.

No trabalho de [CP98], é criada e examinada uma nova entidade: o *WebBot*. Um *WebBot* é um programa de computador que opera de modo autônomo para realizar uma tarefa, atuando como um conselheiro intelectual e assistente a um parceiro humano (ou a outros *WebBots*). Os *WebBots* são membros da organização que agem, se comunicam, têm memória e requerem coordenação. A questão específica investigada neste artigo envolve os efeitos da honestidade do *WebBot* em relação ao comportamento organizacional individual e coletivo. Nos testes foram construídas organizações onde os *WebBots* eram todos honestos ou todos desonestos. Os resultados mostraram que, ao longo do tempo, *WebBots* honestos fazem mais perguntas enquanto aprendem mais, enquanto *WebBots* desonestos passam a perguntar menos.

Similarmente, no modelo da Quinta Disciplina uma das propriedades que se destacou foi a honestidade do agente em suas interações na organização. De fato, enquanto no trabalho de [CP98] a presença de agentes desonestos resulta em uma redução de desempenho organizacional,

na seção 5.4.2 mostrou-se que a ausência de interações confiáveis entre os agentes inviabiliza o desenvolvimento de uma Organização Aprendiz.

Para finalizar as citações de artigos presentes em [PCG98] é apresentado o trabalho de [SD98]. A construção e a capacidade de reconstrução de organizações que incluem humanos ou agentes computacionais é o ponto central desse trabalho. Duas perspectivas advindas de teorias organizacionais são utilizadas para definir um arcabouço que descreve o problema da estruturação da organização: a teoria da contingência, que destaca a importância da correspondência organização-ambiente; e a teoria sócio-técnica, segundo a qual organizações são sistemas tanto sociais quanto técnicos.

Analogamente, no modelo da Quinta Disciplina tanto agentes humanos como artificiais podem ser representados. Entretanto, a reestruturação dinâmica de organizações híbridas não é tratada neste modelo. De fato, neste modelo a estrutura da organização é representada simplesmente como um grafo conexo onde os nós representam equipes e as arestas representam um relacionamento entre essas equipes. A teoria de Senge não detalha tipos de estruturas organizacionais.

O objetivo do trabalho apresentado em [YS99] é explorar um arcabouço conceitual orientado a agentes e baseado em papéis para modelagem de *workflow*[1]. Nesse trabalho processos de negócios são vistos como uma coleção de agentes autônomos, solucionadores de problemas, que interagem com outros quando interdependências ocorrem. Além disso, o *workflow* é modelado como um conjunto de papéis relacionados. Papéis são definidos em termos de metas, qualificações, obrigações, permissões, protocolos, etc.. São adotados protocolos para governar as interações entre papéis. Aos agentes são atribuídos papéis baseados na avaliação de qualificações e capacidades. O comportamento de um agente é conseqüência de seus estados mentais, tais como intenções, crenças, metas, capacidades, compromissos, etc. Papéis são definidos como uma coleção de deveres, que são modelados como obrigações; e direitos, que são modelados como permissões. Uma vez que um papel tenha sido atribuído a um agente, o agente herda as obrigações e permissões específicas

[1]Este termo não foi traduzido por ser de uso comum na área de computação. No contexto do trabalho citado, a tarefa de *workflow* consiste em descrever a coordenação e desempenho do trabalho realizado em uma organização.

desse papel. A coordenação do *workflow* é obtida através da comunicação entre os agentes. Como conseqüência do fato de papéis representarem um mecanismo importante na construção de agentes organizacionais, o processo de modelagem do *workflow* torna-se similar ao processo de desenho organizacional.

Diferentemente do modelo da Quinta Disciplina, no trabalho citado acima, o modelo da organização é decorrente da definição de papéis organizacionais e da descrição da coordenação e do desempenho desses papéis. Assim, esse modelo é voltado a processos organizacionais, enquanto o modelo da Quinta Disciplina é voltado a uma teoria organizacional. Além disso, nesse trabalho o processo de atribuição de papéis aos agentes é detalhado, o que não ocorre no modelo apresentado nesta tese.

No trabalho apresentado em [K. 99] é relatado um estudo sobre a aprendizagem em Sistemas Multi-Agentes. Esse estudo se baseou no enfoque de [Arg77] para a abordagem da Aprendizagem Organizacional e considera quatro tipos de aprendizagem. O primeiro é denominado de laço simples individual[2] e incrementa o desempenho dentro do escopo de uma norma individual. O segundo é denominado de laço duplo individual[3] e incrementa o desempenho mediante mudança de uma norma individual. O terceiro é denominado de laço simples organizacional e incrementa o desempenho dentro do escopo de uma norma organizacional. Finalmente, o quarto é denominado de laço duplo organizacional e incrementa o desempenho mediante mudança de uma norma organizacional. Para permitir a implementação do sistema, normas individuais são implementadas pelo conhecimento dos indivíduos e normas organizacionais são implementadas pelo conhecimento organizacional. Para apoiar essas normas o conhecimento individual é implementado por um conjunto de regras e o conhecimento organizacional é implementado por um conjunto de conhecimentos individuais. Nesse trabalho, os agentes são implementados por Sistemas Classificadores (*Learning Classifier Systems* - LCS [Gol89]) e o aprendizado do Sistema Multi-Agentes é apoiado por uma extensão do LCS denominada de Sistema Classificador Orientado à Aprendizagem Orga-

[2]Tradução do autor para o termo: *single-loops*.

[3]Tradução do autor para o termo: *double-loops*.

nizacional[4] (*Organizational-learning oriented Classifier System* - OCS). O OCS se baseia em uma arquitetura de Aprendizagem de Máquina baseada em Algoritmos Genéticos[5] (*Genetics-Based Machine Learning* - GBML) e é composto por vários sistemas LCS.

De modo similar ao apresentado nesta tese, o artigo citado acima também apresenta um modelo computacional de uma teoria organizacional vinculada a abordagem de Aprendizagem Organizacional. Entretanto, o tipo de modelagem apresentado é bastante distinto do apresentado nesta tese, pois se utiliza de técnicas de aprendizagem computacional e algoritmos genéticos. Na verdade, o enfoque desse trabalho é diferente do apresentado nesta tese: conceitos definidos em uma teoria organizacional são implementados para permitir o estudo do desempenho computacional num processo que envolve a aprendizagem em Sistemas Multi-Agentes. Mais especificamente, esse trabalho é um representante da área de pesquisas denominada de Aprendizagem em Sistemas Multi-Agentes [WS96, Wei97, Wei01]. Entretanto, deve ser destacado que alguns conceitos do enfoque de [Arg77] encontram correspondência na teoria de Senge. Assim, os laços duplos individual e organizacional são comparáveis, respectivamente, ao domínio pessoal e ao desenvolvimento da visão compartilhada na teoria da Quinta Disciplina. Portanto, existem esses pontos de similaridade entre o modelo de [K. 99] e o modelo da Quinta Disciplina.

Em [HSB02], é apresentado um modelo para especificação de organizações de Sistemas Multi-Agentes que se concentra em aspectos funcionais, estruturais e deônticos. A estrutura envolve os conceitos de papel, relações entre papéis e grupos. Quanto à dimensão funcional, esta inclui os conceitos de missões e planos globais, estruturados em um tipo de árvore de decomposição de metas [6]. No modelo apresentado, as dimensões funcionais e estruturais são independentes de modo a aumentar a flexibilidade do Sistema Multi-Agentes. Assim, a dimensão funcional pode ser alterada sem que haja a necessidade de se modificar a estrutura da organização. Apenas a dimensão deôntica, relativa a permissões e obrigações, tem que ser adaptada de acordo com as

[4]Tradução do autor.

[5]Tradução do autor.

[6]Esta estrutura é denominada de *Social Scheme* pelos autores do artigo.

mudanças desejadas.

Traçando-se um paralelo com o trabalho reportado nesta tese, observa-se que os conceitos de missões e planos globais podem ser comparados com o papel desempenhado pelo conjunto envolvendo visão, valores e propósitos compartilhados no modelo da Quinta Disciplina. Entretanto, no caso do artigo citado é definida uma estrutura para associar as missões e planos globais com metas de mais baixo nível. Ao invés, no modelo da Quinta Disciplina, o desenvolvimento de visão, valores e propósitos compartilhados é dinâmico e resulta de processos envolvendo interações.

Em resumo, as pesquisas mencionadas acima lidam com ferramentas computacionais em geral, e tecnologias de Sistemas Multi-Agentes em particular, para modelar e simular organizações, ou empregam métodos formais, como lógica de primeira ordem, para investigar características de uma teoria organizacional; ou ainda, lidam com o uso de perspectivas advindas de teorias organizacionais para a construção de modelos que possam ser úteis para a implementação de sistemas computacionais ou para a construção de arcabouços de Sistemas Multi-Agentes.

Desta forma, observa-se pontos em comum entre os trabalhos citados e o trabalho apresentado nesta tese. No modelo aqui reportado também é utilizado um método formal para modelar uma teoria organizacional. Entretanto, a novidade aqui está no uso de um método advindo da área de Engenharia de Software para a construção do modelo formal. Além disso, diferentemente dos trabalhos citados, o modelo apresentado nesta tese foi baseado em um arcabouço conceitual e formal de Sistemas Multi-Agentes: o SMART. Como conseqüência, este modelo formal na verdade representa uma perspectiva da implementação da teoria estudada sobre um arcabouço de Sistemas Multi-Agentes.

Adicionalmente, como o SMART e o modelo formal para a Quinta Disciplina não apresentam um detalhamento muito restritivo quanto ao tipo de agente que pode fazer parte da organização, então o modelo produzido deve ser aplicável tanto a organizações humanas, quanto a organizações artificiais (compostas por agentes computacionais) e também a organizações híbridas.

É importante destacar ainda, que, também diferentemente dos trabalhos mencionados, o modelo apresentado nesta tese formaliza os principais aspectos da teoria proposta por Senge, não

se restringindo a características organizacionais específicas, tais como: coordenação, trabalho em equipe, cooperação; que constituem o foco de alguns dos trabalhos citados.

7.2 Considerações Finais

7.2.1 O Modelo Formal

Nesta seção são apresentadas algumas questões relativas ao processo de formalização da teoria de Senge e também questões referentes às características do modelo formal da Quinta Disciplina.

Inicialmente, deve-se destacar que durante o processo de formalização foram observadas situações em que a referência básica para a teoria da Quinta Disciplina [Sen90] apresenta alguns conceitos de modo ambíguo. Por exemplo, Senge afirma em [Sen90, p. 147] que metas e objetivos são distintos de visões. Entretanto, em outro ponto do livro [Sen90, p. 149] o autor afirma que a visão corresponde a um destino específico, uma imagem concreta do futuro desejado. Nesse trabalho de especificação formal são necessárias definições claras para cada conceito e, desta forma, considerou-se que visões, quer sejam pessoais ou compartilhadas, correspondem a metas. Além disso, alguns conceitos (ou tipos) usados neste modelo formal não são definidos explicitamente na teoria de Senge. Por exemplo, não há detalhes em [Sen90] ou [SKR+94] relativos à constituição dos planos dos agentes, ou às suas habilidades. Portanto, a formalização desses conceitos decorre de uma interpretação da teoria da Quinta Disciplina, de conceitos introduzidos no próprio arcabouço SMART, ou ainda, originados de pesquisas recentes da área de Sistemas Multi-Agentes.

Surgiram outras questões envolvendo os conceitos de autonomia, tensão criativa e motivação.

É importante destacar diferenças envolvendo o conceito de autonomia, tal como usado no SMART, e no modelo aqui apresentado. No primeiro caso, a autonomia é uma decorrência do fato de um agente ter motivações, de forma que o agente possui a capacidade de criar suas próprias metas. No modelo formal da teoria da Quinta Disciplina, o *LearningOrgAgent* é autônomo, mas, adicionalmente, está inserido em um contexto organizacional. Portanto, esse agente tem que adotar

as metas associadas a um dado papel para que possa desempenhar com sucesso suas atividades. Caso contrário, por ser um agente autônomo, ele pode se recusar a adotar essas metas, mas não poderá desempenhar esse papel na organização. Além disso, papéis têm um nível de autonomia, num sentido organizacional, associado. Assim, o agente pode ter a permissão para gerar suas próprias metas no contexto de um dado papel. Claramente, a autonomia associada às motivações do agente é um pré-requisito para que este possa desempenhar papéis que possuam autonomia organizacional associada.

Adicionalmente, o conceito de tensão criativa na teoria da Quinta Disciplina está relacionado ao o conceito de motivação no SMART. Na verdade, como a tensão criativa pode ser vista como uma função da medida da distância entre o estado atual do ambiente e as metas mantidas por um dado agente, a própria necessidade de redução da tensão criativa pode ser interpretada como uma motivação.

Considerando-se as propriedades derivadas do modelo formal apresentado no capítulo 5, é interessante destacar a importância que algumas características individuais apresentam em uma organização que planeje implementar a teoria da Quinta Disciplina. Assim, é possível observar que os agentes precisam ser honestos, cooperativos, tenazes e que as interações entre esses agentes também precisam ser honestas: é fundamental que haja confiança entre os agentes envolvidos em interações. Portanto, há diversas restrições quanto às características dos agentes que são membros desse tipo de organização formal.

Outras questões importantes incluem: normas e estrutura organizacional, centralização versus descentralização, colaboração e ação coordenada, planejamento e construção da organização. Embora sejam relevantes, estas questões não puderam ser analisadas no contexto do modelo formal da Quinta Disciplina, uma vez que se referem a aspectos que não são claramente definidos na teoria de Senge e para os quais, em alguns casos, são apresentadas apenas definições simplificadas no modelo. Não obstante, são apresentadas abaixo algumas considerações sobre essas questões.

- As normas organizacionais devem favorecer a autonomia e a pró-atividade

Cada agente do modelo formal da Organização Aprendiz precisa desenvolver o domínio pessoal. Assim, todo agente deve tentar descobrir qual é a visão de futuro que ele realmente almeja realizar e deve desenvolver uma visão clara da realidade, incluindo sua contribuição para o estado atual dessa realidade. Entretanto, em organizações que estabeleçam normas muito restritivas, pode ocorrer a delegação de pouca autoridade, autonomia, e responsabilidade para os escalões de níveis hierárquicos mais baixos da organização. Como conseqüência, agentes desses níveis não podem perceber claramente como suas ações produzem o estado atual da realidade, afinal, eles estão apenas obedecendo ordens, e não têm o poder necessário para realizar as mudanças que desejem.

- Centralização, descentralização e a Organização Aprendiz

  Em uma organização centralizada, poder, autoridade, recursos e responsabilidade estão concentrados em um pequeno grupo de seus membros. Tal organização não delega autonomia para os demais níveis de sua hierarquia. Desta forma, planejamento, controle e tomada de decisões estão centralizados em um pequeno grupo de agentes enquanto as ações são realizadas por um grande contingente de agentes. Como uma conseqüência da complexidade das interações envolvendo um grande número de agentes (e as limitações humanas para lidar com tal complexidade) esta organização lida com papéis, os quais correspondem a modelos básicos do indivíduo na organização. Além disso, não há necessidade de modelos mais complexos dos agentes nos níveis hierárquicos mais baixos, pois as normas, decisões e recursos são gerenciados pelo nível administrativo e impostos aos demais membros. Os gerentes esperam que os demais agentes desempenhem suas atividades e atinjam seus resultados tal como definido em seus respectivos papéis. Tal situação corresponde a uma hierarquia de comando e controle que apenas requer obediência dos agentes [Sen90, p. 222].

  Por outro lado, em uma organização descentralizada diversos níveis hierárquicos gerenciam recursos e têm autoridade e responsabilidade. Entretanto, também nesse caso a complexidade de interações envolvendo um grande número de agentes pode se tornar um problema.

Assim, também existem papéis nesse tipo de organização. Porém, adicionalmente, cada agente pode possuir modelos mais detalhados dos demais agentes com os quais interage mais freqüentemente, por exemplo, no contexto de uma equipe. Nessa situação, o potencial para produzir melhores resultados pode crescer não só em conseqüência da alocação de recursos e delegação de autoridade para um maior número de agentes, mas também pelo surgimento de interações mais eficientes entre os agentes.

- Por que os agentes colaboram

  Senge reconhece basicamente três tipos de comportamentos dos agentes em relação a uma visão compartilhada:

  - Participação: é o processo de se tornar parte de algo por opção.
  - Comprometimento: inclui a participação e, adicionalmente, um sentimento de responsabilidade em relação à realização da visão.
  - Aceitação: o agente segue e apóia a visão, fazendo apenas o que se espera dele. Entretanto, ele não participa nem se compromete com ela. Há ainda, diferentes níveis de aceitação: genuína, formal, hostil, apática e não-aceitação.

  Senge também reconhece que, apenas tomando-se como base o comportamento do agente, é difícil diferenciar aceitação genuína do comprometimento ou da participação. Entretanto, de acordo com Senge [Sen90, p. 221], o comprometimento traz energia, paixão e excitação.

  Em uma Organização Aprendiz ideal (do ponto de vista de Senge) os agentes desenvolvem comprometimento em relação à visão compartilhada. Os agentes não apenas obedecem as "regras do jogo", mas também sentem-se responsáveis por tal jogo, chegando a mudar as regras que eventualmente impeçam ou dificultem a realização da visão.

  Assim, em tal organização a colaboração entre os agentes ocorre no nível do comprometimento com metas conjuntas. Entretanto, tal comprometimento resulta do fato de que as

razões subjacentes às ações conjuntas estão enraizadas nas motivações do agente. Como conseqüência, a colaboração também depende das motivações dos agentes da organização.

- Ação coordenada

  As atividades e tarefas organizacionais são distribuídas entre as diversas equipes que constituem a organização, embora a teoria de Senge não defina claramente como isso deva acontecer na Organização Aprendiz. Entretanto, as equipes devem ter autonomia para definir como os agentes irão atuar e podem propor ações ou estratégias de modo a lidar com limitações ou dificuldades peculiares a um ambiente em particular onde a equipe opere. Tal característica confere flexibilidade à organização.

- Planejamento e aprendizagem

  Nesta teoria, o planejamento também é visto como uma oportunidade para o aprendizado e revisão dos modelos mentais dos agentes da organização. Ao estudar diferentes cenários futuros, os agentes podem refletir sobre as hipóteses subjacentes embutidas em cenários específicos, os relacionamentos entre essas hipóteses e como possíveis alterações nessas hipóteses podem levar a cenários diferentes.

- Por quê e como organizações são formadas

  Pode-se considerar que uma Organização Aprendiz é formada inicialmente por uma célula original que forma uma equipe aprendiz, com agentes motivados a desenvolver as cinco disciplinas, e na qual emergem diretrizes compartilhadas iniciais. Estas diretrizes englobam os valores, propósito e visão iniciais, e devem ser apresentadas e discutidas com os demais membros da organização. Deve ser incluído nesse processo o estímulo ao desenvolvimento das cinco disciplinas por parte dos demais agentes, bem como a divulgação dos princípios e ações associados a cada disciplina.

  A organização é construída e se desenvolve porque seus membros têm em suas visões pessoais e motivações a necessidade de criar o futuro desejado e crêem mutuamente que eles podem

realizá-lo por meio da colaboração na organização.

Finalmente, é importante destacar que o modelo formal apresentado neste trabalho corresponde à interpretação da teoria de Senge efetuada pelo autor desta tese e retrata as principais características dessa teoria. Assim, não houve neste trabalho a pretensão de produzir uma tradução total e detalhada da teoria da Quinta Disciplina em um modelo formal. Por exemplo, a maior parte das capacidades e atividades associadas ao pensamento sistêmico estão encapsuladas em funções. Entretanto, o nível de abstração apresentado neste modelo mostrou-se adequado para o estudo de propriedades importantes da teoria da Quinta Disciplina e para discutir características individuais e organizacionais que devem ser consideradas em situações de implementação da teoria da Quinta Disciplina, quer seja em uma organização humana em organizações artificiais, ou híbridas.

7.2.2 O Estudo de Caso

Quanto ao estudo de caso apresentado no capítulo 6 desta tese, foi possível efetuar testes em que ciclos completos de operações de atendimento, gerência e cozinha foram validados. Desta forma, conclui-se que a especificação do modelo simplificado é coerente com os pressupostos avaliados da teoria de Senge e que os resultados obtidos com os testes estiveram de acordo com os comportamentos esperados do sistema.

É importante também destacar que, neste trabalho, houve a necessidade de especificação completa do domínio e imagem de todas as funções presentes no modelo. Este é um requisito decorrente da técnica usada no estudo de caso. Entretanto, o domínio e imagem de algumas das funções não é finito. Esse é o caso, por exemplo, da função *effectinteraction*.

Foram processados ciclos completos de ações que representam o atendimento ao cliente, e operações de gerência e cozinha. Assim, a especificação deste modelo foi validada e os resultados foram compatíveis com o comportamento esperado do sistema: para o estado inicial de cada agente as operações no estudo de caso produziram corretamente estados válidos no modelo formal.

Adicionalmente, com relação à consistência, também foi possível observar que violações às

restrições impostas pela especificação foram reconhecidas corretamente. Por exemplo, não foi possível criar uma valoração para um agente cujas diretrizes eram compostas pela visão do gerente, propósitos associados ao cozinheiro e valores do atendente, uma vez que esta combinação de valores, propósitos e visão não estava especificada nos conjuntos domínio e imagem das respectivas funções de validação de consistência de diretrizes. Portanto, usando esta técnica é possível validar se os dados de testes usados neste estudo de caso seguem as hipóteses subjacentes na teoria de Senge[7]. Conclui-se então que usando conjuntos adequados de dados de testes, aspectos particulares de uma dada organização podem ser validados em relação a estas hipóteses, e a validação da aderência de uma organização em relação a uma teoria organizacional específica pode ser investigada.

Este uso de métodos formais para a modelagem de sistemas em geral apresenta novas perspectivas, e revela novas potencialidades, ainda não exploradas, em relação ao uso destes métodos.

Este estudo de caso também mostra que diferentes tipos de agentes podem ser modelados com o uso de diferentes níveis de abstração para representar aspectos como: modelos mentais, domínio pessoal, pensamento sistêmico, entre outros. Desta forma, um modelo de agente como o do atendente poderia facilmente representar um empregado humano ou um atendente automatizado que recebe ordens, processa pagamentos e entrega as refeições. Neste caso, diretrizes, metas, planos e domínio pessoal, por exemplo, poderiam ser codificados como programas de computador. Tais programas, por sua vez, poderiam ser simples máquinas de estados finitos, ou poderiam usar modelos adaptativos mais complexos para representar clientes, colegas, a organização e os relacionamentos entre essas entidades. Supondo-se que esse modelo represente as lojas de uma cadeia de lanchonetes, a partir de uma perspectiva gerencial, é possível então, a utilização do mesmo modelo formal para representar lojas com diferentes níveis de automação.

[7]De acordo com a representação formal destas hipóteses no modelo formal apresentado nesta tese.

7.3 Conclusões

Conforme discussão apresentada na seção 1.1, processos de formalização em geral podem permitir a sistematização, interpretação e abstração de conceitos intuitivos e experiências, e a comunicação não ambígua e rigorosa de um sistema de conhecimentos. Na pesquisa apresentada nesta tese o sistema de conhecimentos escolhido foi a teoria da Quinta Disciplina.

Neste trabalho, os conceitos apresentados em linguagem textual por Senge foram codificados como estruturas da linguagem Z, respeitando-se as regras impostas por esta linguagem para definir e relacionar estas estruturas.

O resultado deste trabalho, um modelo formal da teoria da Quinta Disciplina, atendeu a dois propósitos:

- Validação

  O modelo pôde ser utilizado para se efetuar a validação da aderência de uma dada organização fictícia aos preceitos da teoria de Senge.

- Modelar Diferentes Níveis de Automação da Organização

  Usando o mesmo modelo, tanto agentes humanos quanto agentes artificiais, tais como sistemas computacionais, podem ser representados. De um ponto de vista gerencial, a visão da organização, por intermédio do modelo, será a mesma.

A pesquisa e projeto reportados nesta tese apresentaram como suas principais contribuições:

- O modelo formal da Quinta Disciplina, com as características citadas de precisão e rigor.

- A utilização deste modelo como ferramenta de apoio para a validação e modelagem citados acima.

- Discussão de propriedades observadas no modelo formal e que devem ser consideradas na implementação da teoria de Senge em organizações artificiais, humanas, ou híbridas.

- A utilização de métodos formais para a modelagem de sistemas em geral, apresentando novas perspectivas e potencialidades em relação ao uso destes métodos.
- Um estudo de caso baseado na técnica de animação da especificação formal.

7.4 Trabalhos Futuros

O desenvolvimento de um modelo formal para a teoria da Quinta Disciplina com base em um arcabouço de Sistemas Multi-Agentes, e a investigação de propriedades e características desse modelo representou o foco do trabalho apresentado nesta tese. Entretanto, alguns outros temas associados à esse trabalho mostraram-se relevantes para o desenvolvimento de investigações e trabalhos futuros. Estes temas incluem:

- Refinamento do modelo apresentado visando sua implementação computacional, usando ferramentas de apoio, tais como os ambientes para desenvolvimento de Sistemas Multi-Agentes *act*SMART [AL01, dL04] ou o ambiente SACI [HS00].
- Investigação da viabilidade da construção de um arcabouço para SMA de grande porte que implemente características observadas no modelo formal da Quinta Disciplina.
- Maior detalhamento de pontos não cobertos pela teoria de Senge, tais como: planejamento, capacidades, emergência da visão compartilhada. Tal trabalho pode envolver o uso de resultados obtidos em outras pesquisas que apresentam um maior detalhamento desses aspectos, como por exemplo, [HSB02], ou modelos que especifiquem mais detalhadamente processos, tais como o processo de emergência de modelos mentais organizacionais a partir de modelos mentais individuais, como apresentado em [Kim93].
- Especificação de um modelo geral que envolva diferentes tipos de estruturas organizacionais: centralizada, descentralizada, etc.; e diferentes teorias organizacionais: Clássica, Relações Humanas, Contingencial, etc..

Referências Bibliográficas

[AL01] R. Ashri and M. Luck. Towards a Layered Aproach for Agent Infrastructure: the Right Tools for the Right Job. In *Proceedings of the Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS*, pages 9–16. http://citeseer.nj.nec.com/445235.html, 2001. [264]

[Arg77] C. Argyris. Double Loop Learning in Organizations. *Harvard Business Review*, pages 115–125, Sept-Oct 1977. [14, 21, 253, 254]

[BCV98] R. H. Bordini, J. A. Campbell, and R. Vieira. Extending Ascribed Intensional Ontologies with Taxonomical Relations in Anthropological Descriptions of Multi-Agent Systems. *Journal of Artificial Societies and Social Simulation*, 1(4), 1998. [4]

[BSC94] R. Barden, S. Stepney, and D. Cooper. *Z in Practice.* Prentice Hall, 1994. [74]

[BW00] R. W. Brennan and O. William. A Simulation Test-Bed to Evaluate Multi-Agent Control of Manufacturing Systems. In J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, editors, *Proceedings of the 2000 Winter Simulation Conference.* http://citeseer.nj.nec.com/518721.html, 2000. [41]

[CD65] V. Cangelosi and W. Dill. Organizational Learning: Observations Toward a Theory. *Administrative Sciences Quarterly*, 10:175–203, 1965. [15]

[Che80] B. Chellas. *Modal Logic: An Introduction.* Cambridge University Press: Cambridge, England, 1980. [5, 43]

[Chi00] I. Chiavenatto. *Introdução à Teoria Geral da Administração. Edição Compacta.* Editora Campus Ltda, second edition, 2000. [7, 8, 11, 12, 13, 174]

[CLW99] M. Crossan, H. Lane, and R. White. An Organizational Learning Framework: from Intuition to Institution. *Academy of Management Review*, 24(3):522–537, 1999. [16, 22]

[CP94] K. M. Carley and M. J. Prietula, editors. *Computational Organization Theory.* Lawrence Erlbaum Associates, Publishers: Hillsdale, NJ, 1994. [19, 42, 249]

[CP98] K. M. Carley and M. J. Prietula. Webbots, Trust, and Organizational Science. In M. J. Prietula, K. M. Carley, and L. Gasser, editors, *Simulating Organizations*, pages 3–22. AAAI Press/The MIT Press, 1998. [1, 20, 42, 251]

[Dec98] K. Decker. Coordinating Human and Computer Agents. *Lecture Notes in Computer Science*, 1364:77–98, 1998. [41]

[Dil94] A. Diller. *Z An Introduction to Formal Methods.* John Wiley & Sons, second edition, 1994. [74]

[dL96] M. d'Inverno and M. Luck. A Formal View of Social Dependence Networks. In Zhang and Lukose, editors, *Distributed Artificial Intelligence Architecture and Modelling: Proceedings of the First Australian Workshop on Distributed Artificial Intelligence*, pages 115–129. Springer-Verlag: Heidelberg, Germany, 1996. [80, 81, 83, 274]

[dL01] M. d'Inverno and M. Luck. *Understanding Agent Systems.* Springer-Verlag, 2001. [4, 5, 42, 43, 59, 60, 61, 64, 67, 69, 95, 96, 101, 123, 124, 125, 128, 174, 177]

[dL04] M. d'Inverno and M. Luck. *Understanding Agent Systems.* Springer-Verlag, second edition, 2004. [59, 264]

[DLPW95] K. Decker, V. Lesser, M. V. Nagendra Prasad, and T. Wagner. MACRON: An architecture for multi-agent cooperative information gathering. In T. Finin and J. Mayfield, editors, *Proceedings of the CIKM '95 Workshop on Intelligent Information Agents*, Baltimore, Maryland, 1995. [42]

[DS89] R. Duke and G. Smith. Temporal Logic and Z Specifications. *Australian Computer Journal*, 21(2):62–66, 1989. [146]

[DTC97] D. Dunphy, D. Turner, and M. Crawford. Organizational Learning as the Creation of Corporate Competencies. *Journal of Management Development*, 16(4):232–244, 1997. [16]

[EW95] O. Etzioni and D. S. Weld. Intelligent Agents on the Internet: Fact, Fiction, and Forecast. *IEEE Expert*, 10(3):44–49, 1995. [42]

[Fay50] H. Fayol. *Administração Industrial e Geral.* Ed. Atlas, São Paulo, 1950. [9]

[FFMM94] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language. In N. Adam, B. Bhargava, and Y. Yesha, editors, *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463, Gaithersburg, MD, USA, 1994. ACM Press. [130]

[fIPA00] Foundation for Intelligent Physical Agents. FIPA ACL Message Structure Specification. url: http://www.fipa.org/, 2000. [130]

[Fis97] C. Fischer. CSP-OZ: a combination of Object-Z and CSP. In H. Bowman and J. Derrick, editors, *Proc. 2nd IFIP Workshop on Formal Methods for Open Object-Based Distributed Systems (FMOODS)*, pages 423–438, Canterbury, UK, 1997. Chapman and Hall, London. [172]

[FL85] C. M. Fiol and M. A. Lyles. Organizational Learning. *Academy of Management Review*, 10(4):803–813, 1985. [14]

[Fla99] J. E. Flaherty. *Peter Drucker: Shaping the Managerial Mind.* Jossey-Bass Publishers - San Francisco, Calif., 1999. [12]

[fSIEC02] International Organization for Standardization - International Electrotechnical Commission. Information technology - Z formal specification notation - Syntax, type system and semantics - ISO/IEC 13568. url: http://www.iso.org/, 2002. [5]

[Gar93] D. A. Garvin. Building a Learning Organization. *Harvard Business Review*, pages 78–91, Jul-Aug 1993. [14, 22]

[GB03] W. Grieskamp and R. Büssow. The ZETA System. 2003. url: http://uebb.cs.tu-berlin.de/zeta. [175]

[Gol89] E. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989. [253]

[Gru93] T. R. Gruber. A Translation Approach to Portable Ontologies. *J. on Knowledge Acquisition*, 5(2):199–220, 1993. [171]

[HG98] B. A. Huberman and N. S. Glance. Fluctuating Efforts and Sustainable Cooperation. In M. J. Prietula, K. M. Carley, and L. Gasser, editors, *Simulating Organizations*, pages 89–103. AAAI Press/The MIT Press, 1998. [20, 250]

[Hoa78] C. A. R. Hoare. Communicating Sequential Processes. *Communications of the ACM*, 21(8):666–677, 1978. [5, 43, 172]

[HS00] J. F. Hübner and J. S. Sichman. SACI: Uma Ferramenta para Implementação e Monitoração da Comunicação entre Agentes. In M. C. Monard and J. S. Sichman,

editors, *IBERAMIA/SBIA 2000 open discussion track: proceedings*, pages 47–56. http://www.lti.pcs.usp.br/saci/doc/iberamia2000-saci.pdf, 2000. [264]

[HSB02] J. F. Hübner, J. S. Sichman, and O. Boissier. A Model for the Structural, Functional, and Deontic Specification of Organizations in Multiagent Systems. In G. Bittencourt and G. L. Ramalho, editors, *Proceedings of the 16th Brazilian Symposium on Artificial Intelligence (SBIA'02)*, LNAI 2507, pages 118–128, Porto de Galinhas, PE, Brazil, 2002. Springer. [22, 42, 254, 264]

[Jac97] J. Jacky. *The Way of Z: Practical Programming with Formal Methods.* Cambridge University Press, 1997. [43, 74]

[Jon90] C. B. Jones. *Systematic Software Development using VDM.* Prentice Hall, second edition, 1990. [5, 43]

[K. 99] K. Takadama and T. Terano and K. Shimohara. Can multiagents learn in organization? – analyzing organizational learning-oriented classifier system. In *IJCAI'99 Workshop on Agents Learning about, from and other Agents.* http://citeseer.nj.nec.com/123628.html, 1999. [21, 253, 254]

[Kim93] D. Kim. The Link Between Individual and Organizational Learning. *Sloan Management Review*, 35(1):37–50, 1993. [15, 264]

[KP99] J. Kamps and L. Pólos. Reducing Uncertainty: A Formal Theory of Organizations in Action. *American Journal of Sociology*, 104:1776–1812, 1999. [19, 248]

[KWW98] M. C. Kang, L. B. Waisel, and W. A. Wallace. Team Soar: A Model for Team Decision Making. In M. J. Prietula, K. M. Carley, and L. Gasser, editors, *Simulating Organizations*, pages 23–45. AAAI Press/The MIT Press, 1998. [20, 249]

[Ld01] M. Luck and M. d'Inverno. A Conceptual Framework for Agent Definition and Development. *The Computer Journal*, 44(1):1–20, 2001. [78]

[Lin98] Z. Lin. The Choice Between Accuracy and Errors: A Contingency Analysis of External Conditions and Organizational Decision Making Performance. In M. J. Prietula, K. M. Carley, and L. Gasser, editors, *Simulating Organizations*, pages 67–87. AAAI Press/The MIT Press, 1998. [20, 250]

[LyLLd01] F. Lopez, y Lopez, M. Luck, and M. d'Inverno. A Framework for Norm-Based Inter-Agent Dependence. In *Proceedings of The Third Mexican International Conference on Computer Science. SMCC-INEGI*, pages 31–40. http://citeseer.nj.nec.com/lopez01framework.html, 2001. [4]

[MH96] M. Masuch and Z. Huang. A Case Study in Logical Deconstruction: Formalizing J. D. Thompson's Organizations in Action in a Multi-Agent Action Logic. *Computational and Mathematical Organization Theory*, 2(2):71–114, 1996. [19, 248]

[Mil89] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989. [5, 43]

[MLd03] S. Munroe, M. Luck, and M. d'Inverno. Towards Motivation-Based Decisions for Worth Goals. To appear as a paper in The Third International/Central and Eastern European Conference on Multi-Agent Systems - CEEMAS 2003. Prague, Czech Republic, 6 2003. [4]

[Mou01] A. V. Moura. *Especificações em Z: uma Introdução.* Editora da Unicamp, 2001. [74]

[MSB99] J. Morabito, I. Sack, and A. Bhate. *Organization Modeling.* Prentice-Hall, Inc., 1999. [6]

[Non91] I. Nonaka. The Knowledge-Creating Company. *Harvard Business Review*, pages 96–104, November-December 1991. [16]

[PCG98] M. J. Prietula, K. M. Carley, and L. Gasser, editors. *Simulating Organizations.* AAAI Press/The MIT Press, 1998. [6, 18, 19, 21, 249, 252]

[PFL+99] Y. Peng, T. Finin, Y. Labrou, R. S. Cost, J. Long B. Chu, W. J. Tolone, and A. Boughannam. An Agent-Based Approach for Manufacturing Integration - the CIIMPLEX Experience. *International Journal of Applied Artificial Intelligences*, 13(1–2):39–64, 1999. [41]

[PST96] B. Potter, J. Sinclair, and D. Till. *An Introduction to Formal Methods and Z.* Prentice Hall, second edition, 1996. [74]

[Rob00] S. P. Robbins. *Organizational Behavior.* Upper Saddle River, NJ. Prentice-Hall, Inc, 2000. [6, 174]

[SD98] Y. So and E. H. Durfee. Designing Organizations for Computational Agents. In M. J. Prietula, K. M. Carley, and L. Gasser, editors, *Simulating Organizations*, pages 47–64. AAAI Press/The MIT Press, 1998. [21, 252]

[Sen90] P. Senge. *The Fifth Discipline - The Art and Practice of the Learning Organization.* Currency Doubleday, 1990. [1, 5, 17, 25, 29, 31, 32, 33, 34, 75, 102, 105, 256, 258, 259]

[Sen02] P. Senge. *A Quinta Disciplina - A Arte e Prática da Organização Que Aprende.* Editora Best Seller, 13th edition, 2002. [25, 103]

[SKR+94] P. Senge, A. Kleiner, C. Roberts, R. Ross, and B. Smith. *The Fifth Discipline Fieldbook - Strategies and Tools for Building a Learning Organization.* Currency Doubleday, 1994. [25, 37, 96, 106, 151, 256]

[Spi89] J. M. Spivey. *Understanding Z - A Specification Language and its Formal Semantics.* Cambridge University Press, 1989. [43, 74]

[Spi92] J. M. Spivey. *The Z Notation: A Reference Manual.* url: http://spivey.oriel.ox.ac.uk/ mike/zrm/, 2nd edition, 1992. [5, 43, 74]

[SS04a] L. P. Silva and F. C. S. Silva. A Case Study for the Fifth Discipline Formal Model. *Computational and Mathematical Organization Theory*, 2004. (submitted). [175]

[SS04b] L. P. Silva and F. C. S. Silva. A Formal Model for the Fifth Discipline. *The Journal of Artificial Societies and Social Simulation*, 2004. (accepted). [73]

[Syc98] K. Sycara. Multiagent Systems. *AI Magazine*, 19(2):79–92, 1998. [40, 41, 42]

[Tay03] F. W. Taylor. *Shop Management.* Harper and Bros., New York, 1903. [9]

[Tho67] J. D. Thompson. *Organizations in Action: Social Science Bases of Administrative Theory.* McGraw-Hill, New York, 1967. [19, 248]

[Wan90] H. Wang. *Computation, Logic, Philosophy: A Collection of Essays.* Kluwer Adacemic Publishers, 1990. [2]

[Wei97] G. Weiss, editor. *Distributed Artificial Intelligence Meets Machine Learning - Learning in Multi-Agent Environments.* ECAI'96 Workshop LDAIS - Budapest, Hungary. Springer-Verlag, Berlin, 1997. [254]

[Wei01] G. Weiss, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.* MIT Press, 2001. [168, 254]

[WJ95] M. J. Wooldridge and N. R. Jennings. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2):115–152, June 1995. [42]

[WS96] G. Weiss and S. Sen, editors. *Adaptation and Learning in Multi-Agent Systems.* IJCAI'95 Workshop - Montreal, Canada. Springer-Verlag, Berlin, 1996. [254]

[YS99] L. Yu and B. Schmid. A Conceptual Framework for Agent Oriented and Role Based Workflow Modeling. In *Presented at CaiSE Workshop on Agent Oriented Information Systems (AOISt99) - Heidelberg, Germany.*

http://www.knowledgemedia.org/netacademy/publications.nsf/all_pk/1318, 1999. [21, 252]

[ZTC03] ZTC. ZTC - A Type Checker for Z Notation. 2003. url: http://se.cs.depaul.edu/fm/ztc.html. [43, 92, 246]

APÊNDICE A

Notação Z - Um Resumo e Esquemas Genéricos

A.1 Esquemas Genéricos

Neste apêndice são apresentados alguns esquemas genéricos que definem as funções *mapseq*, *mapset*, *trifirst*, *trisecond* e *trithird*. As funções *mapseq* e *mapset* são apresentadas em [dL96] e são descritas neste trabalho apenas para facilitar a leitura das especificações contidas nesta tese. A função *mapseq* recebe uma função e uma seqüência e aplica a função a cada elemento da seqüência. A função *mapset* recebe uma função e um conjunto e aplica a função a cada elemento do conjunto. Adicionalmente, são definidas também as três funções genéricas *trifirst*, *trisecond* e *trithird* que são aplicadas a triplas e permitem a recuperação, respectivamente, do primeiro, segundo e terceiro elementos da tripla.

$[X, Y]$

$$mapseq : (X \pfun Y) \fun \seq X \pfun \seq Y$$
$$mapset : (X \pfun Y) \fun \power X \pfun \power Y$$

$$\forall seqs : \seq X;\ xs : \power X;\ fun : X \pfun Y \bullet$$
$$mapseq\, fun\, seqs = \{n \in \nat \mid n \in 1 \upto \#seqs \bullet (n, fun(seqsn))\} \land$$
$$mapsetfunxs = \{x : xs \bullet funx\}$$

$[X, Y, \mathbb{Z}]$

$$
\begin{array}{l}
trifirst : (X \times Y \times \mathbb{Z}) \rightarrow X \\
trisecond : (X \times Y \times \mathbb{Z}) \rightarrow Y \\
trithird : (X \times Y \times \mathbb{Z}) \rightarrow \mathbb{Z} \\
\hline
\forall x : X;\ y : Y;\ z : \mathbb{Z} \bullet trifirst(x, y, z) = x \\
\forall x : X;\ y : Y;\ z : \mathbb{Z} \bullet trisecond(x, y, z) = y \\
\forall x : X;\ y : Y;\ z : \mathbb{Z} \bullet trithird(x, y, z) = z
\end{array}
$$

A.2 Notação Z - Resumo

$p \wedge q$	conjunção lógica	$A \nrightarrow B$	função parcial
$p \Rightarrow q$	implicação lógica	$A \rightarrow B$	função total
$\forall X.q$	quantificação universal	dom R	domínio de relação
$x \in A$	x é membro de A	ran R	imagem de relação
$\varnothing$	conjunto vazio	seq A	seqüência finita
$A \subseteq B$	A subconjunto de B	in	subseqüência de uma seq.
$A \subset B$	A é subconjunto próprio de B	*head s*	primeiro el. de uma seq.
$A \times B$	produto cartesiano	*last s*	último el. de uma seq.
$A \cup B$	união de A e B	$\langle i, j, .. \rangle$	seqüência
$A \cap B$	intersecção de A e B	$\mathbb{P} A$	conjunto potência
$A \setminus B$	conjunto diferença entre A e B	$\mathbb{P}_1 A$	conjunto pot. não vazio
$\bigcup A$	união de elementos de A		
$\#A$	tamanho de conj. finito		

Convenções e definições:

a, b	identificadores
p, q	predicados
A, B	conjuntos
S	esquema antes de operação
S'	esquema após operação
ΔS	mudança de estado
ΞS	sem mudança de estado

Esquema vertical:

Nome

Declarações

Predicados

Tipo básico: $[T_1, ..., T_n]$

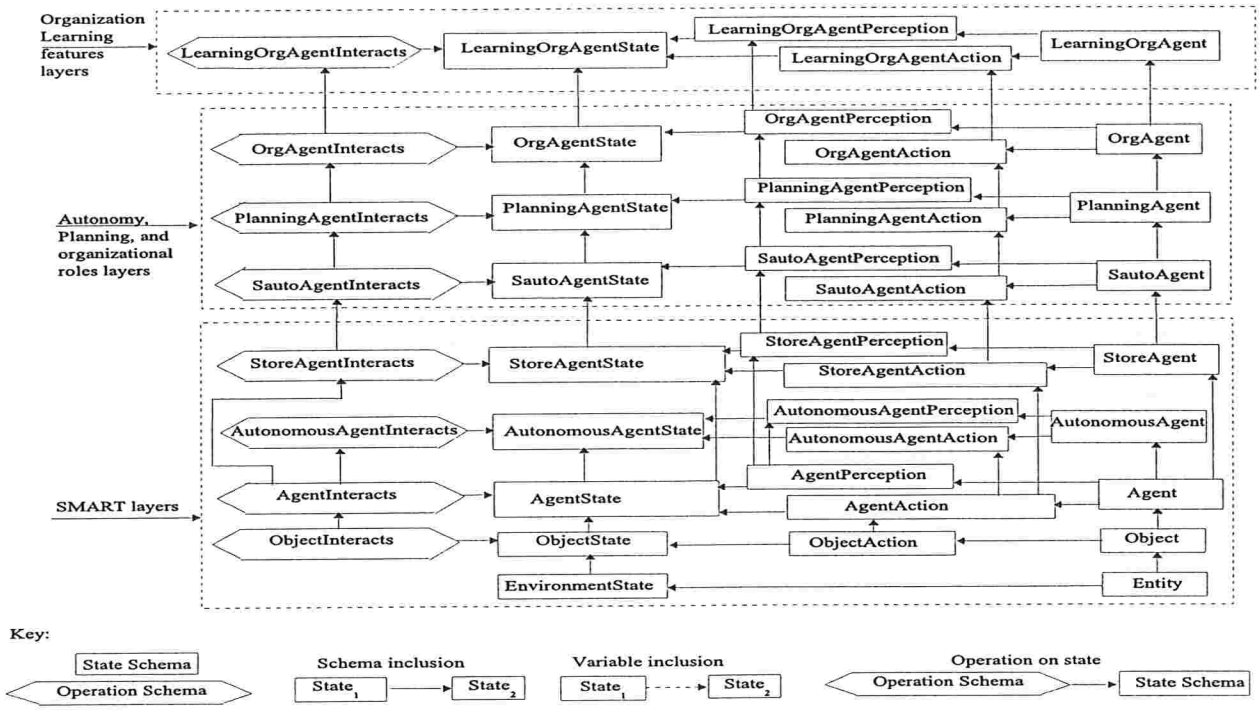
Definição abreviada: *identificador* == *Expressão*

Tipo construído:

$$
\begin{aligned}
T ::= & \; c_1 \mid ... \mid c_m \\
\mid & \; d_1 \langle\langle E_1[T] \rangle\rangle \\
\mid & \; ... \\
\mid & \; d_n \langle\langle E_n[T] \rangle\rangle
\end{aligned}
$$

Diagrama de Estrutura de Esquemas

O diagrama abaixo apresenta a estrutura básica de esquemas do modelo formal para a Quinta Disciplina.



Índice Remissivo