PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

FACULDADE DE INFORMÁTICA

PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ESPECIFICAÇÃO DE REQUISITOS DE AGENTES DE USUÁRIO EM Z

Andréa Aparecida Konzen

Dissertação apresentada como requisito parcial à obtenção do grau de mestre em Ciência da Computação.

Orientadora: Profa. Dra. Lucia M. M. Giraffa

Porto Alegre, novembro de 2002

Mãe, pai e Patricia,

dedico este trabalho a vocês.

Agradecimentos

A Deus, que me guiou da melhor forma possível, para que eu pudesse atingir meus objetivos.

À minha mãe, Helena, por ter confiado e por ter apostado na realização de mais um objetivo em minha vida. Devo parte dessa conquista a ti. Obrigada por todo apoio, amor, compreensão, preocupação (ás vezes exagerada, ☺), dedicação e por ter me ajudado em todos os momentos que necessitei. Tu és a melhor mãe do mundo!

Ao meu pai, Arno, que hoje não está mais nesse plano, mas tenho certeza que está feliz por esta conquista. Nunca vou esquecer quando me dizia que a "a maior herança que um pai pode dar ao seu filho é o estudo." E, para isso, sempre tive muito incentivo e apoio por parte dele e da mãe. Obrigadão!!!

À Patricia, minha mana querida. Obrigada pela amizade, carinho, apoio e, principalmente, por "quebrar" vários "galhos" para mim nesses dois anos.

Ao Pedro, pelo amor, carinho, amizade e apoio. Obrigada por ter me ajudado sempre quando precisei!! Sabes que tenho um carinho muito especial por ti.

À profa. Lucia Maria Martins Giraffa, minha orientadora, pela amizade, dedicação, atenção, ajuda, compreensão, carinho e palavras de consolo nos momentos difíceis. À você meu agradecimento especial.

Ao prof. Flávio Moreira Oliveira, orientador em 2001 e co-orientador em 2002. Obrigada pela dedicação, atenção, ajuda, compreensão, paciência, orientação e amizade. E, ainda por ter mantido as reuniões semanais sobre a dissertação durante o ano de 2002. Agradeço especialmente a você também.

Ao prof. Ricardo Bastos, pela disposição em ajudar quando foi necessário e pela oportunidade que me deu para fazer estágio de docência em uma de suas disciplinas.

Aos meus queridos colegas e amigos do PPGCC, Leticia Machado, Luiz Augusto Pizzato, Scheila Moreira Souza, Leonardo Castanheira, Eduardo Redhel, Luciano Cassol, Roberto Zanoni, Karina Benato, Leonardo Sewald, Alexandre Amory, Luciano Ramis, Cristiano Both, André Farina, Cristiano Meneguzzi, Cassiano Hess, Anete Espindola, Alberto Chemale, Caroline

Gasperin, Lucio Mauro Duarte, José Carlos Sant'Anna, João Carlos Furtado e Newton Müller. Pela amizade, pelos momentos de descontração, pelas trocas de conhecimento e convívio diário.

À Ir. Silvani agradeço especialmente pela amizade sincera, carinho, conversas, gargalhadas, conselhos e palavras de conforto nos momentos difíceis e energia positiva eu sempre me passou.

À minha amiga Liane um agradecimento especial pelo seu apoio, incentivo e sua amizade.

Às minhas eternas amigas Jaqueline Hoffman e Andréa Konrath, que mesmo estado distante, mostraram sempre sua verdadeira amizade e apoio.

À Ângela Fronckowiak, Ingrid Finger, Dulci Boettcher, Paulo Araújo e Lissi Bender Azambuja pelo incentivo que sempre me deram a fim de que o sonho de cursar mestrado fosse concretizado. Muito obrigada!!

À UNISC por ter concedido licença do meu trabalho para que eu pudesse cursar o mestrado.

Ao corpo docente do PPGCC, pela interação e disposição.

Aos funcionários do PPGCC, Dario, Leandro, Viviane e José Carlos pela disposição, ajuda e amizade.

Ao PPGCC-FACIN da PUCRS pelos recursos oferecidos.

À CAPES que tornou possível a realização dessa pesquisa, através do auxílio financeiro.

A todos aqueles que, direta ou indiretamente, contribuíram para que este trabalho fosse realizado.

Abstract

User agents, also called interface agents or personal assistants, have the goal of simplifying common tasks performed by users. This work presents a proposal for the specification of user agents, which includes the definition of the relevant properties for these agents, according to the role they play in a multiagent society. After that, we defined the requirements of each property. We present a methodology that defines the steps to be followed in order to specify the requirements associated to each basic property of user agents. The formal specification language used is based on Z in order to make it possible the migration of works with visual representation to a formal representation, such as the works developed by the MASP (Multi Agent System develoPment) research group, which focus on the UML (Unified Modeling Language) modeling, and representation based in mental states. In order to evaluate this work we applied it to a cooperative multiagent ITS (Intelligent Tutoring System) in the Web. In this application, the interface agent plays an essential role in the support of the interactions between the student and the environment.

Keywords: Multiagent Systems, User Agents, Software Engineering, Z Specification, Personalization, Autonomy, Learning.

Resumo

Os agentes de usuários, também chamados de agentes de interface ou assistentes pessoais, possuem o objetivo de simplificar as tarefas rotineiras realizadas por um usuário. Este trabalho apresenta uma proposta para especificação dos agentes de usuário, a qual inclui a definição das propriedades que definem e influenciam na arquitetura dos agentes e decorrem do papel que o agente possui na sociedade multiagente. A partir daí, determina-se os requisitos que competem a cada uma das propriedades definidas. Apresenta-se uma metodologia que esboça as etapas a serem seguidas para a realização da especificação dos requisitos associados às propriedades básicas dos agentes de usuário. A notação utilizada para especificação formal é baseada em Z, com o intuito de tornar possível a migração de trabalhos com representação visual para uma representação formal, tais como os trabalhos desenvolvidos no grupo de pesquisa MASP (*Multi Agent System develoPment*), os quais possuem enfoque em modelagem UML (*Unified Modeling Language*) e representação através de estados mentais. Para fins de avaliação buscou-se uma aplicação no domínio dos STI (Sistemas Tutores Inteligentes), envolvendo a idéia de um STI multiagente cooperativo na Web, onde o agente de interface possui um papel preponderante no auxílio às interações entre o aluno e o ambiente.

Palavras-chave: Sistemas Multiagentes, Agentes de Usuário, Engenharia de Software, Especificação Z, Personalização, Autonomia, Aprendizagem.

SUMÁRIO

SUMÁRIO ..... **VI**

LISTA DE FIGURAS ..... **VIII**

LISTA DE TABELAS ..... **X**

LISTA DE SÍMBOLOS E ABREVIATURAS ..... **XI**

1 INTRODUÇÃO **1**

1.1 CONTEXTO DO TRABALHO 2
1.2 MOTIVAÇÃO 3
1.3 CARACTERIZAÇÃO DO PROBLEMA 3
1.4 QUESTÃO DE PESQUISA 4
1.5 OBJETIVO GERAL 5
1.5.1 Objetivos específicos 5
1.6 CONTRIBUIÇÕES DO TRABALHO 5
1.7 ORGANIZAÇÃO DO VOLUME 7

2 REFERENCIAL TEÓRICO **8**

2.1 SISTEMAS MULTIAGENTES E AGENTES 8
2.2 AGENTES DE USUÁRIO 9
2.2.1 Propriedades dos agentes de usuário 12
2.2.1.1 Aprendizagem 13
2.2.1.2 Autonomia 13
2.2.1.3 Personalização 14
2.3 ENGENHARIA DE SISTEMAS MULTIAGENTES 15
2.3.1 Projeto de Sistemas Multiagentes 16
2.3.1.1 Aspectos fundamentais para o projeto de SMA 17
2.4 METODOLOGIAS DE PROJETO 25
2.4.1 Metodologia GAIA 25
2.4.2 Metodologia MAS-CommonKADS 26
2.4.3 Metodologia orientada a agentes para modelagem operacional 27
2.4.4 Metodologia MaSE 28
2.5 FORMALISMOS PARA SMA 30
2.5.1 Proposta de Souza 31
2.5.1.1 Diagrama de Casos de Uso em SMA 32
2.5.1.2 Diagrama da Sociedade 32
2.5.1.3 Diagrama de Interação Seqüencial 34
2.5.1.4 Interações entre os diagramas 35
2.5.2 Lógica Temporal de Ações (TLA) 36
2.5.3 Modelo de Dependência de Ator (DA) 38
2.5.4 Estados Mentais - BDI 40
2.5.5 Formalismo Z 44

3 ESPECIFICANDO AGENTES DE USUÁRIO **47**

3.1 REQUISITOS ASSOCIADOS ÀS PROPRIEDADES DOS AGENTES DE USUÁRIO 47
3.1.1 Aprendizagem 48

3.1.1.1 Requisitos sociais 48
3.1.1.2 Requisitos internos 48
3.1.2 Autonomia 49
3.1.2.1 Requisitos sociais 49
3.1.2.2 Requisito interno 49
3.1.3 Personalização 50
3.1.3.1 Requisitos sociais 50
3.1.3.2 Requisitos internos 50
3.2 APLICAÇÃO DO FORMALISMO 51
4 ESTUDO DE CASO 62
4.1 TRABALHO UTILIZADO 62
4.1.1 O Agente Diagnóstico 64
4.1.2 A interação entre os agentes 66
4.2 ESPECIFICAÇÃO DO AGENTE DIAGNÓSTICO 66
5 CONSIDERAÇÕES FINAIS 77
6 REFERÊNCIAS BIBLIOGRÁFICAS 81
ANEXO A 88

LISTA DE FIGURAS

Figura 1 - Esquema do contexto do trabalho 2
Figura 2 - Esquema da ligação do trabalho com outros 6
Figura 3 - Visão de um Sistema Multiagente 8
Figura 4 - Esquema de um agente 9
Figura 5 - Processo de projeto de um SMA [SIC95] 17
Figura 6 - Visão geral da metodologia MaSE 29
Figura 7 - Exemplo de Diagrama de Casos de Uso 32
Figura 8 - Exemplo de Diagrama da Sociedade [SOU01] 33
Figura 9 - Componentes do Diagrama de Interação Seqüencial [SOU01] 35
Figura 10 - Especificação do protocolo *Sender/Receiver* 37
Figura 11 - Estrutura de uma especificação em TLA+ 38
Figura 12 - Exemplo de Modelo de Dependência de Ator 39
Figura 13 - Arquitetura BDI 43
Figura 14 - Exemplo de um esquema em Z 45
Figura 15 - Esquema *Entidade* 53
Figura 16 - Esquema *Acoes* 53
Figura 17 - Esquema *Objeto* 53
Figura 18 - Esquema *Agente* 54
Figura 19 - Esquema *Agente de Usuário* 54
Figura 20 - Esquema *EstadoAmbiente* 55
Figura 21 - Esquema Δ*EstadoAmbiente* 56
Figura 22 - Esquema *Aprendizagem* 56
Figura 23 - Esquema *Autonomia* 56
Figura 24 - Esquema *Personalização* 56
Figura 25 - Esquema *AutonomiaUsuario* 57
Figura 26 - Esquema *EstadoUsuario* 58
Figura 27 - Esquema Δ*EstadoUsuário* 58
Figura 28 - Esquema *ObservarUsuario* 59
Figura 29 - Esquema *AcoesAgente* 59
Figura 30 - Esquema *AcoesUsuario* 59

Figura 31 - Organização dos esquemas 60
Figura 32 - Arquitetura multiagente do sistema [AND01] 62
Figura 33 - Esquema *Agente de Usuário* 69
Figura 34 - Esquema *Aprendizagem* 69
Figura 35 - Esquema *Autonomia* 70
Figura 36 - Esquema *Personalização* 70
Figura 37 - Esquema *EstadoZDP* 70
Figura 38 - Esquema Δ*EstadoZDP* 71
Figura 39 - Esquema *EstadoCore* 71
Figura 40 - Esquema Δ*EstadoCore* 71
Figura 41 - Esquema *ObservarUsuario* 72
Figura 42 - Esquema *Regiao* 73
Figura 43 - Esquema Δ*EstadoRegiao* 73
Figura 44 - Esquema *EstadoDiagnostico* 74
Figura 45 - Esquema Δ*EstadoDiagnostico* 74
Figura 46 - Estruturação dos esquemas 74
Figura 47 - Exemplo de agrupamento de esquemas 75

LISTA DE TABELAS

Tabela 1 - Diagramas x Aspectos fundamentais de SMA 31

Tabela 2 - Definição dos tipos utilizados no formalismo 60

Tabela 3 - Definição dos símbolos utilizados no formalismo 61

Tabela 4 - Definição dos tipos utilizados no estudo de caso 76

LISTA DE SÍMBOLOS E ABREVIATURAS

ACL	*Agent Communication Language*
ADEPT	*Advanced Decision Environment for Process Tasks*
AEF	Agente Entidade Funcional
ARCHON	*ARchiteture for Cooperative Heterogeneous ON-line systems*
AUML	*Agent Unified Modeling Language*
BDI	*Belief, Desire and Intention*
CIMOSA	*Computer Integrated Manufacturing Open System Architeture*
DA	*Dependency Actor*
FIPA	*Foundation Intelligent Physical Agents*
IA	Inteligência Artificial
IAD	Inteligência Artificial Distribuída
IDEF	*Integration DEfinition for Function modeling*
IRMA	*Intelligent Resource-bounded Machine Architeture*
KQML	*Knowledge Query Manipulation Language*
LISP	*LISt Processing*
MaSE	*Multiagent Systems Engineering*
MASP	*Multi Agent System Development*
OA	Orientado a Agente
OOSE	*Object-Oriented Software Engineering*
PGP	Planejamento Global Parcial
SMA	Sistema Multiagente
STI	Sistema Tutor Inteligente
TLA	*Temporal Logic of Actions*
UML	*Unified Modeling Language*
X-BDI	*eXecutable Belief, Desire and Intention*
ZDP	Zona de Desenvolvimento Proximal

1 INTRODUÇÃO

Um tópico muito discutido nos últimos anos na pesquisa em IA (Inteligência Artificial) tem sido a adoção da abordagem de SMAs (Sistemas Multiagentes) para diferentes aplicações. Pelo fato dos SMAs possuírem características (propriedades) que os diferem de sistemas tradicionais, muitos trabalhos têm apresentado propostas de formalizações, metodologias, conceitualizações, técnicas, métodos e protocolos para a aplicação desse tipo de abordagem na concepção de diferentes sistemas.

A investigação acerca de sistemas interativos que se adaptam ao usuário é outro aspecto importante a ser considerado quando se trabalha com SMAs. No entanto, o processo de desenvolvimento em agentes de usuário, mais precisamente em especificação desses agentes, conforme estudos realizados ao longo dos trabalhos individuais realizados por [KON01a] e [KON01b], mostraram que existem muitas questões ainda em aberto. Com estes resultados disponíveis e os resultados existentes no projeto MASP[1](*Multi Agent System develoPment*), verificou-se a necessidade de especificar os requisitos das propriedades básicas dos agentes de usuário, através de uma linguagem de especificação formal.

Logo, foi necessário definir-se quais seriam as propriedades básicas desses agentes e selecionar um formalismo robusto que permitisse integrar, através de especificação formal, os resultados prévios do grupo (como resultado prévio podemos citar a modelagem BDI e utilização de AUML (*Agent Unified Modeling Language*) na especificação de SMAs).

As três propriedades de um agente de usuário que foram definidas como básicas neste trabalho são: a aprendizagem, a autonomia e a personalização. Assim, salienta-se que estas propriedades possuem grande importância, pois representam as principais funcionalidades dos agentes de usuário num sistema, no contexto desse trabalho [KON02].

Acredita-se que os aspectos sociais e internos (arquitetura) dos agentes devem ser tratados separadamente, devido ao conjunto de diferenças envolvendo sua modelagem e projeto. Considerando os aspectos dos agentes de usuário, Souza [SOU01] trata a questão somente no nível da sociedade de agentes (aspectos sociais), podendo eventualmente ser estendido para os

[1] MASP (*Multi Agent System Development*): este projeto tem como finalidade envolver a especificação de linguagens de modelagem para sistemas multiagentes e um conjunto de atividades a serem desenvolvidas para a construção de um sistema baseado neste paradigma.

agentes de usuário. Logo, como o modelo de [SOU01] não engloba os aspectos internos desses agentes, buscou-se alternativas diferentes da proposta original apresentada em [KON01c].

O conjunto de estudos realizados apontou para a necessidade de especificar os requisitos das propriedades básicas dos agentes de usuário. Dessa forma, após levantamento de alguns formalismos, identificou-se um formalismo que mais se adequa à necessidade de especificação dos agentes de usuário considerando o contexto do projeto MASP.

1.1 Contexto do trabalho

O contexto deste trabalho de pesquisa é apresentado na Figura 1.

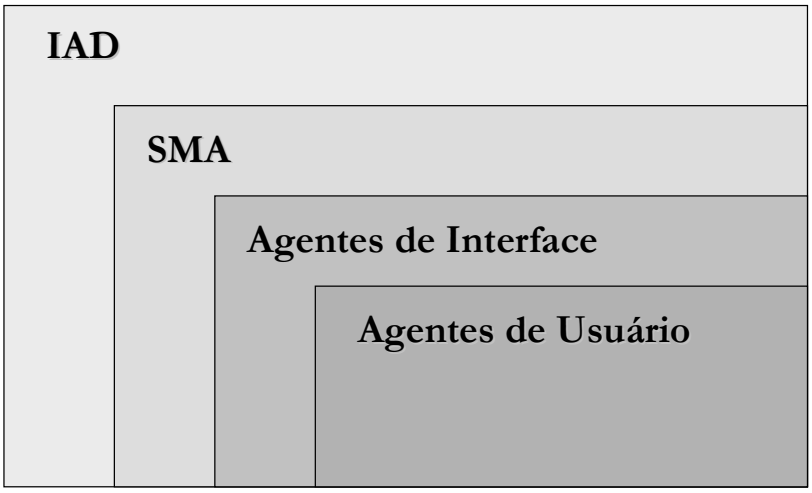


Figura 1 - Esquema do contexto do trabalho

O foco principal desse trabalho se concentra na tecnologia de agentes, mais precisamente na especificação de agentes de usuário, o qual está inserido como sub-área de agentes de interface.

O grupo MASP, do PPGCC (Programa de Pós-Graduação em Ciência da Computação) da PUCRS (Pontifícia Universidade Católica do Rio Grande do Sul), está desenvolvendo pesquisas com o intuito de envolver a especificação de uma linguagem de modelagem para sistemas multiagentes e o conjunto de atividades a serem desenvolvidas para a construção de um sistema baseado neste paradigma.

Nesta pesquisa, busca-se unir trabalhos do grupo com outros enfoques em termos de modelagem, através de uma nova abordagem no que se refere aos agentes de usuário.

1.2 Motivação

Este trabalho teve origem a partir da investigação dos trabalhos de Goulart [GOU01] e Souza [SOU01]. Analisando-se o trabalho de [GOU01] viu-se a possibilidade de aprimorar estudos sobre os agentes de usuário. E, tendo como base o trabalho de [SOU01], que trata da questão da modelagem de SMA, decidiu-se modelar/especificar os agentes de usuário, tomando como ponto de referência a modelagem de agentes de usuário sob uma visão mais formal.

Para tal, foi necessário explorar mais detalhadamente os agentes de usuário, com enfoque maior em suas propriedades, baseando-se no fato de que cada vez mais existe a necessidade de aplicar a tecnologia desses agentes nos ambientes de aprendizagem associados a modelagem do aluno/usuário.

O maior detalhamento das propriedades dos agentes de usuário possibilita a união de trabalhos do grupo de pesquisa MASP através de uma notação formal já que existem trabalhos que utilizam AUML e representação através de estados mentais (BDI – *Belief, Desired and Intention)*. Dessa forma, busca-se contribuir com pesquisa em desenvolvimento, no projeto MASP, através de uma representação diferenciada das usadas habitualmente para especificar agentes.

1.3 Caracterização do problema

Em trabalhos preliminares de Goulart [GOU01], Konzen [KON01b] e Souza [SOU01] constatou-se que o projeto e desenvolvimento em agentes de usuário, mais precisamente a questão da especificação de agentes de usuário, ainda é um tópico em aberto na área e, sem consenso por parte dos pesquisadores. Ou seja, pode-se dizer que ainda persistem problemas nessa questão, tanto no nível de aplicações educacionais, quanto para aplicações em outros domínios.

Utilizando como referência o trabalho de Souza [SOU01], observou-se a necessidade de considerar aspectos sociais e internos dos agentes de usuário, no momento da definição dos requisitos associados às propriedades básicas. Também, houve a necessidade, a partir daí, de definir-se as propriedades que devem ser consideradas como básicas neste trabalho.

Como conseqüência disso, existe a necessidade de especificar esses agentes, de modo que seja de fácil compreensão e que torne possível a migração e a utilização em outros trabalhos já existentes no nosso grupo de pesquisa MASP.

Como o grupo de pesquisa em SMAs do PPGCC da PUCRS possui uma linha investigativa envolvendo questões sobre a direção de uma metodologia que permita especificar e modelar SMAs e tendo como um de seus estudos de caso aplicações educacionais (STI multiagentes), este problema foi identificado como relevante dentro do contexto do projeto e foi adotado para servir de base para este trabalho de mestrado.

1.4 Questão de pesquisa

Levando em consideração a caracterização e o escopo do problema acima descrito adotou-se como questão de pesquisa norteadora desse trabalho:

Quais propriedades básicas do agente de usuário devem ser consideradas quando da definição de uma linguagem para especificação formal de agentes?

Associadas a esta questão, consideram-se as seguintes hipóteses:

Hipótese 1: As propriedades básicas de um agente de usuário são a personalização, a autonomia e a aprendizagem.

Hipótese 2: As propriedades de personalização, autonomia e aprendizagem englobam aspectos sociais e internos dos agentes.

Hipótese 3: Uma linguagem de especificação de agentes de usuário deve permitir a descrição das propriedades de personalização, autonomia e aprendizagem.

Originalmente, pretendia-se utilizar o trabalho de Souza [SOU01] e estender o seu modelo, para atender às questões envolvendo o agente de usuário. Em vez disso, optou-se por complementar o modelo visual com uma notação não visual. Pois, a partir da especificação dos requisitos das propriedades em Z como a notação possui esquemas que são de fácil mapeamento para classes, pode-se estender o modelo. Ou seja, a partir de uma especificação formal traduz-se para um modelo mais visual. Aproveita-se para salientar que o modelo de Souza [SOU01] é interessante e de grande importância para a pesquisa no grupo MASP, porém, a utilização de Z traria maior diversidade para os trabalhos associados ao projeto.

1.5 Objetivo geral

Identificar e aplicar um formalismo para especificar os requisitos de agentes de usuário, considerando as propriedades básicas definidas neste trabalho: aprendizagem, autonomia e personalização.

1.5.1 Objetivos específicos

- Desenvolver um estudo para identificar as propriedades básicas de um agente de usuário;
- Estudar os aspectos internos dos agentes a fim de especificar formalmente para agentes de usuário;
- Desenvolver um estudo para identificar um formalismo a fim de especificar agentes de usuário, que permita utilizar os resultados já existentes no grupo de pesquisa onde este trabalho está inserido;
- Definir os requisitos das propriedades básicas a fim de especificá-los através de uma notação;
- Definir uma metodologia inicial para a realização da especificação dos agentes de usuário;
- Definir um formalismo para especificar os requisitos das propriedades básicas dos agentes de usuário;
- Aplicar o formalismo proposto em um estudo de caso na área de STI, tomando como ponto de partida, o trabalho proposto por Andrade [AND02], a fim de validar sua aplicabilidade;
- Inserir o trabalho dentro do contexto do projeto MASP, da PUCRS, no que se refere ao projeto e desenvolvimento de sistemas multiagentes. Esta inserção será realizada via estudo de caso.

1.6 Contribuições do trabalho

Como contribuições deste trabalho, consideram-se:

- A definição de um conjunto de propriedades básicas para agentes de usuário;
- A definição de requisitos associados às propriedades básicas dos agentes de usuário;
- A identificação de um formalismo que permita expressar os requisitos das propriedades básicas dos agentes de usuário incorporando resultados já consolidados na área de SMAs;
- A extensão do modelo proposto por Luck [LUC01];
- No contexto do projeto MASP, a utilização de uma nova alternativa, ou seja, a utilização de especificação formal para modelar agentes através do formalismo Z como opção ao trabalho de [MOR98] (BDI - *eXecutable Belief Desire Intention*), possibilita a extensão de modelos já existentes no contexto do projeto e a especificação de trabalhos com enfoque em estados mentais.

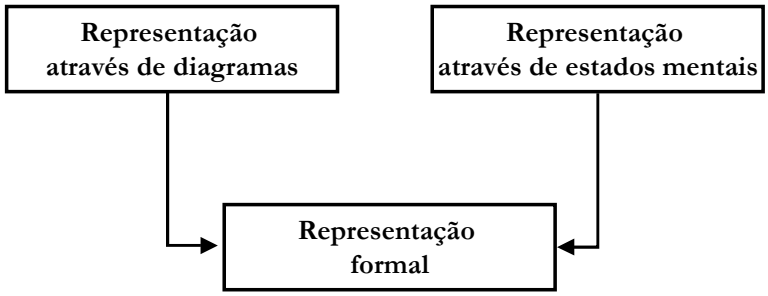


Figura 2 – Esquema da ligação do trabalho com outros

A Figura 2 expressa a ligação deste trabalho com os demais trabalhos existentes no projeto MASP, o qual tem como finalidade envolver a especificação de uma linguagem de modelagem para sistemas multiagentes e o conjunto de atividades a serem desenvolvidas para a construção de um sistema baseado neste paradigma. Este projeto é constituído por um conjunto de trabalhos baseados em representação visual. Ou seja, que apresentam a modelagem e especificação dos agentes com a utilização de AUML, como o trabalho proposto por [SOU01] e, por trabalhos baseados em estados mentais, como por exemplo o trabalho de [ZAM01]. Dessa forma, a proposta de especificação formal realizada, pode tornar possível a migração (para essa especificação) desses trabalhos e outros trabalhos do projeto MASP que seguem na mesma linha;

Assim, pode-se dizer que a especificação formal selecionada é de fácil adaptação devido aos seguintes motivos:

- Considerando a representação visual (representação utilizando AUML, por exemplo) a notação Z possui esquemas que são semelhantes aos diagramas em AUML, o que facilita o mapeamento desses para classes;

- Os trabalhos do grupo utilizam uma arquitetura BDI a qual se baseia na descrição do processamento interno do agente utilizando um conjunto básico de estados mentais (crenças, desejos e intenções), juntamente com lógica multimodal. Esses conceitos também são adaptáveis a especificação Z, onde, podem ser expressos conjuntos de crenças, desejos e intenções para especificar um agente, utilizando lógica de 1ª ordem.

Estes resultados podem ser utilizados futuramente para estender outros projetos que desenvolvem pesquisas envolvendo especificação de agentes.

1.7 Organização do volume

Este trabalho está estruturado em 5 capítulos. O segundo capítulo apresenta o referencial teórico, onde são apresentados conceitos e considerações acerca dos sistemas multiagentes e agentes. Os agentes de usuário e suas propriedades são detalhados sob o ponto de vista de Engenharia de Sistemas Multiagentes, são apresentadas algumas metodologias de projeto e, alguns formalismos para sistemas multiagentes.

No terceiro capítulo é apresentada a especificação dos agentes de usuário, através dos requisitos associados às suas propriedades e a aplicação do formalismo por meio de esquemas em Z.

No quarto capítulo mostra-se o estudo de caso realizado, onde se explica detalhadamente o trabalho utilizado e expõem-se os esquemas aplicados sobre este trabalho. No quinto capítulo apresentam-se as considerações finais desse trabalho e, ao final, são listadas as referências bibliográficas utilizadas. Como anexo a este volume, apresenta-se um detalhamento sobre as definições em Z, utilizadas nesse trabalho.

2 REFERENCIAL TEÓRICO

2.1 Sistemas Multiagentes e Agentes

A IA (Inteligência Artificial), desde sua criação, tem buscado compreender os mecanismos fundamentais que regem o comportamento inteligente. Nos últimos anos, tem-se dedicado grandes esforços à IAD (Inteligência Artificial Distribuída), um subcampo da IA que trata de uma sociedade de resolvedores de problemas ou agentes que interagem entre si de forma a resolver um problema comum. Esta sociedade é comumente denominada de Sistema Multiagente. Ou seja, uma rede de solucionadores de problemas que trabalham coletivamente além das capacidades individuais de cada um.

Nos SMAs investiga-se o comportamento de um conjunto de agentes autônomos, possivelmente, pré-existentes, que interagem objetivando a resolução de um problema que está além da capacidade de um único indivíduo [JEN96]. Dessa forma, o comportamento global do sistema deriva da interação entre os agentes que fazem parte do sistema [ZAM01]. A partir disso, está envolvida a busca por uma funcionalidade neste sistema que permita que estes agentes possam coordenar seus conhecimentos, objetivos, habilidades e planos individuais de uma forma conjunta, em favor da execução de uma ação ou da resolução de algum problema. A cooperação entre os agentes é necessária tanto para a ação quanto para a resolução de problemas em SMA. Nesses casos, pode-se dizer que o agente exibe um comportamento social [ZAM01].

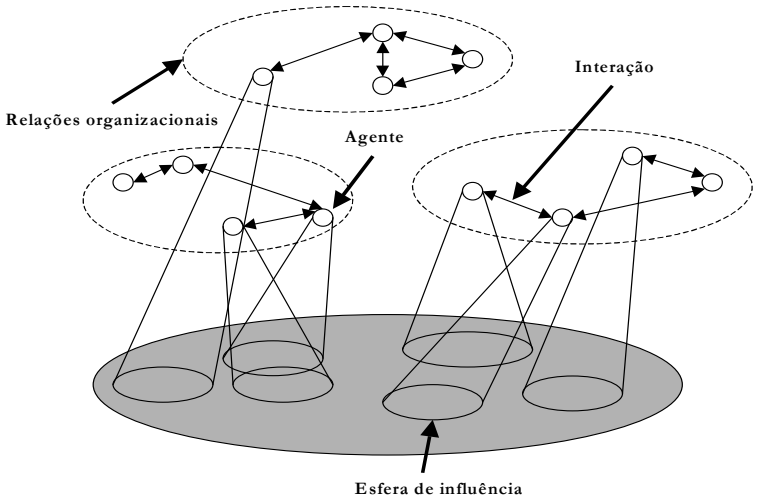


Figura 3 - Visão de um Sistema Multiagente

Para o conceito de agente, Wooldridge [WOO02] determina que: o principal propósito dos agentes é eles serem autônomos. Ou seja, capazes de agir independentemente, demonstrando controle sobre seu estado interno. Assim, um agente é um sistema de computador capaz de agir autonomamente com flexibilidade (ser reativo, pró-ativo e social) em algum ambiente, conforme Figura 4, baseada na figura apresentada em [WOO02].

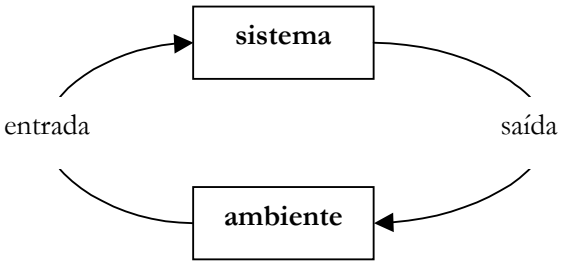


Figura 4 - Esquema de um agente

As teorias de agentes são essencialmente especificações, de como se concebe um agente, quais as suas propriedades, como eles são representados formalmente, entre outros aspectos.

As linguagens de agentes são linguagens que trabalham questões relativas a programação de agentes, considerando quais são as primitivas corretas para o projeto, como por exemplo, executar de maneira eficiente os programas de agente.

Com a evolução dos sistemas e o crescimento da necessidade de se adaptar sistemas ao perfil do usuário, a fim de que ele possa otimizar seu trabalho e, além disso, com a evolução da pesquisa em SMAs, busca-se aprimorar sistemas utilizando a tecnologia de SMAs e, assim, surge a necessidade de empregar agentes de usuário, a fim de personalizar sistemas.

2.2 Agentes de usuário

Para caracterizar um sistema como sendo adaptativo é necessário que um sistema tente modificar o seu comportamento para maximizar a produtividade da atual interação do usuário com o sistema. Dessa forma, é necessário que seja baseada em informação reunida sobre o comportamento atual do usuário. De acordo com isso, Nogueira [NOG00] aborda em seu trabalho três questões interessantes:

- Quais aspectos do comportamento do usuário são úteis para capturar?

- Como um comportamento transmite informação sobre as atuais intenções e preferências do usuário?
- Qual o uso que pode ser feito desta informação capturada?

Pode-se dizer que, o conceito de agentes de usuário ainda não está consolidado. Em diferentes grupos de pesquisa, para se definir um agente, diferentes características podem se fazer necessárias, isto leva a uma grande variedade de definições e características. Dessa forma, para este trabalho, adotou-se que os agentes de usuário podem ser classificados em assistentes pessoais e agentes de interface, sendo que os agentes tratados neste são considerados como assistentes pessoais. Assim, a seguir, são apresentadas definições dos agentes de usuário, dadas por algum autores:

Conforme [NOG00], os agentes de interface são um desenvolvimento relativamente recente e usam uma abordagem baseada em agentes para a construção de sistemas. O principal aspecto que distingue os agentes de interface de outros agentes é o fato de serem pró-ativos e possuírem de um certo grau de autonomia, o qual podem se manifestar de diversas formas.

Conforme Maes [MAE94], agentes de usuários possuem o objetivo de simplificar as tarefas rotineiras realizadas por um usuário. Esse tipo de agente observa e monitora as ações tomadas pelo usuário na interface, aprende novos "atalhos" e sugere melhores formas de desenvolver as tarefas. O agente de interface age como um assistente pessoal autônomo o qual coopera com o usuário na realização de determinadas tarefas. Ele aprende e cria o perfil do usuário basicamente de quatro formas:

- Observando e imitando o usuário;
- Mediante recebimento de um *feedback* positivo ou negativo por parte do usuário;
- Recebendo instruções explícitas do usuário;
- Com ajuda de outros agentes.

Os agentes de usuário necessitam conhecer o problema e/ou o usuário. Assim, estes devem apresentam algumas características fundamentais:

- Podem obter o conhecimento acerca do problema solicitando informações ao usuário ou percebendo variações no ambiente, através da comunicação com outros agentes;

- Devem ser capazes de analisar um conjunto de possíveis soluções para atingir seus objetivos, dependendo de sua capacidade de raciocínio;

- Devem possuir capacidade de tomar decisões para selecionar a melhor forma de solucionar o problema.

Middletor [MID02] considera os agentes de usuário como instâncias de classe que os usuários criam e personalizam, atribuindo tarefas específicas devem ser capazes de receber mensagens de outros agentes, avaliar se é possível ou não atendê-las e respondê-las. Esses agentes também podem iniciar uma comunicação enviando mensagens a fim de solicitar serviços para os usuários.

Os agentes de usuário podem apresentar modelos sobre perfis de usuários. Estes perfis ajudam a compreender as diferenças individuais e a definir interfaces com usabilidade adequada para grupos de indivíduos com características semelhantes. Devido à complexidade de representação das características do ser humano, existem inúmeros perfis para refletir o conjunto de características diferentes dos usuários [WIN00].

Um agente de usuário pode ser descrito como sendo um robô (*softbot*), que funciona como um assistente pessoal que poderá realizar tarefas solicitadas pelo usuário. O *softbot* pode realizar inferências sobre as informações obtidas e a partir daí, traçar um plano a fim de melhor realizar suas tarefas. Este *softbot*, também, deve ser capaz de tratar a ambigüidade, omissão e erros nos dados fornecidos pelo usuário [ETZ94].

Um agente de software (agente de usuário) pode ser descrito como um processo de execução independente, escolhendo autonomamente as ações, quando um evento esperado ou inesperado surge, sem a necessidade de ações externas efetuadas por um usuário. Estes agentes podem ter propósitos diversos, mas tipicamente possuem as capacidades necessárias para realizar as tarefas no ambiente. Conforme Silva [SIL99] e Jennings [JEN98a], esta tecnologia tem vindo a revelar a sua utilidade em diversas áreas de aplicabilidade:

- Aplicações de carácter lúdico, como os jogos interativos, por exemplo pode-se apresentar o *RoboCup* (competição internacional de futebol entre *robots* controlados por agentes de software);

- Área militar ou proteção civil, para simulação de situações de combate, desastres naturais ou outros;

- Sistemas de suporte ao ensino, como exemplo, pode-se mostrar um agente que descreve um procedimento, responde a perguntas e guia o aluno na execução desse procedimento;
- Aplicações de busca inteligente de informação na Internet, o qual tem sido um campo fértil, tanto em investigação, como em aplicações comerciais;
- Outras, tais como: gestão de redes e de serviços, comércio eletrônico, gestão de informação pessoal.

A investigação a respeito dos sistemas interativos que se adaptam ao usuário é um aspecto importante a ser considerado quando se trabalha com agentes de usuário. Pode-se dizer que nas áreas de Comércio Eletrônico e acesso a informações na Web esta investigação tem aumentado significativamente e obtido grande importância. Segundo Jamerson [JAM01], as tarefas relevantes nesse contexto são: personalização, criação de assistentes pessoais, aplicação de agentes de usuário à hipermídia adaptativa, interfaces adaptativas relacionadas com agentes, modelagem de estudante/usuário com finalidade pedagógica (considerando aplicações educacionais).

2.2.1 Propriedades dos agentes de usuário

Para definir-se as propriedades básicas dos agentes de usuário (aprendizagem, autonomia e personalização), toma-se como ponto de referência as definições dadas na literatura por autores que trabalham com esses agentes, focando as propriedades diretamente para os agentes de usuário, e por autores que trabalham com agentes em geral. Assim, apresentam-se definições clássicas das propriedades.

Como propriedades dos agentes de usuário, para o escopo desse trabalho, apresentam-se três selecionadas, que se considera como básicas: aprendizagem, autonomia e personalização.

Estas propriedades são consideradas como básicas pelos seguintes motivos: a principal tarefa de um agente de usuário é personalizar o conteúdo ao usuário; não existindo a personalização, não é necessária a existência desse agente. Para que esta personalização aconteça, é fundamental que o agente de usuário aprenda as tarefas e, assim, adapte às solicitações do usuário, acontecendo a aprendizagem. E a propriedade de autonomia, por consenso de vários autores, já é definida como básica para qualquer agente e aqui reforçada no contexto da interação do agente com o usuário, pois os agentes devem ser, de certa forma, autônomos para realizar suas tarefas.

2.2.1.1 Aprendizagem

Considerando a colocação de Foner [FON97], todos os agentes de usuário devem possuir a propriedade de aprendizagem, pois primeiramente, acontece o aprendizado da tarefa para posteriormente, esta ser desenvolvida. Contudo, alguns autores não salientam em seus trabalhos essa propriedade.

Conforme Wooldridge [WOO02], sensores constituem a maneira com que o agente recolhe dados sobre o ambiente e os atuadores que utiliza são a forma do agente manipular o ambiente. Assim, em um ambiente de *software*, sensores e atuadores podem não ter significados físicos mas a idéia de que um programa pode ter a habilidade de sentir e manipular é aplicável e cada vez mais difundida. Os efeitos de um sistema sobre seu ambiente podem ser de vários graus, mas o aspecto importante da aprendizagem é o sistema poder sentir e entender as mudanças que ele está fazendo e reagir de maneira diferenciada baseado no que ele observou.

Assim, para este trabalho definiu-se que, com a propriedade de aprendizagem o agente de usuário possui a capacidade de executar uma tarefa com maior eficiência do que em execuções anteriores, ou seja, a cada execução o agente de usuário armazena características importantes que vão surgindo e estas são aplicadas a cada realização de tarefas semelhantes às anteriores.

A aprendizagem também pode ser realizada da seguinte forma: o agente de usuário observa o ambiente ou o usuário, conseqüentemente entende esse ambiente/usuário, e a partir daí entra a propriedade de personalização (detecção e busca de mudanças). Assim o agente modifica seus compromissos em relação ao ambiente/usuário, melhorando assim a realização de suas tarefas frente ao usuário.

2.2.1.2 Autonomia

Para a propriedade de autonomia tem-se como ponto de partida inicial a definição clássica dada por Wooldridge [WOO02] dizendo que o agente opera sem a intervenção humana e possui algum tipo de controle sobre suas ações e seu estado interno.

Conforme Foner [FON97], os agentes detentores de alto grau de autonomia podem planejar suas ações independentemente daquelas de seus usuários. Dessa forma, apresentam como aspectos básicos ações periódicas, execução espontânea e iniciativa.

Considerando que o projeto está focado na modelagem/especificação de agentes de usuário, salienta-se a definição de Eric Yu [YU01], o qual afirma que na autonomia o agente tem

iniciativa e pode atuar independentemente, o que é definido como um comportamento não totalmente previsível na modelagem, assim os agentes não são totalmente conhecíveis nem controláveis.

Primeiramente, coloca-se que os agentes podem possuir diferentes graus de autonomia. Conforme o grau de autonomia que o usuário concede ao agente ele pode ou não fazer determinadas tarefas/atividades, isso inclui ser ou não ser benevolente e atender ou não a solicitações de outros agentes.

Para contextualizar o grau de autonomia de usuário nesse trabalho, partiu-se do seguinte pressuposto: todo agente autônomo é pró-ativo, ou seja, possui a capacidade de elaborar planos e, assim, pode-se considerar que esta é uma característica interna do agente. O agente também precisa ser independente, assim este apresenta um aspecto social.

Em agentes de usuário considera-se como definição para a propriedade de autonomia escolher a ação que esse vai executar baseando-se principalmente, na experiência e conhecimento que já adquiriu através de interações anteriores e também no conhecimento básico que possui (entendemos por conhecimento pré-definido no agente no momento da implementação).

2.2.1.3 Personalização

Segundo Foner [FON97], na personalização o agente deve agir de forma diferente com usuários diferentes, sendo que o agente deve ser capaz de personalizar tarefas e auxiliar os usuários a desenvolvê-las da melhor forma possível. O enfoque de Foner baseia-se em: para que a personalização aconteça, primeiramente deve acontecer o aprendizado da tarefa e, posteriormente esta deve ser desenvolvida, como citado anteriormente na seção 2.2.2 na propriedade de aprendizagem.

Foner [FON97] considera, também, como uma característica importante a capacidade dos agentes de usuário de ignorar pedidos de usuários que se tornam desagradáveis em algum momento da interação. Salienta-se que a tarefa de selecionar pedidos "desagradáveis" é complexa, pois o que pode ser muito importante para o usuário pode representar para o agente o oposto.

Conforme Winckler [WIN00] personalização é a tendência a atribuir características similares às humanas, tais como, emoções e sofisticadas capacidades de planejamento para animais e objetos. A base dessas atribuições é freqüentemente chamada de intencionalidade, onde, fala-se dessas coisas (animais e objetos) como se elas pudessem formar intenções.

Analisando esse conceito pode-se dizer que se defende um conceito que contrapõe esse, o qual será referenciado a seguir.

Levando em consideração as definições desses autores, pode-se resumir a propriedade de personalização, para o contexto desse trabalho, como a habilidade que um agente de usuário tem de "conhecer" o indivíduo (usuário), ou seja, manter informação sobre o usuário. Dessa forma, o agente deve concentrar-se em várias características como objetivos, hábitos, interesses, preferências e necessidades do usuário para que possa atendê-lo e adequar o conteúdo/site da melhor forma possível.

2.3 Engenharia de Sistemas Multiagentes

Em Engenharia de Software, a construção de *software* de alta qualidade de maneira produtiva é viabilizada por um conjunto de métodos, ferramentas e procedimentos. O caminho para a evolução no desenvolvimento de *software* passa por uma combinação de métodos abrangentes para todas as fases de desenvolvimento do *software*, melhores ferramentas para automatizar estes métodos, blocos de construção mais poderosos para a implementação do *software*, melhores técnicas para a garantia da qualidade do *software* e uma filosofia de coordenação predominantemente, controle e administração [PRE95].

Conforme Jennings [JEN99] inúmeros trabalhos apresentam conceitualizações, formalizações, protocolos, técnicas e métodos para aplicação da abordagem multiagentes para resolução de problemas na concepção de *software*. Isso se deve ao fato dessa abordagem possuir algumas características que viabilizam a resolução de problemas adequando-se aos mais complexos. Ou seja, uma forma considerada como não tradicional.

Um agente pode ser visto como um *software* dotado de um conjunto de propriedades (autonomia, reatividade, habilidade social, comunicação, etc.) que lhe permitam desempenhar o seu papel e objetivos dentro de um sistema composto por outros agentes e/ou processos. Dependendo do papel/objetivo dos agentes dentro do ambiente onde estão inseridos, esses terão ou não certas propriedades.

Na literatura encontram-se os agentes classificados segundo duas abordagens: reativos e racionais (cognitivos). Cada abordagem possui um conjunto de pressupostos que orientam a especificação e funcionalidades associadas a cada agente. Na abordagem reativa salientam-se os trabalhos de [BRO87], [DEM90] e [DRO00]. Na abordagem cognitiva destacam-se os trabalhos de [COH87], [SIC95], [SHO93] e [JEN98a].

O surgimento da tecnologia de agentes representa uma abordagem promissora para desenvolver aplicações complexas, pois propõe um importante avanço em matéria de abstração, permitindo uma compreensão, modelagem e desenvolvimento mais naturais em relação a outros paradigmas. Entretanto, um ponto fundamental para o sucesso de um projeto utilizando a tecnologia de agentes é a seleção de uma metodologia apropriada.

Conforme Wooldridge [WOO00b], um sistema baseado em agentes pode ser considerado uma sociedade ou organização artificial. A idéia de comparar um sistema multiagente a uma sociedade é natural e possibilita chegar-se no próximo nível da hierarquia de conceitos da metodologia, que são os papéis.

O pesquisador Müller [MUL97] afirma que a tecnologia de SMAs está migrando da pesquisa de laboratório para os centros de engenharia de *software*. Se esta tecnologia contribui para o aumento do potencial dos SMAs, esta terá um crescimento importante para o desenvolvimento e emprego de metodologias, acessibilidade para engenharia de *software*, especificação, projeto e especificação de sistemas multiagentes.

Segundo Wooldridge e Jennings [WOO99], é necessário que uma metodologia capture aspectos como: a flexibilidade do agente, o seu comportamento autônomo, as interações e a complexidade das estruturas organizacionais nas quais os agentes estão inseridos.

2.3.1 Projeto de Sistemas Multiagentes

Segundo Sichman [SIC95] e Monard [MON00] o processo de projeto dos SMAs possui três momentos relevantes: a concepção, o projeto e a resolução do problema. Conforme a Figura 5, depois que os agentes são concebidos é ativada a fase de projeto. Na fase de projeto, a concepção das interações, a concepção dos agentes e a concepção das organizações podem ocorrer em paralelo ou não, pelo motivo da área de IAD não possuir uma definição preestabelecida de como essas acontecem. Dessa forma, essas fases dependem da visão do projetista do sistema. Considera-se que algumas propriedades globais do sistema, que eram predefinidas pelo projetista do sistema na fase de projeto de um SMA, agora são representadas de forma explícita e efetivamente utilizadas pelos agentes no sistema.

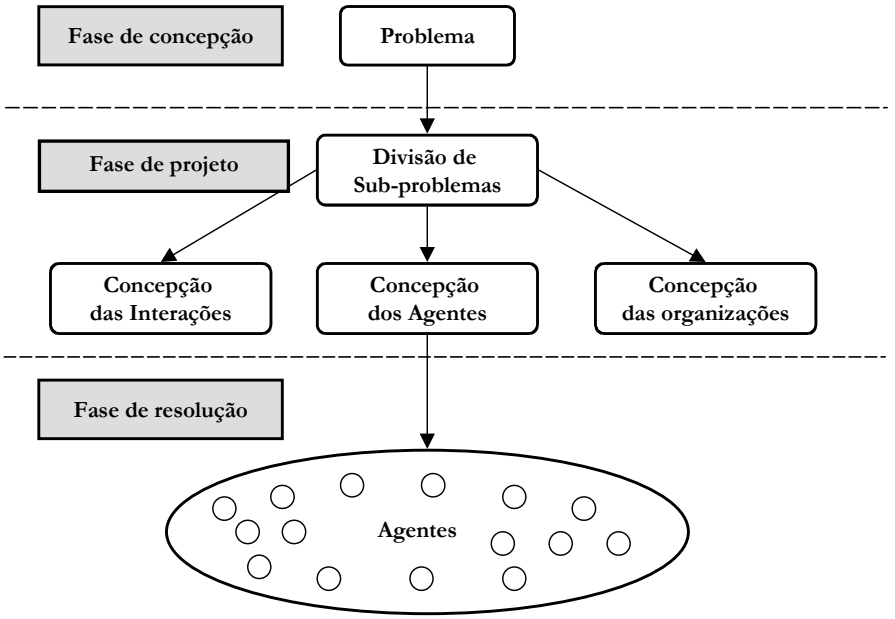


Figura 5 - Processo de projeto de um SMA [SIC95]

Para o SMA resolver problemas coerentemente, os agentes devem:

- Comunicar-se entre si;
- Coordenar suas atividades;
- Negociar;
- Detectar e resolver conflitos.

Os conflitos entre os agentes podem variar de simples contenção de recursos limitados a computações complexas onde os agentes discordam por causa de discrepâncias entre seus domínios de especialidade (*expertise*).

A seguir serão apresentados os principais aspectos para que um projeto de Sistemas Multiagentes seja elaborado.

2.3.1.1 Aspectos fundamentais para o projeto de SMA

Organização

Organização é quando existe algum tipo de conexão entre os agentes e se a execução combinada de atividades entre esses possibilitar o alcance de melhores resultados, se esses resultados forem comparados a atuações não conjugadas.

Conforme [BAS98], uma organização precisa ser gerenciada para tornar-se operacional. Para tanto, são necessárias as seguintes ações: planejamento, eventuais reestruturações físicas e conceituais, tomada de decisão, ações de coordenação. Além disto, é fundamental que uma organização possua conhecimento sobre o mundo externo, no que tange aos seus fatos e modelos conceituais. Uma organização garante a sua sobrevivência a partir de sua capacidade de aprendizado e de ajuste às características do mundo externo.

Moulin [MOU96] diz que uma organização é entendida como um conjunto de agentes que possuem compromissos mútuos e globais, compartilham crenças comuns e passam a ter intenções conjuntas quando atuam juntos no sentido de atingir um mesmo objetivo. Ressaltam os autores que um agente possui suas próprias crenças, intenções e compromissos, e que estas individualidades devem caminhar no sentido de permitir cooperação entre estes agentes, de modo a possibilitar a existência de uma organização de fato.

Cada agente de um grupo deve executar um papel; a partir de uma definição clara deste papel e de sua atribuição a um agente habilitado para sua execução, cria-se a possibilidade de o grupo desenvolver suas atividades em conjunto, formando um time coerente, capaz de atingir os resultados desejados em acordo com seus objetivos.

Coordenação

Jennings [JEN98a] coloca que, o processo de coordenação dos agentes precisa garantir que: todas as partes componentes de um problema estejam incluídas nas atividades de pelo menos um agente; os agentes interajam de forma que suas atividades sejam desenvolvidas e integradas no sentido de uma solução global; os membros do grupo de trabalho atuem de forma determinada e consistente; todos estes objetivos sejam atingidos de acordo com as disponibilidades computacionais e as limitações de recursos.

O autor enfoca que existem três razões principais pelas quais as ações de vários agentes necessitam ser coordenadas:

- Existência de interdependência entre as ações dos agentes;
- Necessidade de que o grupo de agentes respeite restrições globais do problema;
- Necessidades de procedimentos que garantam a harmonia na execução de uma única ação de forma conjunta por mais de um agente.

Jennings e Wooldridge [JEN98a] propõem que o processo de coordenação seja baseado nos conceitos de compromisso e convenções. O desenvolvimento de modelos integrados de

coordenação envolve a compreensão dos conceitos fundamentais que sustentam as interações entre os agentes. Primeiramente, nesta análise determina-se a perspectiva a partir de onde a coordenação pode ser descrita, sendo que o processo de coordenação pode ser melhor estudado a partir da análise da estrutura interna de um agente, como suas crenças, desejos, intenções, preferências, entre outros. Em segundo lugar, é decidido o modelo a ser utilizado para descrever os agentes, onde o modelo de coordenação é fundamentado sobre a Hipótese da Centralização de Compromissos e Convenções, que afirma que todos os mecanismos de coordenação podem ser reduzidos a compromissos e suas convenções associadas.

Os compromissos correspondem a promessas assumidas com um determinado curso de ação e convenções, as quais fornecem recursos para o monitoramento dos compromissos em circunstâncias de mudança.

Segundo Bastos [BAS98], quando os compromissos são assumidos em conjunto pelos agentes, o problema está no fato de que o estado geral do compromisso fica distribuído entre os agentes participantes do mesmo. Neste caso, deve-se assegurar que todos os agentes envolvidos comuniquem-se entre si para a manutenção do conhecimento por todos sobre o andamento de seu compromisso conjunto.

Ainda de acordo com [BAS98], as convenções referem-se às políticas adotadas pelos agentes para governarem a reconsideração dos seus compromissos. Descrevem as circunstâncias sobre as quais um agente pode reconsiderar seus compromissos. Tais convenções, ao serem elaboradas, devem levar em conta a necessidade de um balanceamento entre a constante reconsideração dos compromissos e a inflexibilidade nesta reconsideração. Este equilíbrio é fundamental, tendo em vista que implica a performance do agente e a capacidade de resposta às mudanças ambientais.

Assim como as convenções individuais de um agente descrevem as circunstâncias sobre as quais o mesmo deve reconsiderar seus compromissos, faz-se necessário o estabelecimento de convenções sociais a serem compartilhadas por todos os agentes, especificando como um determinado agente deve comportar-se com relação aos membros da comunidade quando os seus compromissos são alterados.

O autor salienta que a definição das convenções para as situações que envolvem ações conjuntas, deve considerar que os agentes necessitam compartilhar tanto informações globais sobre o *status* do compromisso conjunto, como do compromisso individual de cada agente com o grupo em realizar sua parte.

Negociação

Conforme Müller [MUL97], a negociação é um mecanismo que permite aos agentes autônomos chegarem a um consenso em relação a um determinado problema. Dessa forma, durante o processo de negociação pode ser necessária a modificação dos planos locais dos agentes envolvidos, objetivando chegar a uma decisão em comum. Mecanismos de negociação entre agentes são utilizados para reconhecimento e resolução de conflitos e para alocação de tarefas e recursos.

Bastos [BAS98] salienta que para existir a negociação é necessária a utilização de uma linguagem de comunicação apropriada e, que os agentes tenham a capacidade para tomar decisões.

Na negociação o modelo de Redes de Contrato, proposto por Smith [SMI80] é bastante utilizado. Nela a distribuição de atividades do sistema é um processo interativo que ocorre através da negociação entre agentes autônomos. As redes consistem de um conjunto de agentes (ou nodos) que negociam entre si, através da troca de mensagens. Pode-se identificar três tipos de agentes:

- **Gerente**: é o agente que identifica a atividade a ser satisfeita e designa um agente para executá-la;
- **Proponente**: são os agentes que apresentam uma proposta para executar uma atividade;
- **Contratado**: refere-se ao agente-proponente, cuja proposta foi aceita pelo gerente e que, conseqüentemente, passa a ser o responsável pela execução da atividade.

Neste modelo um agente-gerente solicita algum tipo de serviço e difunde uma requisição de tarefa, para a qual os demais agentes enviam propostas com ofertas. O gerente usa, então, uma função de avaliação para escolher a melhor entre as ofertas recebidas.

Cooperação

Uma política de cooperação se faz necessária num sistema multiagente já que é ela quem dita a maneira com que estes demonstram suas necessidades a outros agentes a fim de realizar uma determinada tarefa.

Conforme [PAR97], o processo de cooperação pode acontecer de duas maneiras: *Task Sharing* e *Result Sharing*. O primeiro tipo se inicia quando um agente tem uma tarefa para realizar e necessita da ajuda dos outros agentes na sociedade. Já no último, os agentes disponibilizam

informações para a sociedade, acreditando que algum agente possa necessitar dela em algum momento.

Um agente ao realizar uma determinada tarefa geralmente necessita da ajuda dos outros agentes que compõem o sistema, ou atua no auxílio de outros para satisfazer o objetivo geral. Dessa forma, os agentes precisam saber como encontrar alguém com quem cooperar visando a busca desse objetivo.

Planejamento

Segundo Bastos [BAS98], os pesquisadores, na visão clássica, tinham como premissa que as mudanças no ambiente da aplicação resultavam tão somente das ações dos agentes. Hoje, considera-se a possibilidade da existência de eventos que fogem ao controle dos agentes envolvidos na aplicação, existindo então a necessidade de que os planos sejam monitorados e revisados dinamicamente, de modo a responder a estes eventos e suas conseqüências.

A pesquisa em planejamento para múltiplos agentes divide-se em duas abordagens básicas: planejamento multiagente centralizado e planejamento multiagente distribuído [MOU96].

No planejamento multiagente centralizado o planejamento é realizado antes da execução das ações. Um plano coletivo deve ser elaborado centralizadamente por um agente e posteriormente alocado aos agentes executores. O agente elaborador do plano centralizado possui a função de receber os planos individuais de cada agente, reconhecer a interação entre os objetivos destes agentes através de seus planos e modificar estes planos de modo a remover os possíveis conflitos.

No planejamento multiagente distribuído, cada agente elabora os seus planos individuais e troca informações com outros agentes visando a identificação e resolução de conflitos eventuais, neste planejamento não existe um agente com uma visão global das atividades do grupo de agentes. Eventos inesperados podem surgir a qualquer momento, estabelecendo as necessidades de intercalar planejamento e execução para cada agente, em resposta à dinâmica destas demandas.

O planejamento distribuído pode apresentar problemas como:

- Um agente poderia formular um curso de ação que levasse em conta as ações a serem executadas ao mesmo tempo por outros agentes;
- Um agente poderia descobrir o curso de ação a ser realizado por outros agentes;
- Um agente poderia identificar de que forma os outros agentes poderiam antecipar suas próprias ações;

- Um agente poderia influenciar outros agentes a mudarem os modelos de suas ações;
- Um agente poderia comprometer-se com seus próprios modelos.

Conforme [DUR96], o fato de que os planos de um agente podem mudar a qualquer momento é ressaltado, e essa mudança de planos de um agente pode ser justificada pelos resultados de suas próprias ações, pelos resultados das ações de outros agentes, por mudanças no ambiente, por mudanças no objetivo do agente ou por mudanças na percepção do agente quanto ao contexto multiagente em que está inserido.

Dessa forma, como consta no trabalho de Bastos [BAS98] ele propõe uma abordagem denominada PGP (Planejamento Global Parcial), o qual fornece aos agentes a habilidade de construir planos locais, trocar informações sobre seus planos, construir planos globais parciais com base nas informações recebidas, revisar seus planos locais a partir dos planos globais parciais e escolher ações para a realização de seus planos locais. Destaca-se a importância de que os planos locais devem envolver caminhos alternativos, tendo em vista, as incertezas originadas pela dependência com o resultado de ações prévias e com as mudanças ambientais.

O plano global parcial captura as atividades concorrentes planejadas por cada um dos agentes para a realização do processo de coordenação. Em PGP, a ênfase da coordenação é a de permitir o cumprimento das atividades de cada agente através da garantia da execução das atividades antecedentes relacionadas e de evitar atividades redundantes [BAS98].

Comunicação e interação

Conforme [BAS98] a interação entre agentes pode ocorrer através de comunicação ou em função da modificação do mundo no qual eles estejam atuando.

De acordo com Bond [BON88] e Gasser [GAS01] o processo de interação propicia a um conjunto de agentes combinar seus esforços na busca de solução de problemas distribuídos. Interações são inerentemente dependentes das ações de coordenação de pelo menos dois agentes. Dessa forma, salienta que é importante perceber que o processo de coordenação entre agentes não depende necessariamente de comunicação, uma vez que eles podem coordenar suas ações baseadas no conhecimento que possuem dos modelos dos outros.

Os aspectos fundamentais a serem considerados para a interação entre agentes são:

- Entre quem a interação deve ocorrer;
- Quando a interação deve ocorrer;

- Qual deve ser o conteúdo da interação ou comunicação;
- Como a interação dever ser realizada, definindo os processos e recursos a serem utilizados;
- Por que a interação é necessária;
- Com base em que mecanismo será estabelecida a compreensão mútua.

As interações através de procedimentos ocorrem quando uma mudança ambiental tem significado para o domínio de funções de um determinado agente. Essa alteração de ambiente é percebida pelo agente como uma mensagem, levando-o a executar uma ação no sentido de modificar o ambiente em resposta à mesma. Interações através de convenções ocorrem em função de mudanças ambientais, que são independentes do domínio do problema e cujo significado é interpretá-lo através de convenções. Normalmente o termo comunicação é associado a este segundo caso, tendo em vista que, no primeiro não existe a troca direta de mensagens entre agentes [BAS98].

Existem duas principais estratégias de suporte à comunicação em um SMA, a Troca de Mensagens (*Message Passing*)[2] [HEW77] diretamente entre os agentes, e a utilização de Quadro-Negro (*Blackboard*)[3] [HAY85]. Essas estratégias são conhecidas como arquiteturas da sociedade de agentes.

Nos SMAs um grupo de agentes se reúne para resolver um problema específico. Dessa forma, como salienta Hübner [HUB00], os agentes não precisam necessariamente ter sido desenvolvidos utilizando as mesmas ferramentas e modelos. Entretanto, esse grupo de agentes deve concordar no uso de uma linguagem comum. Além disso, como nas sociedades humanas, a comunicação é mais eficiente quando seus usuários seguem certas regras, chamadas de protocolos de comunicação. Então, com o objetivo de viabilizar a comunicação entre os agentes e possibilitar a descrição das regras que gerenciam o seu fluxo de informação foram propostas algumas linguagens de comunicação de agentes chamadas de KQML (*Knowledge Query Manipulation Language*) e ACL (*Agent Comunnication Language)*, onde, essas linguagem possuem especificações como a FIPA-ACL (*Foundation Intelligent Physical Agents – Agent Comunnication Language).*

---

[2] *Message Passing* (Troca de Mensagens): Na arquitetura de troca de mensagens, os agentes comunicam-se diretamente uns com os outros enviando mensagens assíncronas. Não existe o papel do quadro-negro como intermediário do processo de interação, mas pode haver a presença de um ou mais agentes facilitadores de comunicação.

[3] *Blackboard* (Quadro-Negro): A sociedade baseada na arquitetura quadro-negro (*blackboard*) dita que os agentes não se comunicam diretamente mas sempre através do quadro-negro. Essa arquitetura surgiu antes do aparecimento dos sistemas multiagentes.

KQML (Knowledge Query Manipulation Language)

A KQML é uma especificação de linguagem e protocolo de comunicação entre agentes, onde seu objetivo é ser um meio comum de troca de informação e conhecimento entre agentes.

Hübner e Sichman [HUB00] apresentam que esta especificação (KQML) foi amplamente adotada pelo fato de possuir algumas características importantes numa ACL:

- Qualquer linguagem (como por exemplo, LISP, Prolog, português), pode ser usada para escrever o conteúdo das mensagens;
- As informações que são necessárias para escrever o conteúdo das mensagens estão incluídas na própria comunicação;
- Quando os agentes trocam mensagens KQML, o mecanismo de transporte é transparente, ou seja, como a mensagem sai do agente emissor e chega no receptor;
- Formato das mensagens é simples, fácil de ler por pessoas e de ser analisada por um *parser*[4].

A especificação KQML enfoca um grande conjunto de mensagens pré-definidas (performativas), as quais definem as operações possíveis de serem executadas pelos agentes. Assim, aqueles que estão em conformidade com a KQML podem responder a estas mensagens de maneira adequada independentemente da estrutura do seu emissor. [FIN94]

As arquiteturas que utilizam KQML adicionam um agente facilitador à sociedade. Este agente sabe quem são os agentes da sociedade e que requisições cada um é capaz de responder, dessa forma, o procedimento de apresentação entre os agentes é simplificado [HUB00].

Conforme [FIN94] a linguagem KQML não soluciona ou sugere como deve estar estruturada a implementação dos agentes, sendo assim, ela se identifica como uma linguagem de propósito geral, onde independentemente do ambiente ou estrutura dos agentes, é possível usá-la para prover a comunicação num sistema multiagentes.

FIPA ACL (Foundation Intelligent Physical Agents Agent Comunnication Language)

Conforme Hübner [HUB00] a especificação de ACL da FIPA está baseada nos mesmos princípios que o KQML e, tem a mesma sintaxe para as mensagens.

---

[4] *Parser*: Programa que faz análise sintática de uma entrada textual.

Esta especificação consiste de um conjunto de tipos de mensagens e a descrição formal de sua pragmática (o efeito das mensagens na mente do emissor e do receptor). Neste aspecto, a FIPA ACL difere do KQML, a semântica dos tipos de mensagens não é exatamente a mesma. Além da ACL, a FIPA também especifica uma notação para descrição de protocolos de comunicação, que é exemplificada com protocolos para requisição de serviços, rede de contratos e outros.

Salienta-se que a ACL possui a vantagem de garantir o padrão sobre o KQML, onde pessoas diferentes podem usar a mesma linguagem para implementar agentes diferentes que não conseguem se comunicar.

2.4 Metodologias de projeto

Com a evolução da tecnologia de agentes, tornou-se necessário o desenvolvimento de metodologias para o desenvolvimento de aplicações segundo o paradigma que visa a padronização das técnicas de construção de sistemas multiagentes. Algumas metodologias já foram desenvolvidas, como a GAIA, MAS-CommonKADS, metodologia orientada a agentes para modelagem operacional e MaSE, as quais apresentam-se a seguir.

2.4.1 Metodologia GAIA

A metodologia GAIA foi desenvolvida para análise e projeto de sistemas baseados em agentes, por estes serem flexíveis, terem autonomia em resolver problemas, possuírem interação e pela estrutura organizacional do sistema [WOO00b].

Wooldridge e Jennings [WOO99] afirmam que a metodologia GAIA é apropriada para o desenvolvimento de sistemas como ADEPT (*Advanced Decision Environment for Process Tasks*)e ARCHON (*ARchiteture for Cooperative Heterogeneous ON-line systems)* pois, estes possuem grande escala em aplicações no mundo real e apresentam as seguintes características:

- Os agentes fazem uso de um recurso computacional (como se cada agente possuísse os recursos de um processo UNIX);
- O objetivo é obter um sistema que maximize algumas medidas de qualidade global, mas que seja sub-ótimo pelo ponto de vista dos componentes do sistema. Gaia não possui a finalidade de modelar sistemas que possuem conflitos;

- Nos agentes heterogêneos, diferentes agentes podem ser implementados usando diferentes linguagens de programação, arquiteturas e técnicas;
- A estrutura de organização do sistema é estática, com isso, o relacionamento entre agentes não muda em tempo de execução;
- O sistema completo contém um pequeno número de diferentes tipos de agentes (menos que 100).

A metodologia GAIA permite que o analista passe da especificação de requisitos para o projeto, que é suficientemente detalhado, permitindo que seja implementado diretamente. Na metodologia GAIA, conforme Wooldridge [WOO00b], o analista migra de um conceito abstrato para um outro incrementalmente mais concreto. Cada migração sucessiva inclui aspectos na implementação, e restringe o espaço de sistemas que podem ser implementados para satisfazer a especificação de requisitos original.

Os principais conceitos dessa metodologia podem ser divididos em duas categorias: abstratos e concretos. *Entidades abstratas* são aquelas usadas durante a análise representando os conceitos do sistema, mas que não necessariamente tem alguma ligação com o sistema. *Entidades concretas*, em contraste, são usadas dentro do processo de projeto, e serão direcionadas, em contrapartida, em tempo de execução do sistema.

O objetivo do estágio de análise é desenvolver um entendimento do sistema e sua estrutura, chamada de organização de sistema, que é tida como uma coleção de papéis, que se relacionam com outros e que fazem parte de padrões sistemáticos e institucionalizados de interação com outros papéis [WOO00b].

Na fase de projeto, o objetivo é transformar o modelo de análise em modelo de baixo nível de abstração com técnicas de projetos tradicionais (incluindo técnicas de OO - Orientação a Objetos) que possam ser aplicadas de maneira a implementar o agente. Em outras palavras, GAIA se preocupa em como a sociedade de agentes, coopera para atingir objetivos em nível de sistema e o que é solicitado para cada agente.

2.4.2 Metodologia MAS-CommonKADS

Conforme Iglesias [IGL98], MAS-CommonKADS é uma extensão da metodologia CommonKADS[5] para a modelagem de sistemas multiagentes.

[5] CommonKADS: principal metodologia estruturada de suporte a engenharia de conhecimento

A aplicação dessa metodologia consiste do desenvolvimento de diferentes modelos, onde cada modelo consiste de entidades a serem modeladas e suas relações. Essa metodologia define os seguintes modelos [IGL98]:

- *Modelo de agente*: especifica as características dos agentes, como capacidade de raciocínio, habilidades, serviços, grupos e hierarquias de agentes em modelo de organização;
- *Modelo de atividade*: descreve a tarefa que o agente pode executar, como objetivos, decomposições, ingredientes e métodos de resolução de problemas, entre outros;
- *Modelo de especialidade*: descreve o conhecimento necessário para o agente alcançar seus objetivos, assim como a estrutura do relacionamento entre agentes;
- *Modelo de organização*: descreve a organização na qual o SMA irá ser introduzido e a organização social da sociedade de agente;
- *Modelo de coordenação*: descreve a conversação entre agentes, como interações, protocolos e capacidades requeridas;
- *Modelo de comunicação*: detalha a interação do agente de *software* e humano e os fatores humanos para o desenvolvimento da interface dos usuários;
- *Modelo de projeto*: coleta os modelos previstos e consiste de três sub-modelos, como projeto de rede para esboço de aspectos relevantes da infra-estrutura da rede do agente; projeto de agente para divisão ou composição dos agentes na análise, conforme o critério prático e seleção da arquitetura de agente mais adequada para cada agente; e projeto de plataforma para seleção da plataforma de desenvolvimento do agente para cada arquitetura de agente.

2.4.3 Metodologia orientada a agentes para modelagem operacional

Conforme Kendall [KEN96] esta metodologia propõe a combinação de metodologias orientadas a objeto OOSE (*Object Oriented Software Egeneering)* e metodologias de modelagem operacional IDEF (*Integration DEfinition for Function modeling)* e CIMOSA (*Computer Integrated Manufacturing Open System Architeture).*

O método de IDEF é um padrão para a definição de requisitos em empresas de manufatura e outras organizações. O IDEF compreende três modelagens de metodologias: o modelo da função ($IDEF_0$), modelo de informação ($IDEF_1$), e o modelo dinâmico ($IDEF_2$).

Na metodologia orientada a agentes os modelos identificados são:

- *Modelo funcional:* descreve as funções (saídas, entradas, mecanismos e controle) usando diagramas $IDEF_0$ que incluem a seleção de possíveis métodos que dependem da entrada e do controle.

- *Modelo caso de uso:* descreve os atores envolvidos em cada função, usando a notação de caso de uso OOSE.

- *Modelo dinâmico:* este modelo é entendido pela análise da interação dos objetos. Os casos de uso são representados em diagramas de evento passo-a-passo.

- *Sistema orientado a agente*: é composto por identificação de agente (os atores dos casos de uso são identificados como agentes); protocolos de coordenação ou *scripts* (descritos no diagrama de estados); invocação de plano (diagramas de eventos passo-a-passo são ampliados para incluir condições para indicação quando um plano deve ser invocado); crenças, sensores e executores (entradas das funções deveriam ser modeladas como crenças ou obtidas de objetos via sensores, e objetivos executados deveriam ser modelados como mudanças para crenças ou modificações por executores).

2.4.4 Metodologia MaSE

A MaSE (*Multiagent Systems Engineering*) é uma metodologia para Engenharia de Sistemas Multiagentes proposta por Wood [WOO00a], a qual é independente de qualquer arquitetura multiagente, arquitetura de agente, linguagem de programação ou sistema de troca de mensagem.

A metodologia MaSE, conforme [WOO00a] é usada na modelagem de sistemas estáticos. Ou seja, aqueles onde os agentes não podem ser criados, destruídos ou movidos durante a execução. As principais atividades da fase de análise são: captura de objetos do sistema, refinamento dos papéis baseado-se nos objetos e desenvolvimento das interações entre os papéis. Na fase de projeto, em seguida, são definidas as classes de agentes e a arquitetura do sistema.

Pode-se ter uma visão geral da metodologia MaSE e seus modelos na Figura 6.

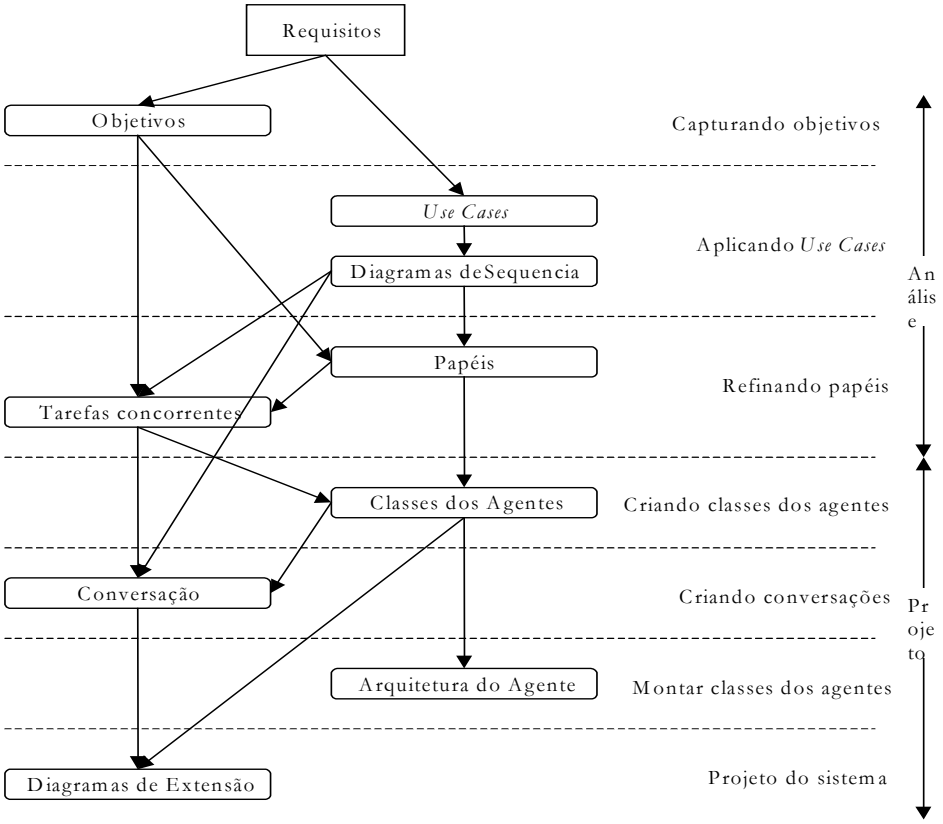


Figura 6 - Visão geral da metodologia MaSE

A seguir apresenta-se o detalhamento de cada fase da metodologia MaSE:

A primeira fase de MaSE é **capturando os objetivos**, na qual relaciona-se as especificações iniciais do sistema e as transformando em um conjunto estruturado de objetivos através do diagrama de hierarquia de objetivos. Esta fase possui duas partes: identificar e estruturar os objetivos.

Conforme Wood [WOO00a], a segunda fase da metodologia é **aplicando Use Cases**, onde a conversação entre os agentes é a verdadeira espinha dorsal de um SMA, sendo que, eles capacitam a distribuição das operações que é a força da tecnologia de agentes. Então, essa fase da metodologia MaSE trata da construção dessas conversações e cria Use Cases para facilitar o processo.

O terceiro passo da metodologia MaSE é **refinando papéis**, na qual os objetivos estruturados do Diagrama de Hierarquia de Objetivos são transformados em uma forma mais útil para a especificação de papéis em SMA. Papéis são usados para definir classes dos agentes e capturar os objetivos do sistema durante a fase de projeto.

Em seguida, apresenta-se a fase, **criando classes de agentes**. Nesta fase na qual as classes dos agentes são identificadas pelos papéis, onde o resultado é um diagrama das Classes dos Agentes, no qual são descritas as classes dos agentes e as conversações entre elas. Durante esta fase, as classes de agentes consistem em dois componentes: papéis e conversações. Em uma fase posterior, detalhes internos são adicionados.

Dando continuidade, a próxima fase é **construindo conversações**, onde uma conversação na metodologia MaSE define um protocolo de coordenação entre dois agentes. Uma conversação consiste de dois Diagramas de Comunicação, uma para o iniciador (*initiator*) da mensagem e um para aquele que responde (*responder*).

Mostrando classes de agentes, é a fase em que as classes internas dos agentes são criadas. Conforme Wood [WOO00a], existem três métodos para montar os componentes para definir uma classe de agente. O primeiro é resgatar uma das quatro arquiteturas pré-definidas (BDI, reativo, *planning* (planejamento) e baseado em conhecimento). O segundo método consiste em resgatar os componentes pré-definidos montando-os em uma arquitetura definida pelo usuário. O terceiro método consiste em definir os componentes e arquitetura sem nenhum componente preestabelecido.

A fase final da metodologia MaSE é o **projeto do sistema**, o qual recebe as classes dos agentes e as instâncias como agentes reais, usando o Diagrama de Disposição para mostrar os números, tipos e locações de agentes dentro do sistema. Esta fase é a mais simples da metodologia MaSE pois, grande parte do trabalho foi concluída nas etapas anteriores.

2.5 Formalismos para SMA

A maioria dos modelos e linguagens de agentes são entendidas como abstrações de comportamento computacional, eventualmente para serem transformadas em *softwares* (programas). Tais modelos são necessários para especificação e construção de *software*. Diferentes gêneros de modelos são necessários em diferentes estágios e aspectos na Engenharia de Software [YU01]. Como a tecnologia de agentes está em crescente desenvolvimento, a atenção está voltada para um conjunto de modelos complementares, notações e métodos para cobrir o ciclo de vida do *software*.

Nesse contexto foram estudados alguns formalismos que podem ser utilizados no processo de especificação de SMAs. Também, a fim de definir um formalismo que melhor se adapte e possua mais vantagens para que seja aplicado no enfoque desse trabalho.

Estes formalismos são: Proposta de Souza, Lógica Temporal de Ações (TLA), Modelos de Dependência de Ator (DA), Estados Mentais – BDI e, o Formalismo Z.

2.5.1 Proposta de Souza

No trabalho desenvolvido por Souza [SOU01] é proposto um conjunto de diagramas para o processo de elaboração das partes integrantes do modelo de análise para um SMA.

Conforme Souza [SOU01] para representar SMAs, utiliza-se uma linguagem extensão de UML (*Unified Modeling Language*) propondo extensões de AUML. Para permitir a aceitação da tecnologia AUML, os agentes são apresentados como uma extensão de objetos ativos, exibindo autonomia dinâmica[6] e autonomia determinística[7].

Souza [SOU01] justificou sua proposta de trabalho baseando-se no fato de que a UML, sendo uma linguagem desenvolvida para o paradigma OO, não atende a representação de sistemas OA (Orientado a Agentes), porque possuem características e propriedades diferenciadas.

A Tabela 1 apresenta a proposta de Souza [SOU01], relacionando os Diagramas que compõe a modelagem (com os aspectos fundamentais) para projetar Sistemas Multiagentes. Uma vez que, segundo Souza, a Orientação a Objetos não atende a representação de sistemas Orientados a Agentes.

Tabela 1 - Diagramas x Aspectos fundamentais de SMA

Diagramas	Propriedades
Diagrama de Casos de Uso	cooperação, conflito e negociação
Diagrama da Sociedade	interação e comunicação, estrutura e organização
Diagrama de Interação Seqüencial	cooperação, conflito, negociação, interação e comunicação, coordenação, planejamento

A seguir são apresentados os Diagramas de Casos de Uso, da Sociedade e de Interação Seqüencial desenvolvidos por Souza [SOU01].

[6] Autonomia dinâmica: é a habilidade de iniciar ação sem invocação externa.
[7] Autonomia determinística: é a habilidade de recusar ou modificar um pedido externo.

2.5.1.1 Diagrama de Casos de Uso em SMA

Conforme Souza [SOU01], o conceito de casos de uso pode ser estendido para representar os cenários de um SMA de forma a ilustrar e facilitar a compreensão e composição do modelo de análise do sistema. A forma como é representado o Diagrama de Casos de Uso para SMAs é a mesma usada pela UML, sendo que em SMAs são representados os fluxos presentes na sociedade e as interações entre os agentes, as quais caracterizam suas funcionalidades.

A diferença existente nos casos de uso em SMA, é a identificação de agentes (**quando** e **quem** interagem com os atores) ao invés de identificação de objetos em UML. Nos casos de uso são representadas as propriedades de cooperação, conflito e negociação em um SMA.

Os Diagramas de Casos de Uso possuem grande importância para visualizar, especificar, documentar e modelar o comportamento do sistema. Como exemplo de Diagrama de Casos de Uso pode-se apresentar a Figura 7, onde é representado o cenário *Criar agentes*.

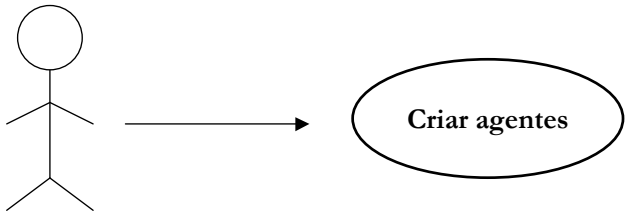


Figura 7 - Exemplo de Diagrama de Casos de Uso

2.5.1.2 Diagrama da Sociedade

Souza [SOU01] salienta que o conceito de classe de objeto pode ser estendido para classe de agente envolvendo a descrição de um conjunto de agentes que compartilham atributos, serviços, papéis e responsabilidades. Uma classe de agente é uma entidade que representa o comportamento do agente, as suas responsabilidades na sociedade e define sua arquitetura para criação de novas instâncias de agentes na sociedade, conforme Figura 8.

O Diagrama da Sociedade representa as propriedades de interação e comunicação, estrutura e organização de um SMA, porque descreve hierarquias de associação durante sua interação.

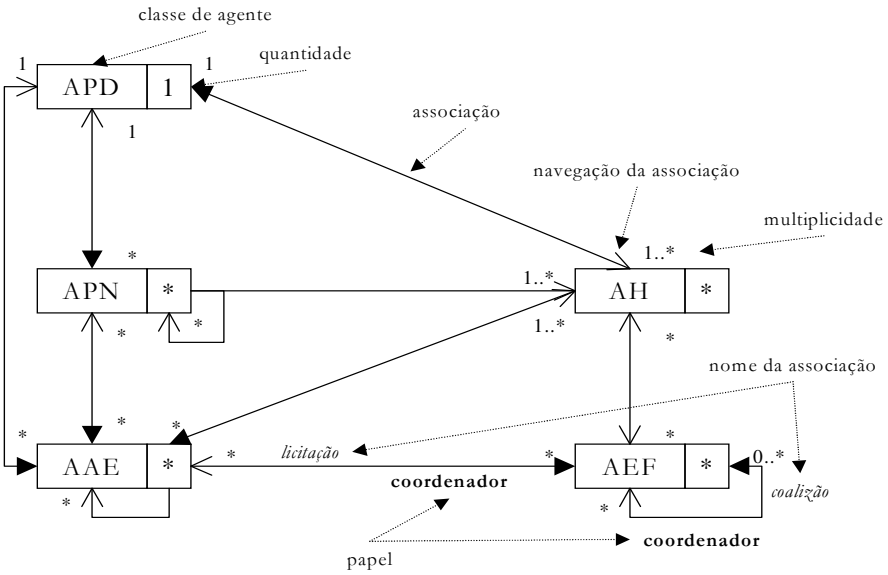


Figura 8 - Exemplo de Diagrama da Sociedade [SOU01]

Classe de Agente

Uma classe deve representar os atributos, serviços e papéis que serão desempenhados pelos agentes, as responsabilidades e a quantidade de instâncias de agentes para a classe.

Associações

Na representação do Diagrama da Sociedade, é necessário apresentar associações entre as classes de agentes. Uma associação pode ter um nome, utilizado para descrever a natureza do relacionamento, onde este nome é usado quando existe mais de uma associação conectando o mesmo par de classes. Na associação encontra-se também a multiplicidade que a cardinalidade que um conjunto pode assumir. Para os agentes é necessária a interação a fim de permitir que suas atividades sejam desenvolvidas e integradas no sentido de uma solução ser obtida com colaboração do grupo. Ainda é necessário definir aspectos hierárquicos de autoridade, a fim de representar relações envolvendo comando entre os mesmos.

Interface

A *interface* na modelagem de um SMA é uma reunião de assinaturas definidas para especificar um serviço de uma classe, onde se preocupa com *o que* o agente vai fazer. Para cada papel, tem-se uma *interface* que especifica esses serviços que podem ser desempenhados por um agente através de assinaturas simplificadas para facilitar a representação. A *Interface* preocupa-se

com **o que** o agente vai fazer, pois **como fazer** deve ser representado na arquitetura interna do agente (etapa do projeto de agente).

Modelo de Funções

O Modelo de Funções relaciona as responsabilidades inerentes a cada papel que o agente pode assumir, associando a seus respectivos serviços complementando as informações contidas no Diagrama da Sociedade.

2.5.1.3 Diagrama de Interação Seqüencial

O conceito de Diagrama de Interação pode ser estendido para SMA, com o objetivo de representar os cenários envolvidos nos casos de uso identificados para o sistema. Souza [SOU01] propõe o Diagrama de Interação Seqüencial, que é utilizado para fazer a modelagem dos aspectos dinâmicos do sistema.

Os Diagramas de Interação Seqüencial representam a troca de mensagens entre os agentes, e justificam-se por permitir visualizar, especificar, construir e documentar a dinâmica da sociedade de agentes em seus vários casos de uso.

Para que os agentes nos SMA sejam representados, é preciso definir os aspectos hierárquicos de autoridade (tipo associação de comandos ou associação de comunicação) e é preciso estabelecer um protocolo de comunicação entre os agentes.

Na Figura 9, é apresentado um exemplo de Diagrama de Interação Seqüencial onde: o agente AEF (Agente Entidade Funcional) localizado a esquerda, indica o cliente da mensagem representado pela letra c. O agente AEF localizado a direita, representa o agente fornecedor da mensagem, mas sem possuir um nome de instância, pois neste caso não é uma instância e sim um conjunto com vários membros. O agente AEF com o papel de coordenador inicia o processo fazendo uma solicitação a todos os membros da coalizão. Logo após, recebe uma proposta do agente fornecedor, enviada da mesma barra de ativação, caracterizando seqüência no processo. A resposta pode ser confirmando a solicitação ou não confirmando. Apenas um AC (ato de comunicação) será enviado, devido a presença da caixa de decisão com um **x**. Em resposta a solicitação confirmada, um AC é enviado para uma nova barra de ativação informando contrato fechado, encerrando o processo. E em resposta a solicitação não confirmada, um novo AC é enviado solicitando redefinição. Finalizando envia-se a proposta, encerrando assim o processo.

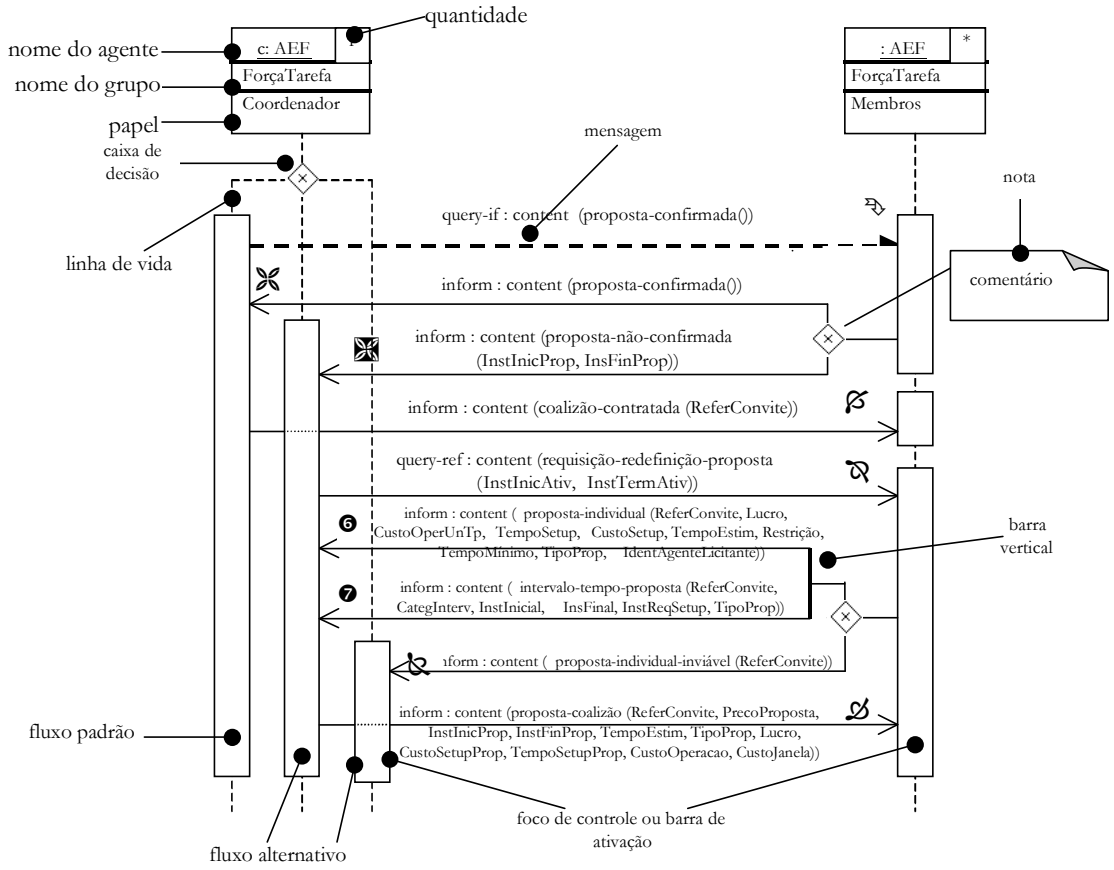


Figura 9 - Componentes do Diagrama de Interação Seqüencial [SOU01]

2.5.1.4 Interações entre os diagramas

Conforme [SOU01], no processo de modelagem e refinamento dos diagramas é fundamental que haja consistência entre os mesmos, de forma a compor de maneira integrada as três visões requeridas para criação de um sistema.

Na proposta de Souza [SOU01] o caminho natural do processo de modelagem de um SMA, inicia-se a partir dos atores que disparam os processos que geram os Diagramas de Casos de Uso.

Os atores podem ser humanos, outros agentes, ou um sistema externo que inicializa o processo. Os agentes envolvidos nos processos (casos de uso) pertencem a sociedade de agentes, compondo o Diagrama da Sociedade.

Para complementar esse diagrama, foi criado o Modelo de Funções que estabelece para cada um dos papéis de cada agente suas responsabilidades, restrições e serviços. E finalmente

para exibir as mensagens trocadas em cada um desses relacionamentos, assim como o nível hierárquico entre os agentes e o percurso dos AC, foi utilizado o Diagrama de Interação Seqüencial.

2.5.2 Lógica Temporal de Ações (TLA)

A Lógica Temporal de Ações - TLA (*Temporal Logic of Actions*) é uma lógica linear de tempos que define um predicado sobre comportamentos, onde predicado é uma expressão booleana que contém constante e variável e, um comportamento que é definido como uma seqüência infinita de estados, onde um estado modifica valores de uma ou mais variáveis flexíveis, que são variáveis que podem assumir valores diferentes em estados diferentes.

Uma fórmula de TLA é definida como, verdadeira ou falsa em um comportamento. Esta pode ser exemplificada através da execução de um algoritmo como uma sucessão de passos, produzindo um estado novo alterando valores de uma ou mais variáveis. A lógica temporal pode ser descrita como significado semântico dessa sucessão de estados.

A semântica do sistema é o conjunto de comportamentos que este pode assumir. Um par de estados, um representando o estado antigo e outro o novo, definem o que é chamado de passo [LAM97].

A sintaxe e semântica de TLA são compostas de quatro conjuntos:

- No primeiro conjunto existem fórmulas com significado independente do estado, chamadas de constantes, são variáveis fixas cujo valor tem estado independente do que acontece nas fórmulas temporais;
- No segundo conjunto é descrita a argumentação sobre os estados, em particular, chamada de fórmula de estado que incluem funções de estado, predicados de estado (ações), variáveis fixas e flexíveis cujo valor tem estado independente do que pode acontecer nas fórmulas temporais;
- No terceiro conjunto é descrita a argumentação dos pares de estados chamados de fórmulas de transição, estas incluem funções e predicados de transição;
- No quarto conjunto é feita a argumentação sobre os comportamentos, que são sucessões infinitas de estados chamadas de fórmulas temporais, as quais são construídas com fórmulas de outros módulos que usam operadores temporais.

Lamport [LAM99] propôs uma linguagem formal para escrever especificações em TLA chamada TLA+, onde as especificações em TLA+ são divididas em módulos. Novos módulos podem ser adicionados à especificação permitindo a utilização de módulos definidos.

Para exemplificar uma especificação de sistema feita com TLA, será apresentada a especificação de uma interface para transmissão de dados entre dispositivos assíncronos, conforme [LAM99]. Neste sistema, tem-se um *Sender* e um *Receiver* conectados conforme a Figura 10.

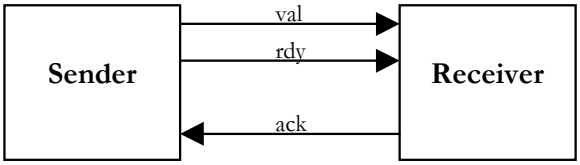


Figura 10 - Especificação do protocolo *Sender/Receiver*

Na Figura 10 dados são enviados através de *val* e as linhas *rdy* e *ack* são usadas para sincronização. *Sender* envia um dado e espera a confirmação de recebimento pelo *Receiver* para enviar o dado seguinte. Quando um dado é enviado, ele é transmitido através de *val* e o sinal em *rdy* é trocado, informando ao *receiver* que o *Sender* enviou um dado e espera confirmação de recebimento. O *Receiver* então confirma o recebimento trocando o sinal em *ack*.

Na Figura 11 é apresentado um exemplo de como escrever uma especificação em TLA+ [LAM00], o qual representa uma estrutura genérica em TLA+. Conforme [LAM00], a parte de *EXTENDS* apresenta a declaração do módulo predefinido a ser incorporado à especificação. Todos os módulos predefinidos a serem utilizados por um módulo novo são declarados após a palavra reservada EXTENDS. Como exemplo, o módulo *Naturals* é um módulo onde estão as definições dos números naturais. Na parte CONSTANT é apresentada a lista de constantes, que são variáveis com valor constante do módulo. VARIABLE mostra a lista de variáveis do módulo. Na parte *TypeInvariant* são determinados os valores que podem ser assumidos pelas variáveis declaradas. O símbolo $\wedge$ determina a conjunção das condições estabelecidas. O mesmo ocorre para a disjunção de condições, onde cada linha é precedida do símbolo $\vee$. A parte *Init* define o predicado inicial, que são as condições iniciais do sistema que está sendo modelado. Na parte *Next* é representada a ação a ser executada no próximo estado, alterando o estado antigo. Em *Spec* é descrita que os componentes válidos deste sistema são seqüência de estados, nos quais, inicialmente, o predicado *Init* é verdadeiro e o predicado *Next* é sempre verdadeiro. A parte

THEOREM define o teorema da especificação e, na parte de *Ações* são definidas as ações que a especificação vai executar.

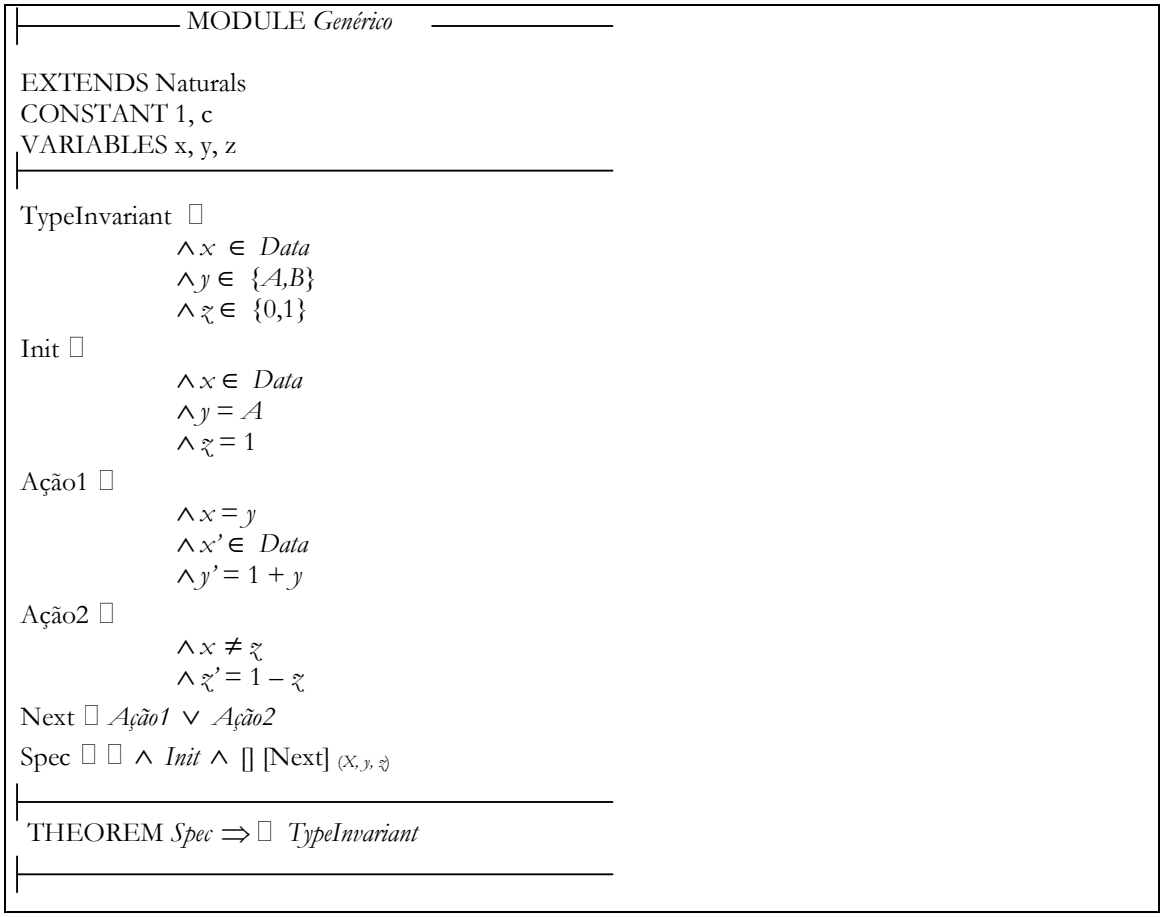


Figura 11 - Estrutura de uma especificação em TLA+

2.5.3 Modelo de Dependência de Ator (DA)

O modelo de Dependência de Ator – DA (*Dependecy Actor*) [YU01] consiste num conjunto de nodos e ligações. Cada nodo representa um ator, e cada ligação entre dois atores indica que um ator depende do outro para alguma coisa, de maneira que o primeiro possa atingir algum objetivo. Através da dependência entre atores, um ator fica apto a atingir objetivos para os quais ele não estaria apto sem a dependência ou não conseguiria atingir tão facilmente. Também se o ator, do qual o outro depende falhar, os objetivos desse outro ator também falham.

A Figura 12 mostra um modelo de dependência de ator para uma simples organização de um projeto de engenharia de *software*, onde um cliente depende de um administrador de projeto para desenvolvimento de sistemas. Este projeto depende de um projetista, um programador, e um testador para trabalhos técnicos. Os membros da equipe técnica dependem de cada um dos

outros para trabalhar em produtos intermediários, tais como projeto, código e resultados de testes. O administrador é também dependente de seu chefe para que os projetos não sejam sobrepostos, os usuários também são dependentes, para a garantia da qualidade da manutenção do sistema. O usuário depende do administrador de projeto para uma performance do sistema amigável.

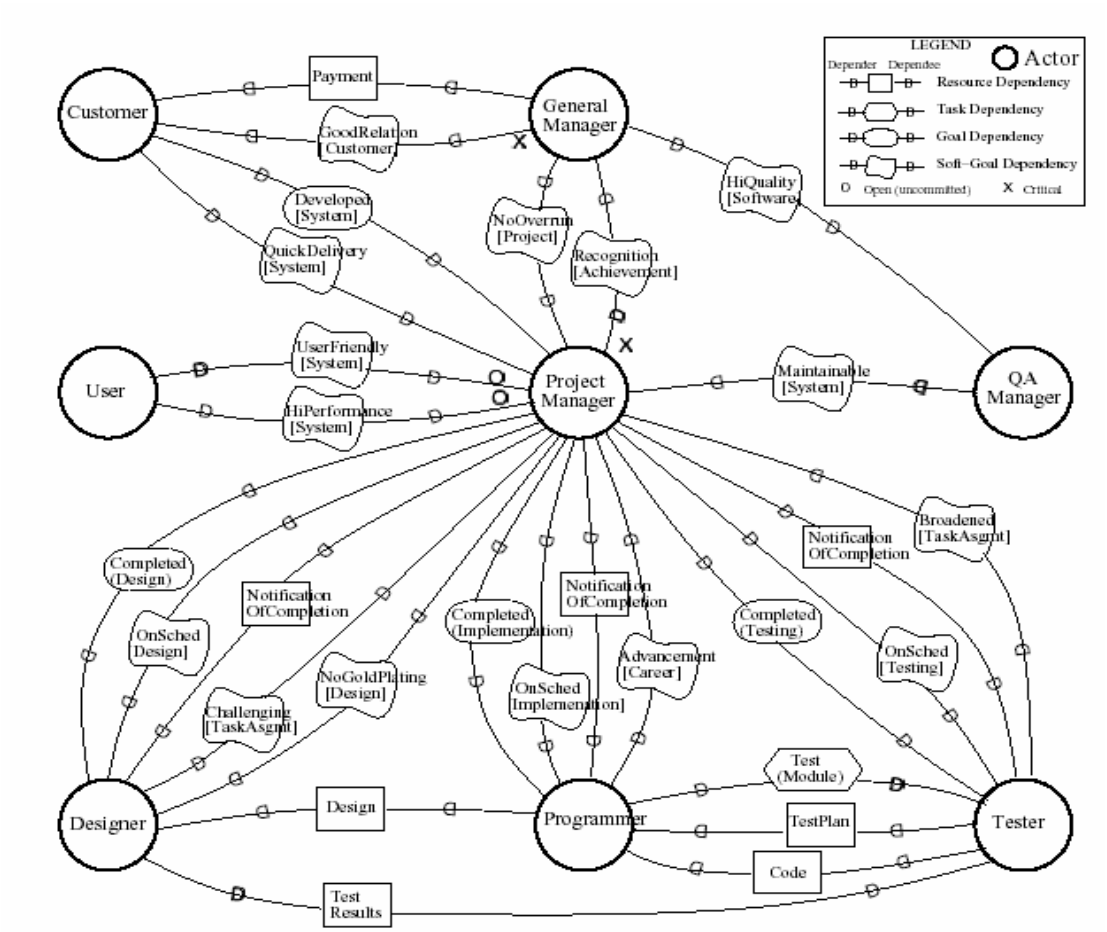


Figura 12 - Exemplo de Modelo de Dependência de Ator

O modelo de dependência de atores possui três principais tipos de dependência, baseadas em categorias ontológicas de declaração, atividade ou entidade: Dependência de Objetivo, Dependência de Tarefa e Dependência de Recurso [YU01].

Na **Dependência de Objetivo** um ator depende do outro para que uma condição se torne verdadeira. Pois, somente um estado final ou conseqüência é especificado. Na Figura 12, o

relacionamento da dependência de objetivo entre o administrador de projeto e sua equipe é para decidir como vai ser feito sua tarefa. O administrador não se preocupa em como o sistema é desenvolvido.

Já na **Dependência de Tarefa** um ator depende de outro para realizar uma atividade. O objetivo dependente para ter a atividade executada não é dado. A descrição da atividade especifica um curso particular da ação. O programador depende, no verificador, testar um módulo via uma dependência de tarefa para especificar um plano de teste. Se o administrador do projeto indicar os passos técnicos para cada membro da equipe alcançar a saída, então o administrador deverá relatar sua equipe por dependência de tarefa.

Na **Dependência de Recurso** um ator depende de outro para acessar uma entidade física ou lógica. O dependente toma a disponibilidade do recurso para ser uma solução de problema não requisitada. Na Figura 12, a dependência de administrador geral, no pagamento do cliente, a dependência dos testadores na programação do código, e o projeto de dependência de administrador em sua equipe para notificação de tarefas completas, são modeladas como dependência de recurso.

Ainda existe um quarto tipo de dependência, a **Dependência Flexível de Objetivo** que é uma variação do primeiro. É diferente da dependência de objetivo no sentido de que não existe um critério a *priori* no que se constitui atingir o objetivo. O significado de um objetivo flexível é especificado em termos de métodos que são escolhidos em curso para perseguir o objetivo. O dependedor contribui para identificar as alternativas, mas as decisões são tomadas pelo dependente.

2.5.4 Estados Mentais - BDI

Agentes dotados de estados mentais são aqueles que possuem a capacidade de selecionar seus objetivos, raciocinar sobre eles, selecionar ou criar planos, detectar conflitos entre planos, executar e revisar os planos, produzir decisões, entre outros [MOU96].

A utilização de estados mentais para modelagem de agentes cognitivos é chamada de abordagem mentalista [MOR98]. Nesta visão, o que faz qualquer componente de hardware ou *software* ser um agente é, precisamente, o fato de ele poder ser analisado e controlado em termos destes componentes mentais.

Como idéia principal para utilização de estados mentais tem-se que o agente pode selecionar racionalmente seu curso de ação baseado nas representações desses estados mentais.

Crenças

Conforme [ROA95] crenças são definidas como sendo uma visão informativa do estado do sistema, elas são necessárias para que o agente conheça o ambiente onde ele está inserido. No trabalho de [MOR98] as crenças são constantemente atualizadas através do sensoramento do ambiente, representando ainda as atitudes do agente.

D'Inverno [D'IN98] em seu trabalho diz que as crenças podem ser vistas como simples variáveis (como por exemplo na linguagem PASCAL), mas agentes modelados na arquitetura BDI representam crenças de forma simbólica (como, por exemplo, fatos em PROLOG).

Para representar as crenças de um agente é proposto um operador modal B, sendo que, este apresenta dois argumentos, o primeiro mostra de quem é a crença (agente) e o segundo representa em que o primeiro acredita (uma fórmula). Por exemplo, para dizer que João acredita que o pai de Zeus é Cronus, seria utilizado:

B(joão, pai_de(zeus, cronus))

De forma geral, tem-se:

B(A, F), onde:

A = agente; F = é uma fórmula.

Desejos

Móra [MOR98] diz que um desejo é um fato que representa um estado que é possível de ser atingido, ou então um estado a ser atingido quando certas condições venham a se estabelecer. Conforme [RAO98] um desejo representa o estado motivacional do agente, de certa forma, representa um estado que este deve atingir.

Corrêa [COR 94] em seu trabalho, descreve desejos como:

$\mathbf{D = \{\langle Des, A, P, e, l_0, t_0, v \rangle\},}$ onde:

Des = identificação do desejo; A = identificação do agente; P = uma propriedade; e = um indicador de prioridade do desejo; l_0 = local de ocorrência do desejo; t_0 = instante de ocorrência do desejo; v = registra a ocorrência do agente (0 - ocorreu, 1 - não ocorreu).

Intenções

Conforme [ROA98] quando o agente decide por uma linha de ação, entre tantas outras que ele poderia escolher, o agente passa a ter uma intenção de adotar essa linha de ação. De acordo com Móra [MOR98] uma intenção é vista como um compromisso que o agente assume com um possível futuro específico, pois ao contrário dos desejos não se pode ter intenções conflitantes devido ao fato de que elas devem encontrar suporte nas crenças do agente. Ou seja, o agente não pode intencionar aquilo que ele não acredita.

Chaib-Draa [CHA98] mostra que um agente "A" possui a intenção de atingir "P" se o agente tem um compromisso com "P" e possui planos para atendê-lo, formalizando apresenta-se:

$$\mathbf{int(A,P) = \exists\ cmt(A,P) \wedge tem.planos(A,P)}, \text{ onde:}$$

int = uma intenção; A = agente; P = uma propriedade a ser atingida; cmt = um compromisso do agente com a propriedade; tem.planos = existência de planos para atingir "P".

Uma abordagem muito utilizada dentro das arquiteturas baseadas em estados mentais (crença, desejo e intenção) é chamada de arquitetura BDI (*Belief, Desire and Intention)*. A abordagem BDI baseia-se na descrição do processamento interno de um agente utilizando um conjunto básico de estados mentais (crenças, desejos e intenções) e na definição de uma arquitetura de controle através da qual o agente seleciona racionalmente o curso de suas ações.

Os pesquisadores Anand Rao e Michael Georgeff [RAO98], desenvolveram a lógica BDI com o propósito de fornecer um resumo da semântica ideal para agentes BDI, sobre o trabalho de pesquisadores tanto da Ciência da Computação Teórica como da Inteligência Artificial. Estes demonstraram como um tipo particular de lógica multimodal poderia ser usado para capturar os aspectos da teoria de Bratman [BRA90].

A teoria de Bratman é uma teoria do raciocínio prático, ou seja, o processo de raciocínio que todos nós experimentamos em nossas vidas diárias. Essa teoria concentra-se, em particular, na função que as intenções representam para raciocínio prático.

A arquitetura BDI clássica, apresentada na Figura 13, chamada de IRMA (*Intelligent Resource-bounded Machine Architecture*), proposta por [BRA90] possui como objetivo a viabilização do raciocínio do agente levando-se em consideração a sua limitação de recursos.

Esta arquitetura possui dois tipos de entidades: *processos* (representados por retângulos) e *entidades de armazenamento* (representadas por elipses). Desta forma, as intenções dos agentes são estruturadas em planos de alto nível, sendo que os planos que o agente conhece são armazenados

em uma biblioteca de planos. O *Analisador de Oportunidades* reage a mudanças no ambiente criando opções de ação baseada em eventos não previstos pelo planejamento convencional, que é realizado pelo *Raciocinador Meios-Fim*, o qual tem como entradas as *Crenças* do agente e os planos armazenados na *Biblioteca de Planos*. O *Raciocinador Meios-fim* e o *Analisador de Oportunidades* sugerem opções para o processo de filtragem, representado pelo *Filtro de Compatibilidade* e pelo *Ignorador do Filtro*. O *Filtro de Compatibilidade* verifica se as opções geradas são consistentes com as intenções adotadas atualmente, e as opções sobreviventes são passadas para o processo de *Deliberação*, que pesa as novas opções entre si e as incorpora nos planos do agente. O *Ignorador do Filtro* foi incorporado ao processo de filtragem devido a possibilidade do agente ter conhecimento limitado, que cria situações onde certas opções seriam interessantes apesar da base de crenças indicar que há inconsistência nestas opções [MEN02].

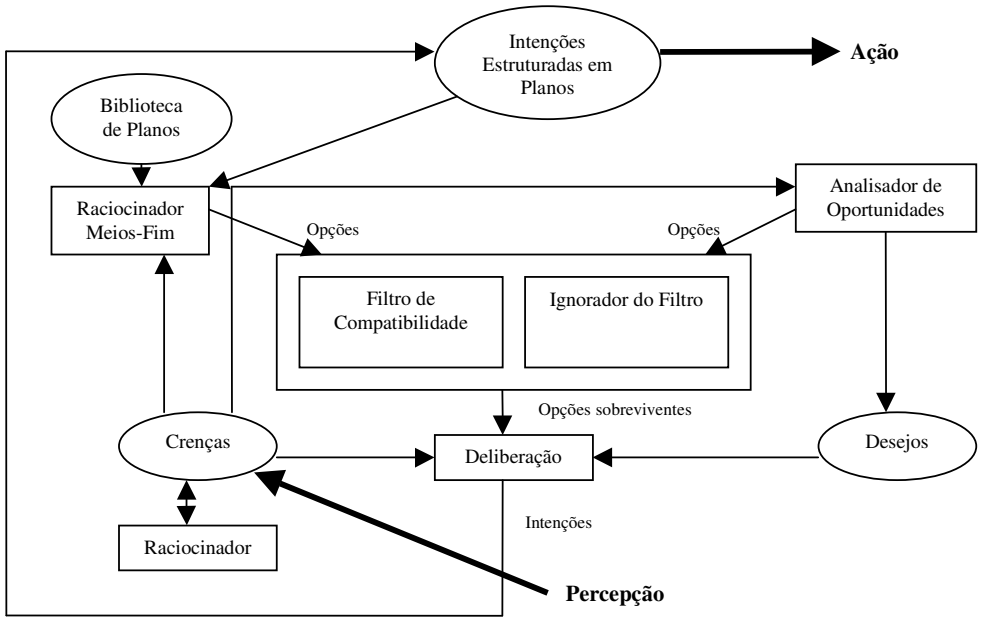


Figura 13 - Arquitetura BDI

Um trabalho relevante, que aborda este tópico, é o projeto Tropos [MYL01], o qual foi criado na premissa de que, para construir *software* a fim de operar dentro de um ambiente dinâmico, é necessário analisar e modelar explicitamente esse ambiente em termos de atores, objetivos e dependências de outros atores.

Esta metodologia suporta quatro fases no desenvolvimento de software: **requisitos iniciais**, que visam entender o problema estudando uma configuração organizacional, tendo como saída um modelo organizacional que inclui relevantes atores externos, seus respectivos

objetivos e suas interdependências; **requisitos posteriores**, onde o sistema é descrito dentro de um ambiente operacional juntamente com suas funções e qualidades relevantes; **projeto arquitetural**, onde a arquitetura global do sistema é definida em termos de subsistemas interconectados através de dados, controle e outras dependências e; **projeto detalhado**, onde o comportamento de cada componente arquitetural é definido em mais detalhes.

Neste projeto a análise prévia de requisitos é focada nas **intenções** das aplicações. Essas intenções são modeladas como objetivos que, através de uma forma de análise orientada a objetivo, eventualmente levam aos requisitos funcionais e não funcionais do sistema a ser modelado. Em *i** (framework de modelagem), proposto por [YU95], o qual sustenta intencionalidade distribuída, aplicações são representadas como atores sociais, que dependem de outros para atingir objetivos e realizar suas tarefas.

2.5.5 Formalismo Z

Z é uma linguagem formal cujo objetivo é a especificação de sistemas. Por ser uma linguagem formal ela possui: um alfabeto, uma sintaxe e uma semântica precisa [LUC95].

Woodcock [WOO96] considera a notação Z uma linguagem matemática com um mecanismo de estrutura poderoso. Em combinação com linguagem natural, este mecanismo pode ser usado para produzir especificações formais. Z é usada para descrever o comportamento corrente do sistema.

Uma especificação em Z consiste de um modelo matemático do estado do sistema e da definição de operações sobre o estado, descrevendo, também, certas operações correspondentes dessa estrutura. Essas operações correspondem às interações do sistema com o meio externo. Quando operações são aplicadas a componentes da estrutura lógica, o estado do sistema se altera, refletindo o efeito da interação no novo estado assumido pelo sistema. A notação Z pode ser usada também para descrever propriedades desejáveis, que os estados do sistema devem satisfazer. Isso permite que certas propriedades de sistemas especificados na notação Z possam ser validadas de maneira simbólica, usando apenas o texto das especificações, sem necessidade de testes sobre algum código compilado [MOU01].

Uma especificação em Z é formada de um número de esquemas (diagramas), os quais possuem uma parte declarativa e uma parte de predicados em lógica de 1ª ordem. Estes esquemas podem ser combinados e usados em outros esquemas [SPI98].

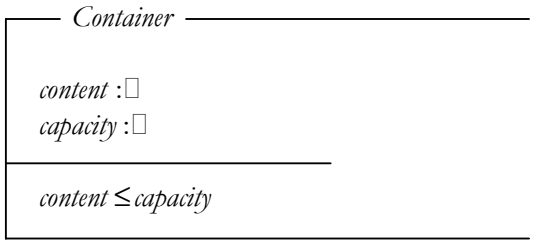


Figura 14 - Exemplo de um esquema em Z

A Figura 14 representa um esquema em Z, onde *Container* é o nome do esquema, a seguir são apresentados as declarações das variáveis do esquema e, no último bloco é colocado o predicado do esquema, no caso, ***contents*** ≤ ***capacity*** .

Conforme Moura [MOU01] extraindo-se a parte formal de uma especificação em Z, tem-se um conjunto mínimo de estruturas e operadores bem definidos que caracterizam completamente o espaço de estados do *software* especificado. Além disso, a parte formal da especificação provê também transformações nesse espaço de estados, as quais refletem as necessidades funcionais do *software*. Uma vez isolada, a parte formal, ou trechos dela, pode ser submetida a ferramentas computadorizadas que suportem, pelo menos, a verificação de propriedades do sistema especificado.

A formulação de um modelo em Z inicia como uma visão informal e intuitiva do cenário final em que se vai inserir o modelo final que se está construindo. Adquirir esta visão intuitiva faz parte do processo criativo em cujo âmbito se desenvolve toda a construção de uma especificação formal [MOU01].

A linguagem de especificação Z em particular foi adotada por D'Inverno [DIN97] como meio de construção de um *framework* formal de agente, o qual tem sido capaz de considerar vários aspectos distintos de agentes e SMAs.

Como benefícios da linguagem de especificação Z, pode-se citar [BAR94]:

- Permite precisão e não ambigüidade na definição de conceitos comuns;
- É suficientemente significativo para permitir uma consistência, descrição unificada e estruturada de um sistema de computador e suas operações associadas, permitindo assim, projetos alternativos de modelos particulares de agentes e sistemas para serem concebidos explicitamente, comparados e avaliados;

- Provê uma base para desenvolvimento de novos conceitos refinados. Através do uso de esquemas e inclusão de esquemas, Z facilita a descrição de sistemas em diferentes níveis de abstração.

Conforme Luck [LUC01] a linguagem de especificação Z já foi utilizada em outros trabalhos para fornecer análise do relacionamento social entre os agentes, para fornecer uma descrição de interações autônomas e para aplicar o modelo em sistemas existentes como meio de avaliar e comparar. Adicionalmente, o *framework* proposto por [LUC95] pode servir como ambiente para o desenvolvimento de uma metodologia de projeto de *software* baseado em agentes.

Utilizando o formalimo Z, [LUC01] propôs um *framewok* conceitual, como citado anteriormente, que serve para compreender e organizar o ambiente de sistemas baseados em agentes. Essa abordagem não é usada para restringir o uso da terminologia através de definições rígidas, mas é usada para prover uma infraestrutura abrangente que pode ser utilizada no entendimento da natureza dos diferentes sistemas. O benefício disso é que a riqueza da metáfora do agente é preservada através dos diversos usos, enquanto as identidades distintas de diferentes perspectivas são destacadas e usadas para direcionar e focar pesquisa e desenvolvimento de acordo com os objetivos particulares das sub-áreas.

Maiores detalhes sobre o formalismo Z, no que se refere a esquemas, descrições de operações, tipos, variáveis, operações com esquemas e um glossário da notação, podem ser visto no Anexo A.

3 ESPECIFICANDO AGENTES DE USUÁRIO

Baseando-se nos estudos descritos sobre formalismos na seção 2.5 do capítulo 2, pode-se dizer que, após uma análise, um problema detectado foi a falta de recursos por parte desses formalismos no momento da descrição de agentes de usuário, em particular, a descrição das três propriedades consideradas como básicas: aprendizagem, autonomia e personalização. Partindo-se dessa colocação, acredita-se que é possível explorar os aspectos do agente de usuário através de especificações em Z.

Justifica-se a utilização do formalismo Z pelo fato de a noção de esquemas facilitar o mapeamento para classes, já que existem outros trabalhos no projeto MASP nesta linha que podem ser mapeados. O formalismo também acrescenta precisão formal na especificação utilizando os esquemas de agente.

Salienta-se também, que é necessário partir-se de uma linguagem formal (especificações em Z, no caso desse trabalho) para que se possa estender mais formalmente os agentes. Assim, com as assertivas de uma especificação Z em linguagem de primeira ordem, torna-se mais fácil de agregar especificações sobre estados mentais.

A especificação Z também foi adotada por ser suficientemente expressiva para permitir uma consistência, unificar e estruturar a descrição de sistemas de computador e associar operações, estruturar especificações (que são feitas possivelmente por Z permitindo esquemas e inclusão de esquemas) e habilitar a descrição de sistemas em diferentes níveis de abstração.

3.1 Requisitos associados às propriedades dos agentes de usuário

Tendo o conjunto de propriedades já estabelecidas, na seção 2.2.1 do capítulo 2, definiram-se os requisitos dessas propriedades que serão observados no momento da especificação dos agentes de usuário. O formalismo Z deverá permitir a descrição desses requisitos através da formalização desses.

Para se chegar aos requisitos associados às propriedades dos agentes de usuário, partiu-se das características principais que cada propriedade apresenta e também se baseou em ações que os agentes podem tomar em relação ao usuário.

3.1.1 Aprendizagem

Na propriedade de aprendizagem partiu da uma definição dada na seção 2.2.2, baseando-se em estudos preliminares e, assim, foram extraídos os requisitos a seguir.

3.1.1.1 Requisitos sociais

Observar o usuário: observar as atuações do usuário no ambiente, tomando como ponto de partida, a reação desse em relação a determinadas situações e, se este segue sempre uma seqüência de atividades quando acessa o sistema/site. O que torna este social é a interação do agente com o usuário observando suas reações no ambiente;

Observar o ambiente[8]*:* observa as modificações ocorridas no ambiente a partir das ações realizadas pelo usuário. Considerando o ambiente como um "todo", supõe-se que o agente interage diretamente com outros agentes que compõe o sistema, o que torna este social;

Inferir tarefas e objetivos do usuário: explorar detalhadamente as tarefas e ações que o usuário está realizando no ambiente e, principalmente, qual o objetivo que este possui no momento da interação. Para que o agente realize essas tarefas é necessário que exista uma interação social entre este e o usuário;

Entender o ambiente[8]*:* entender as modificações que estão ocorrendo no ambiente no momento da interação do usuário, a fim de que ocorra o auxílio no processo de entendimento do usuário. Da mesma forma que no item anterior, é necessário que o agente interaja com o ambiente para que atinja seus objetivos, isso torna este requisito social.

3.1.1.2 Requisitos internos

Basear-se em experiências anteriores: armazenar informações de atuações anteriores para que possam ser utilizadas em tarefas futuras. Os agentes utilizam seu próprio conhecimento adquirido através de outras experiências realizadas em execuções anteriores. Isto o caracteriza como requisito interno;

Armazenar características que considera importante na sua interação: este requisito está associado diretamente com o anterior, pois o agente pode selecionar as características que poderá utilizar

[8] Ambiente: nessa situação, o ambiente é considerado como um todo. Ou seja, o ambiente (sociedade) é a união dos agentes com outros objetos inseridos no sistema.

nas próximas execuções. Somente o agente pode selecionar as características a serem armazenadas, por esse motivo é um requisito interno;

Aplicar informações armazenadas quando necessário: o agente poderá aplicar informações importantes que armazenou de outras execuções, quando encontrar tarefas ou usuários semelhantes. Para que essas informações sejam aplicadas, necessita conhecer os agentes, seus objetivos e tarefas que pretendem realizar, assim essa tarefa se torna do agente em particular.

3.1.2 Autonomia

Com base na definição de autonomia adotada para este trabalho (seção 2.2.3), definiu-se os requisitos listados a seguir.

3.1.2.1 Requisitos sociais

Conhecer o grau de autonomia do usuário: o agente deve reconhecer o quanto o usuário pode ser autônomo no ambiente. Ou seja, qual grau de liberdade que o usuário tem para atuar num sistema que possui restrições baseadas no tipo de usuário que irá utilizá-lo. O agente necessita manter contato (interação) diretamente com o usuário para que consiga conhecer o quanto este é autônomo no ambiente, assim é considerado um requisito social;

Escolher ação que vai executar: baseado na experiência que adquire através da interação com o usuário e no conhecimento "implementado" (pré-definido) que possui, opta por uma ação a executar. Assim, o agente é levado a tomar iniciativas, sempre baseado em informações que possui em seu banco de dados a respeito do usuário. Para que a escolha da ação que irá executar aconteça, além de possuir um conhecimento pré-definido, é necessário que exista interação com outros agentes (interação social), para que este adquira alguma experiência.

3.1.2.2 Requisito interno

Controlar suas ações: o agente deve controlar suas próprias ações, baseando-se nas atuações junto ao usuário e no conhecimento que possui sobre este. Somente o agente pode ter controle sobre suas ações. Ou seja, o controle depende somente deste em particular, caracterizando-o como um requisito interno. Deve haver ações que o agente não precisa de autorização do usuário para executar.

3.1.3 Personalização

Tomando como referência a definição dada para a propriedade de personalização neste trabalho (seção 2.2.4), definimos os seguintes requisitos:

3.1.3.1 Requisitos sociais

Interessar-se pelo usuário: o agente deve ficar atento às ações do usuário e apresentar interesse pelas atividades que o usuário realiza. Para que o agente se interesse pelo usuário é preciso que exista alguma interação entre eles (interação social), conseqüentemente um requisito social;

Manter informação sobre o usuário: o agente deve conhecer o usuário através da observação de suas ações, e concentrar-se nos objetivos, hábitos, interesses e preferências deste. O agente neste requisito necessita interagir com o usuário ou com outro agente, o que torna este social.

Agir de forma diferente com cada usuário: através da observação das ações do usuário o agente deverá agir sempre de forma a se adaptar a este, e assim, conseqüentemente, agirá de forma diferente com cada um. Nesse caso, assume-se que existe interação social entre eles, considerando como um requisito social.

3.1.3.2 Requisitos internos

Salienta-se que a propriedade de personalização não possui requisitos internos. Ou seja, todos os requisitos que esta possui são considerados como sociais neste trabalho. Justifica-se esta afirmação pelo motivo de, na personalização os agentes necessitarem interagir sempre com outros agentes e com o usuário para que consigam realizar suas tarefas e personalizar o conteúdo.

A partir da definição desses requisitos sociais e internos dos agentes de usuário, propõe-se uma especificação que permite expressá-los adequadamente, considerando sempre as três propriedades básicas: personalização, autonomia e aprendizagem.

Em particular, espera-se verificar a descrição dos requisitos sociais e internos dos agentes de usuário após a implementação de sistemas com essa abordagem.

3.2 Aplicação do formalismo

A proposta de formalização dos requisitos associados às propriedades dos agentes de usuário possui vínculo com o trabalho proposto por [LUC01]. Luck, em sua proposta, aplica métodos formais a fim de definir um *framework* para sistemas baseados em agentes. Este utiliza a especificação em Z para prover uma especificação formal acessível em sistemas baseados em agentes. Na sua formalização Luck [LUC01], dá enfoque para a propriedade de autonomia dos agentes, focalizando o que é necessário para um sistema ser autônomo em vários termos, distinguindo, objetos, agência e autonomia.

Assim, pode-se dizer que este trabalho segue a mesma linha do trabalho de [LUC01], pois este também utiliza a especificação Z para representar os agentes. No entanto, dá enfoque às três propriedades básicas dos agentes de usuário: aprendizagem, autonomia e personalização, no que tange aos seus requisitos.

A partir da definição dos requisitos dos agentes de usuário, propõe-se uma especificação que permite expressá-los adequadamente, considerando sempre as três propriedades básicas. Em particular, espera-se verificar a descrição dos requisitos sociais e internos dos agentes de usuário após a implementação de sistemas com essa abordagem.

No momento em que foi definida, de forma genérica, uma metodologia[9] inicial para a descrição do formalismo que expressa as propriedades básicas dos agentes de usuário, apresentou-se como proposta uma seqüência de passos que devem ser seguidos:

1. Primeiramente, como descrito na seção 3.1, definiram-se os requisitos das propriedades associadas aos agentes de usuário;

2. A partir daí, cada requisito associado às propriedades é representado através de um esquema;

3. Os requisitos são agrupados, em função das semelhanças que apresentam entre si;

4. Como estas propriedades são características básicas de um ou vários agentes de usuário; consequentemente, estes esquemas que as expressam são agrupados em um esquema que dá uma visão ainda mais geral, ou seja, o esquema de um agente de usuário genérico, por exemplo;

5. Finalmente, é necessário, para compor a formalização, um esquema que mostre o conjunto das crenças, desejos e intenções que um agente, no caso deste trabalho, deve possuir.

---

[9] Metodologia: observa-se que esta metodologia é somente uma proposta inicial, é o primeiro passo a fim de se definir uma metodologia completa.

A partir disso, no momento de formalizar os requisitos dos agentes de usuário, refinou-se essa metodologia inicialmente proposta, apresentada anteriormente, conforme mostrada a seguir:

1. **Primeiramente, define-se os requisitos do agente**

   - Requisitos de Aprendizagem

   - Requisitos de Autonomia

   - Requisitos de Personalização

2. **Logo após, representa-se os requisitos através de esquemas em Z**

Pode-se dizer que, para cada requisito a ser representado, tem-se um esquema descrevendo o estado das informações associadas ao requisito e um esquema descrevendo a mudança de estado (associada a uma ação efetuada pelo agente). Normalmente, os requisitos implicam em ações do agente, por isso, criamos esquemas representando mudanças de estado.

3. **E, finalmente, esquemas que possuem elementos em comum são agrupados; assim, elimina-se a redundância nos esquemas**

Esta etapa ocorre somente no caso de existirem esquemas redundantes que podem ser simplificados. Ou seja, nos esquemas que expressam ações similares, pode-se unir os predicados que as expressam. No entanto, enfoca-se que a inclusão de esquemas em outros esquemas não implica em agrupar estes e, sim, mostrar quais esquemas são necessários para complementar outros esquemas, em determinadas funções.

Salienta-se, que este formalismo satisfaz características básicas de um sistema. Dessa forma, no momento de especificar um sistema qualquer utilizando-se este formalismo, podem vir a surgir características consideradas como suplementares, com isso, estas podem ser inseridas através dos novos esquemas considerando-se como parte do conjunto de esquemas já existente.

Assumiu-se neste trabalho que os agentes de usuário são deliberativos; assim, os esquemas prevêem conjuntos de crenças, desejos e intenções.

Inicialmente na construção do formalismo que expressa os requisitos associados às propriedades básicas dos agentes de usuário (definidos na seção 3.1, do capítulo 3), estendeu-se o esquema *Agent* de [LUC01], o qual é chamado de *Agente* (Figura 18) neste trabalho. Para isso foi necessário, primeiramente, mostrar outros dois esquemas que o autor salienta serem importantes para definição do esquema *Agente* os esquemas das Figura 15 e Figura 17, respectivamente chamados de *Entidade* e *Objeto*.

Uma entidade é um componente que descreve o mais alto nível de abstração, apenas como uma coleção de atributos. Assim, apresenta-se inicialmente, o esquema *Entidade* (Figura 15), o que é um conjunto de **atributos**, com a restrição de que este é um conjunto não vazio.

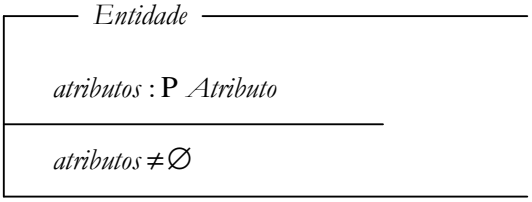


Figura 15 - Esquema *Entidade*

A Figura 16 apresenta o esquema *Acoes* (necessário para determinar o esquema *Objeto*), o qual é uma extensão do esquema *ObjectAct* proposto por [LUC01] em seu trabalho. Este possui uma função de seleção de ação chamada de **ato_objeto**, que determina qual o conjunto de ações é executado em seguida, num dado ambiente, definida como uma função total. Como predicado é definido que as próximas ações a serem realizadas pelo objeto (determinada pela aplicação da função *ato_objeto*) estão contidas nas capacidades.

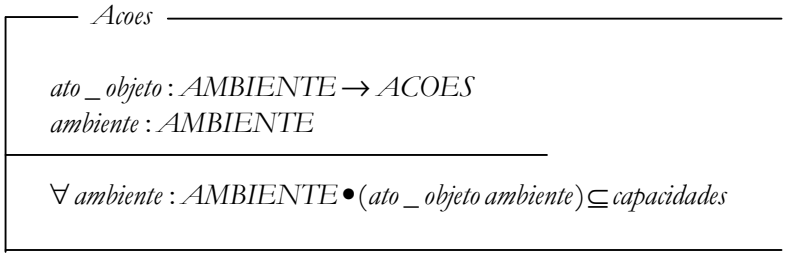


Figura 16 - Esquema *Acoes*

Na Figura 17 apresenta-se o esquema *Objeto*, representando uma extensão do esquema *Object* proposto por [LUC01]. Este inclui o esquema *Acoes* e formaliza a definição de um objeto, incluindo o esquema *Entidade*. Declara como variável as **capacidades** do tipo ACOES. Assim, apresenta-se o predicado que especifica as capacidades como não sendo um conjunto vazio.

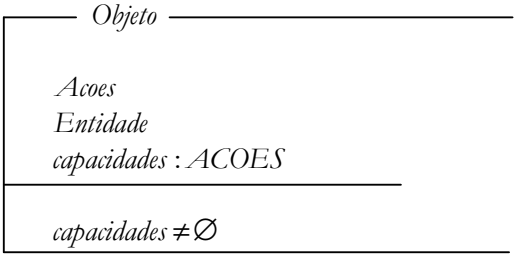


Figura 17 - Esquema *Objeto*

Com base nos esquemas anteriores, [LUC01] apresenta um esquema *Agent* chamado, neste trabalho, de *Agente* na Figura 18, que estendido e definiu-se como agente genérico para este trabalho. Este esquema inclui o esquema *Objeto*, apresentado anteriormente, e define uma variável **objetivos** que retorna um conjunto de objetivos. Este objetivos são definidos, na parte de predicados, como um conjunto de objetivos não vazio. Dessa forma, conforme [LUC01] no esquema *Agente*, é atribuído um conjunto de objetivos que retém alguma instanciação. Um objeto pode originar diferentes instanciações para os agentes.

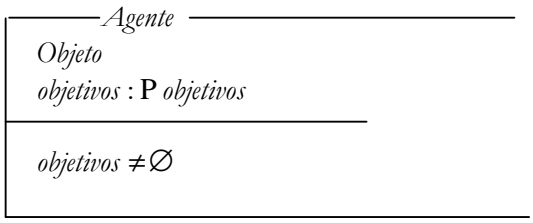


Figura 18 - Esquema *Agente*

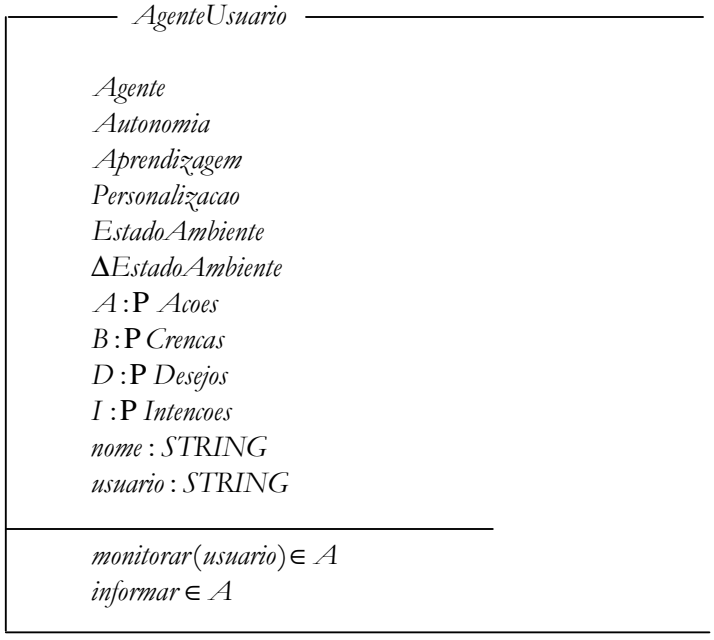


Figura 19 - Esquema *Agente de Usuário*

A Figura 19 apresenta o esquema *Agente de Usuário*, no qual estão inseridos (na parte de declarações) os esquemas *Agente, Aprendizagem, Autonomia, Personalização, EstadoAmbiente, ΔEstadoAmbiente* e, é apresentado um conjunto de ações, representado pela variável **A**, um conjunto de crenças representado pela variável **B**, um conjunto de desejos representado pela variável **D** e um conjunto de intenções, representado por uma variável **I**. Ainda são declaradas as variáveis, **nome** (nome do agente em questão) do tipo STRING e, **usuario** (usuário do sistema) também do tipo STRING. Este esquema possui dois predicados: o predicado onde a ação

monitorar usuário pertence ao conjunto de ações e, o predicado em que a ação **informar** também pertence ao conjunto de ações.

Na Figura 20, mostra-se o esquema *EstadoAmbiente*. Mesmo este esquema sendo ligado diretamente ao agente de usuário, pode-se dizer que também expressa os requisitos relacionados ao ambiente e ao usuário na propriedade de aprendizagem. Este esquema inclui o esquema *AcoesAgente* e possui como variáveis, **ambiente**, **interacoespos** (interações possíveis no ambiente) e **interacoesat** (interações atuais no ambiente) que representam um conjunto de interações e uma função **proximoa**, que representa o próximo conjunto de interações no ambiente a ser projetado. Este esquema expressa, através de seus predicados, as **interacoespos**, que descrevem algumas interações possíveis no ambiente corrente, as quais pertencem as interações no ambiente; **interacoesat**, que mostra um conjunto de interações que são atuais no ambiente. Ou seja, que fazem parte das interações possíveis no ambiente. Ainda apresenta-se o conjunto de ações do agente a ser projetado, as interações atuais do usuário e o ambiente como pertencentes ao esquema *AcoesAgente*.

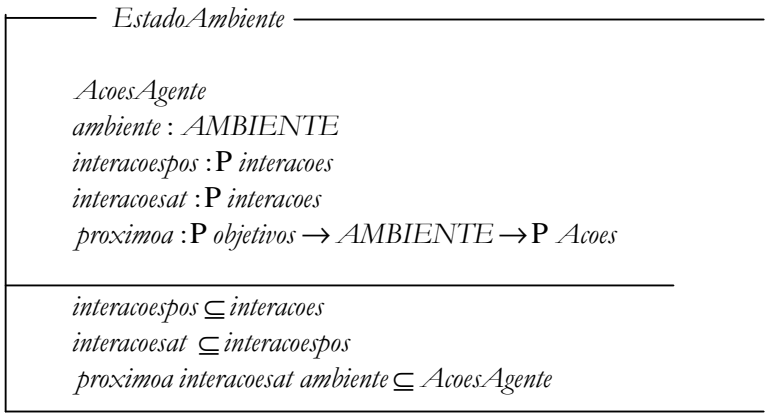


Figura 20 - Esquema *EstadoAmbiente*

O esquema Δ*EstadoAmbiente* representado na Figura 21, mostra a mudança de estado do esquema *EstadoAmbiente* (Figura 20), onde **EstadoAmbiente′**, representa o novo estado. Este possui também uma função chamada **atualizainteracoes**, que possui uma seqüência de ações que retorna as interações realizadas no ambiente. Consequentemente, no predicado **interacoesat** ocorre mudança de estado, onde as interações realizadas no ambiente são atualizadas no ambiente. Observa-se que a mudança de estado (novo estado) é representando por apóstrofe.

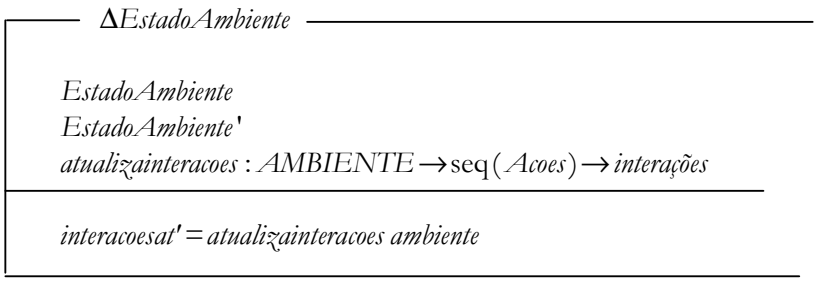


Figura 21 - Esquema Δ*EstadoAmbiente*

Em seguida, na Figura 22, é exposto o esquema *Aprendizagem*, que possui como função incluir o esquema que representa os requisitos associados a esta propriedade. O esquema incluído é *ObservarUsuario.*

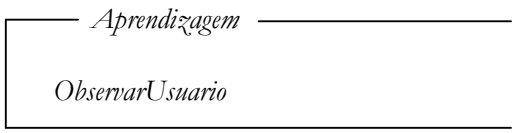


Figura 22 - Esquema *Aprendizagem*

Na Figura 23, é apresentado o esquema *Autonomia* que apresenta a propriedade de Autonomia, onde estão inseridos os esquemas que representam os requisitos associados a essa propriedade: *AcoesAgente* e *AutonomiaUsuario.*

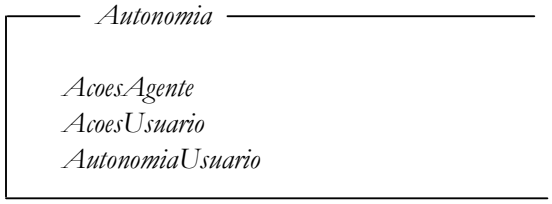


Figura 23 - Esquema *Autonomia*

A Figura 24, especifica o esquema *Personalização* (a propriedade de personalização do agente de usuário). Este esquema inclui os esquemas *EstadoUsuario*, Δ*EstadoUsuario* (mudança de estado do esquema *EstadoUsuario*) e *AcoesUsuario*, os quais expressam os requisitos associados a essa propriedade.

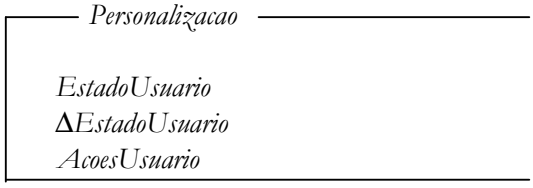


Figura 24 - Esquema *Personalização*

Os esquemas das Figura 22, Figura 23 e Figura 24, possuem a funcionalidade (como é visivelmente apresentado) somente de inserir os esquemas que compõem cada propriedade, definindo, assim, quais são os requisitos associados a cada uma delas.

O esquema *AutonomiaUsuario* (Figura 25) representa o requisito *Conhecer o grau de autonomia do usuário*. Ou seja, o agente deve saber qual grau de liberdade que o usuário tem para atuar num sistema que possui restrições baseadas no tipo de usuário que irá utilizá-lo. Assim este esquema inclui o esquema *AcoesUsuario* e declara variáveis como, **usuario** do tipo USUARIO, **agente** do tipo AGENTE e, **agentes** como um conjunto de agente. Como predicado é definido que, o agente não necessita de autorização para realizar suas ações. E, para toda ação (**a**) do tipo **AcoesUsuario** (que são determinadas como sendo ações que o usuário pode realizar), o conjunto de agentes pode autorizar o usuário a realizar a ação em questão. Esses predicados pertencem ao conjunto de crenças.

AutonomiaUsuario

$$
\begin{array}{l}
AcoesUsuario \\
usuario : USUARIO \\
nome : AGENTE \\
agentes : \mathbf{P}\ Agente \\
\hline
\forall a : AcoesUsuario \bullet autoriza(\{\}, (a)) \in B \\
\forall a : AcoesUsuario \bullet autoriza(agentes, usuario, a) \in B
\end{array}
$$

Figura 25 - Esquema *AutonomiaUsuario*

Na Figura 26, apresenta-se o esquema *EstadoUsuario*, que faz parte da propriedade de personalização. Este determina os requisitos da propriedade de personalização e os requisitos que competem ao usuário na propriedade de aprendizagem. Este possui como variáveis, **ambiente**, **AcoesUsuario** do tipo Acoes, **objetivospos** (objetivos possíveis do usuário) e **objetivosat** (objetivos atuais do usuário), que representam um conjunto de objetivos; **historico**, que mostra a seqüência de ações previamente projetada para um usuário e, uma função **proximo**, que é o próximo conjunto de ações do usuário a ser projetado. Ou seja, um conjunto de objetivos no ambiente que retorna um conjunto de ações do usuário. Como predicado tem-se: os **objetivospos**, que descrevem os objetivos possíveis no ambiente corrente, pertencendo aos objetivos do usuário, **objetivosat**, que mostra um conjunto de objetivos que são atuais para o usuário (que fazem parte dos objetivos possíveis do usuário), o conjunto de ações do usuário a ser projetado, os objetivos atuais do usuário e o ambiente que são declarados como pertencentes esquema *AcoesUsuario*.

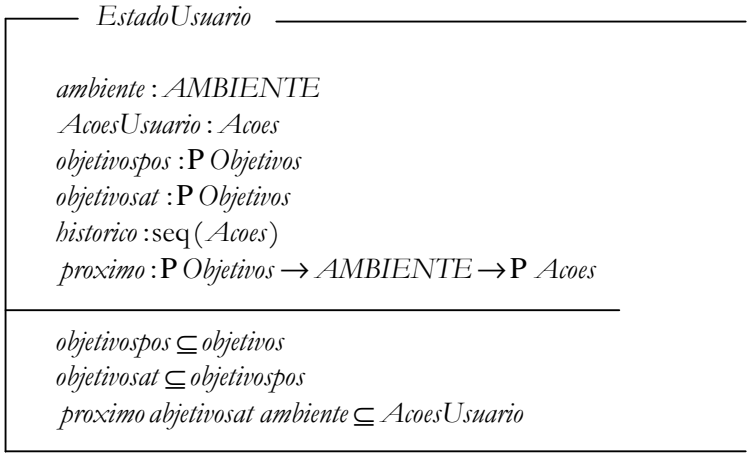


Figura 26 - Esquema *EstadoUsuario*

O esquema Δ*EstadoUsuário* representado na Figura 27, mostra a mudança de estado do esquema da Figura 26, onde **EstadoUsuario′**, representa o novo estado. Este possui uma função chamada **atualizaobjetivos**, que engloba uma seqüência de ações que retorna os objetivos do usuário no ambiente. Como no predicado em **objetivosat** ocorre mudança de estado, onde os objetivos do usuário são atualizados no ambiente e o histórico do usuário recebem atualização.

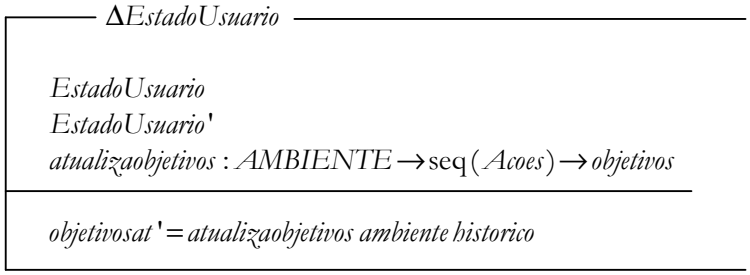


Figura 27 - Esquema Δ*EstadoUsuário*

No esquema *ObservarUsuario,* Figura 28, são representados todos os requisitos associados à propriedade de aprendizagem que dizem respeito o usuário. É inserido o esquema Δ*EstadoUsuário*, representando o novo estado do usuário, o qual também representa a observação que o agente faz em relação ao usuário. Consequentemente, o novo conjunto de objetivos atuais do usuário e a função **proximo** (que retorna o próximo conjunto de ações do usuário) e o ambiente também são declarados pertencentes ao conjunto de crenças.

A Figura 29, representa o esquema *AcoesAgente*, onde são mostrados os requisitos de *Escolher a ação que vai executar* e *controlar suas ações*, da propriedade de autonomia. Este esquema é uma extensão do esquema *AgentAct* proposto por [LUC01] e possui semelhança com o esquema *AcoesUsuario*. Este esquema inclui o esquema *Agente*, declara uma variável **ambiente** do tipo

AMBIENTE e, inclui a função **acoesagente**, que recebe os objetivos e o ambiente e retorna as ações do agente, que devem estar contidas nas capacidades.

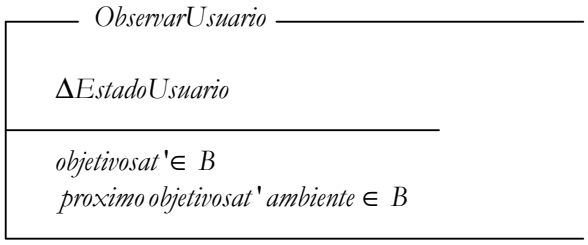


Figura 28 - Esquema *ObservarUsuario*

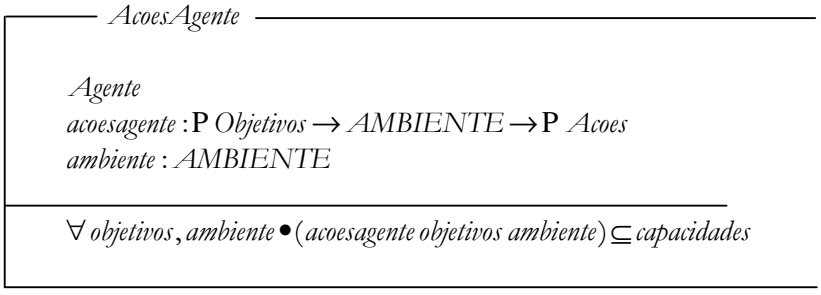


Figura 29 - Esquema *AcoesAgente*

A Figura 30, representa o esquema *AcoesUsuario*, que faz parte da propriedade de autonomia. Este esquema é semelhante ao esquema *AcoesAgente*. Este esquema inclui o esquema *Agente*, declara uma variável **ambiente** do tipo AMBIENTE e, inclui a função **acoesusuario**, que recebe os objetivos e o ambiente e retorna as ações do usuário, que devem estar contidas nas capacidades.

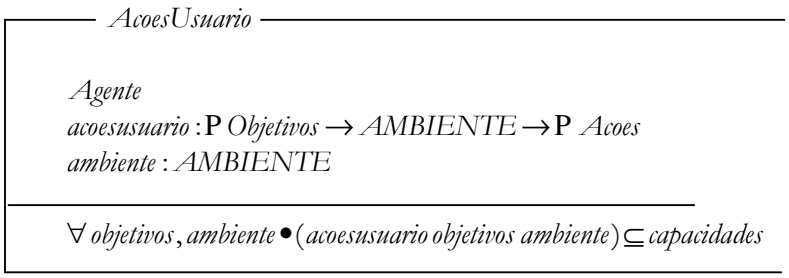


Figura 30 - Esquema *AcoesUsuario*

Salienta-se que, neste formalismo não existe a necessidade de agrupar esquemas similares. Dessa forma, na seção 4.2, do capítulo 4, desse trabalho, é apresentado um exemplo com a ocorrência de agrupamento de esquemas, realizando a terceira etapa da metodologia. Observa-se que, não há necessidade de sempre utilizar este recurso.

A partir da definição dos esquemas, apresenta-se na Figura 31, a organização desses esquemas conforme as inclusões realizadas. Dessa forma, as linhas (no sentido de cima para baixo) indicam a inclusão de esquemas em outros (herança de características de outros esquemas). Como exemplo, pode-se citar: o esquema *AgenteUsuario* inclui *Agente* que inclui *Objeto,* consequentemente, incluindo os esquemas *Entidade* e *Acoes*.

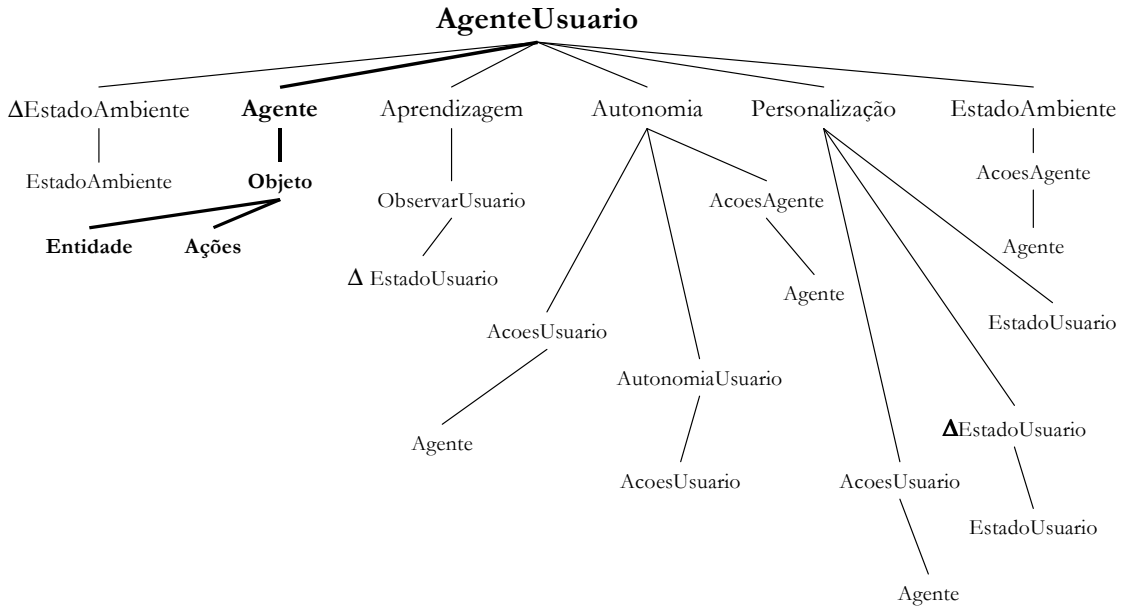


Figura 31 - Organização dos esquemas

A Tabela 2 apresenta as variáveis utilizadas no formalismo proposto com seus respectivos tipos utilizados e determinando a funcionalidade que possuem na formalização.

Tabela 2 - Definição dos símbolos utilizados no formalismo

Símbolo	Significado
≠	Diferente
∅	Conjunto Vazio
Π	Conjunto
∈	Pertence
Δ	Estado
→	Função total
⇒	Implicação
⊆	Subconjunto/Aplicar
•	Conjunção/Separar declaração
dom	Domínio
=	Igual
seq	Seqüência
Λ	Conjunção
′	Estado
V	Disjunção

Na Tabela 1 são apresentados os símbolos em Z utilizados no formalismo proposto com seu respectivo significado.

Tabela 3 - Definição dos tipos utilizados no formalismo

VARIÁVEL	TIPO	FINALIDADE
A	Π Acoes	Conjunto de ações
a	Acoes	Ações do agente/usuário
AcoesAgente	Acoes	Ações do agente (esquema)
Acoesagente	Função total	Conjunto de objetivos no ambiente retornando um conjunto de ações
Acoesusuario	Função total	Função que retorna um conjunto de ações
AcoesUsuario	Acoes	Ações realizadas pelo usuário
agentes	Π agentes	Conjunto de agente
ambiente	AMBIENTE	Ambiente do sistema
ato_agente	Função total	Função de seleção de ação
atributos	Π Atributos	Conjunto de atributos
atualizainteracoes	Função total	Seqüência de ações retornando interações no ambiente
atualizaobjetivos	Função total	Seqüência de ações retornando ações do usuário
autoriza	-	Função
B	Π Crenças	Conjunto de crenças
capacidades	ACOES	Capacidade do agente/usuário
D	Π Desejos	Conjunto de desejos
historico	Seq	Um seqüência de ações do agente/usuário
I	Π Intenções	Conjunto de intenções
interacoes	-	Interações no ambiente
interacoesat	Π Interacoes	Conjunto de interações atuais no ambiente
interacoesat'	-	Novo estado
interacoespos	Π Interacoes	Conjunto de interações possíveis no ambiente
nome	STRING	Agente em questão
objetivos	-	Objetivos do agente/usuário
objetivosat	Π Objetivos	Conjunto de objetivos atuais do usuário
objetivosat'	-	Novo estado
objetivospos	Π Objetivos	Conjunto de objetivos possíveis do usuário
proximo	Função total	Próximo conjunto de ações do usuário
proximoa	Função total	Próximo conjunto de ações no ambiente
usuario	STRING	Usuário do sistema

4 ESTUDO DE CASO

Para fins de avaliação deste trabalho utilizar-se-á o trabalho de [AND02], onde o agente de interface possui um papel preponderante no auxílio às interações entre o aluno e o ambiente. O contexto da aplicação é um STI multiagente cooperativo na WEB. A seguir descreve-se o resumo do trabalho de [AND02].

4.1 Trabalho utilizado

O enfoque principal do trabalho de Andrade centra-se no projeto de um agente pedagógico de diagnóstico para ambientes de aprendizagem colaborativa. O ambiente é formado por uma arquitetura que possui quatro agentes artificiais que irão dar suporte à aprendizagem à distância [AND01]. Os agentes integrantes do ambiente são: agente diagnóstico, agente mediador, agente semiótico e agente colaborador (social).

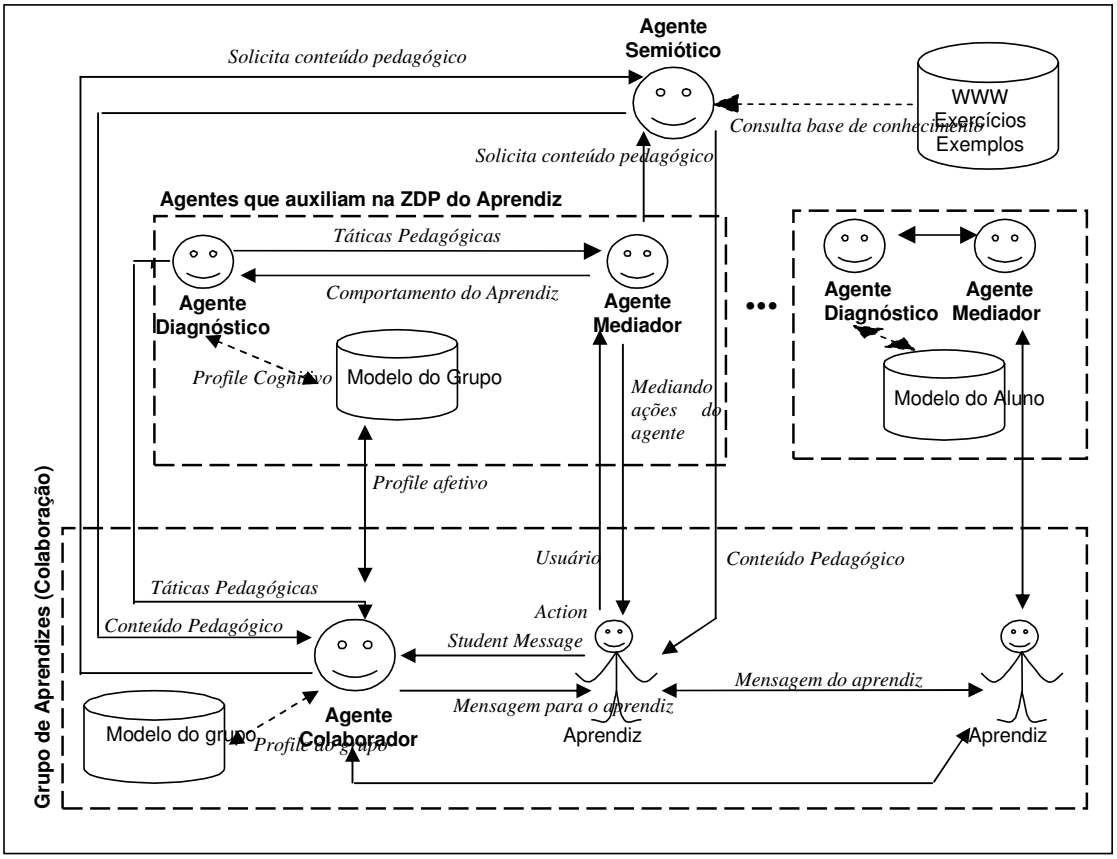


Figura 32 - Arquitetura multiagente do sistema [AND01]

Segundo [AND01], na arquitetura proposta os agentes artificiais têm como função monitorar e auxiliar os agentes humanos em suas atividades colaborativas. Esta arquitetura está sendo projetada independentemente do conteúdo e pode ser voltada a qualquer domínio de conhecimento. Todos os personagens envolvidos, aprendizes e agentes, são modelados como agentes colaboradores integrados em um ambiente de aprendizagem colaborativa. Os agentes diagnóstico e mediador monitoram e interagem com os aprendizes. Os agentes semiótico e colaborador auxiliam nas atividades relacionadas a toda a sociedade. O sistema possui um agente diagnóstico e um agente mediador para cada aluno e um agente semiótico e colaborador para toda a sociedade. A Figura 32 apresenta a arquitetura do sistema multiagente.

O **Agente Mediador** é o agente de interface entre o aprendiz e o sistema. A diferença central entre o agente mediador e o agente diagnóstico é que o primeiro realiza todas as tarefas de interface e comunicação com o usuário, como também monitora as interações entre os usuários. O agente mediador, além da função de mediar a interação do aprendiz com o agente diagnóstico, deverá auxiliar na predição de comportamentos deste usuário permitindo determinar as melhores ações a serem executadas para auxiliar no processo de aprendizagem do aluno.

O **Agente Semiótico** é responsável pelo envio de conteúdos para mediar a aprendizagem do aluno. Para que o agente mediador possa cumprir o seu papel, é necessária a intervenção de conteúdos adequados. Estes conteúdos são compostos por elementos semióticos, constituídos principalmente pela linguagem e signos. Estes elementos são introduzidos pelo agente semiótico para auxiliar na atividade cognitiva do aluno a fim de solucionar um dado problema, fazer uma análise ou pesquisa.

O **Agente Colaborador** conhece todos os agentes da sociedade, como também estes têm conhecimento da sua existência. É capaz de estabelecer a integração da sociedade e construir o modelo do grupo de aprendizes. Uma das atividades desse agente é investigar a existência de um outro aprendiz que tenha, além do conhecimento necessário, crenças e traços de personalidade que propiciem a melhor cooperação entre os aprendizes.

Como o foco do trabalho de Andrade [AND02] é o agente diagnóstico, ele será mais detalhado a seguir.

4.1.1 O Agente Diagnóstico

O agente diagnóstico possui, na sua modelagem, o conceito de Zona de Desenvolvimento Proximal (ZDP) defendida por Vygotsky, visando transformar habilidades potenciais em habilidades reais, buscando expandir a capacidade de desenvolvimento sócio-cognitivo do aluno.

De acordo com [AND02], o sistema possui um agente diagnóstico, com a função de descrever diagnósticos cognitivos, modelando o grupo e sugerindo táticas pedagógicas. Inicialmente, quando a tarefa é proposta para o grupo, o agente diagnóstico deve criar um mecanismo para avaliar quais habilidades estão na região do **Core**[10] ou que estão na região **ZDP**[11]. Para iniciar o processo de diagnóstico, este agente deve propor alguma tarefa para o aprendiz ou o grupo variando o grau de dificuldade.

Este agente define enquanto um agente do tipo *tutor* é capaz de observar o desenvolvimento real do aprendiz e propor atividades que tornem as suas habilidades o mais próximo possível do nível potencial (habilidades que estão ainda fora das aptidões do aluno, mas ele consegue realizar com o auxílio de um outro indivíduo mais capacitado). Como esse agente deve monitorar especificamente um aprendiz, haverá para cada aprendiz um agente diagnóstico que irá atuar na ZDP do aluno.

O agente diagnóstico trabalha o espaço entre crescimento espontâneo do aluno, sendo mediado por especialista que pode ser o professor, outro aluno mais experiente ou o próprio agente diagnóstico.

Para identificar a ZDP do aluno, o agente de diagnóstico deverá delimitar a área de aprendizagem em estudo e delimitar um domínio de conhecimento específico, para então definir seus desempenhos e competências.

Segundo Andrade [AND01], o agente diagnóstico é responsável por estimular àquelas funções do aprendiz que ainda não amadureceram, mas que estão em processo de desenvolvimento. Suas funções são as seguintes:

- Diagnosticar as habilidades e dificuldades dos alunos;
- Variar o grau de seu controle nas formas de organização das atividades conjuntas;
- Propor estratégias de ensino adaptadas ao perfil do aluno;

[10] Core: identifica o conhecimento que se possui, quais as habilidades que o aprendiz já possui.
[11] ZDP: identifica o que não se é capaz de fazer sozinho, mas pode-se fazer com algum suporte. É a distância entre o nível de desenvolvimento real e o potencial. No contexto desse trabalho, este conceito é utilizado a fim de parametrizar a formação de modelos de grupos.

- Alterar gradualmente os apoios, ajudas ou suportes na realização de determinadas tarefas;
- Aumentar o tempo dedicado a certos tipos de tarefas e diminuir o tempo dedicado a outras, de acordo com o desenvolvimento dos alunos;
- Modificar as formas de ajuda/suporte oferecidos, em função das dificuldades dos alunos, desde a atuação dirigida até o oferecimento de pistas (dicas), que vão permitir o trabalho autônomo do aluno;
- Observar os conhecimentos que já são familiares;
- Promover níveis diferentes de intervenção no momento em que o aluno já dominar certos conhecimentos;
- Sugerir atividades através de um processo gradual com constantes ajustes e modificações.

Um dos papéis do agente diagnóstico é estabelecer as táticas de ensino de acordo com uma estratégia pedagógica apropriada ao aluno. Para auxiliar na escolha da melhor estratégia e tática que se adeque ao aluno, o agente diagnóstico deve acessar e atualizar o modelo do aprendiz, identificando suas habilidades e deficiências, que será construído pela observação das ações do usuário. Desta maneira, ele é capaz de indicar a extensão na qual o aluno pode expandir-se, acessando e modificando este modelo.

O conhecimento sobre o aluno pode ser adquirido através de testes de aprendizagem, do mapeamento de *profiles* ou mapa conceitual de navegação do aluno, para a partir daí construir o modelo de aluno com as habilidades e dificuldades, como por exemplo seus questionamentos freqüentemente solicitados.

Além desses elementos citados, o modelo cognitivo do aluno controlado pelo agente diagnóstico leva em conta aspectos qualitativos da interação, focalizando as informações relevantes à interação social. Conforme [AND01] O modelo do aluno contém as seguintes informações:

- **Traços de personalidade:** A relação com os outros indivíduos depende da percepção que se tem do outro, influenciando significativamente a interação. Desta maneira, é necessário modelar também os seus traços de personalidade que irão determinar aspectos importantes da interação como, por exemplo, motivação, participação e afetividade.

- **Crenças do aluno:** As crenças dos alunos são deduzidas das interações. É importante ressaltar que em um grupo colaborativo devem ser modeladas, além das crenças individuais dos alunos, as crenças que possui em relação aos outros membros do grupo que podem ser cognitivas como também perceptuais (traços de personalidade).

4.1.2 A interação entre os agentes

Andrade [AND01] exemplifica o modelo de interação, destacando que inicialmente, o agente diagnóstico está diretamente relacionado com o agente colaborador (social) e com a crença que a interação com este agente pode favorecer o desenvolvimento intelectual do aluno.

Para favorecer o processo de internalização pelo aluno, o agente diagnóstico deverá interagir com o agente mediador permitindo trabalhar o conteúdo, as estratégias e as táticas de forma gradual para que o aluno possa percorrer todo o ciclo de internalização:

- **Fase um**: Assistência com um mediador mais capacitado;
- **Fase dois**: Auto-assistência do aluno;
- **Fase três**: Internalização;
- **Fase quatro**: Desautomatização, surgimento de novos conflitos. Reinício do ciclo.

A fim de que essas fases ocorram, quando o agente diagnóstico verificar que o aluno possui uma deficiência em algum conteúdo que está estudando e verificar a necessidade da assistência de um outro aprendiz, ele enviará uma mensagem ao agente colaborador solicitando o nome de um aprendiz que possua o perfil adequado. Quando o agente colaborador identificar essa pessoa, ele enviará uma mensagem ao agente diagnóstico. O agente diagnóstico repassará essa mensagem ao agente mediador que indicará ao aprendiz nomes de outros colegas com quem ele pode interagir para troca de idéias e explicações.

4.2 Especificação do Agente Diagnóstico

A especificação do Agente Diagnóstico realizou-se a partir da aplicação do formalismo proposto na seção 3.2, do capítulo 3. Juntamente com a aplicação do formalismo é utilizada a metodologia inicial proposta para especificar os requisitos das propriedades básicas dos agentes de usuário. A partir disso, a seguir apresenta-se a especificação realizada acerca do Agente Diagnóstico, mostrando, passo-a-passo, a sua construção através da metodologia.

1º Passo – Definição dos requisitos do agente

Como primeiro passo, definiu-se alguns requisitos que expressam as funções e ações do Agente Diagnóstico. Sendo que, parte destes foram especificados conforme o formalismo proposto. No entanto, os requisitos que não atendem o formalismo proposto, foram especificados de acordo com seu grau de exigência.

Os requisitos que compõem o Agente Diagnóstico e que representam sua funcionalidade são expostos a seguir, classificados de acordo com cada propriedade:

Requisitos de Aprendizagem

O Agente Diagnóstico deve:

1. Verificar se o aprendiz está ou não no nível ZDP a fim de sugerir táticas e escalar o Agente Mediador;

2. Registrar o diagnóstico do aprendiz.

Requisitos de Autonomia

O Agente Diagnóstico deve:

1. Registrar as dificuldades do aprendiz no módulo ZDP;

2. Registrar as habilidades do aprendiz no módulo *Core*;

3. Registrar as suas interações e dos outros agentes.

Requisitos de Personalização

O Agente Diagnóstico deve:

1. Registrar dificuldades ou habilidades no modelo do aprendiz, informados pelo Agente Mediador;

2. Se o aprendiz está em ZDP, o agente avalia as habilidades e sugere a ele algum nível de assistência (suporte);

3. Atualizar o modelo de grupo registrando as táticas aplicadas no grupo (registrar o histórico de suporte em futuras interações).

A partir disso, apresentam-se os esquemas organizados da seguinte forma: um esquema para cada requisito, sendo que, estes são detalhados de acordo com sua ocorrência, através da aplicação da metodologia.

Observa-se que, o **aprendiz**, o qual é referenciado no trabalho de [AND02] é representado pelo atributo **usuário** nessa especificação.

2º Passo – Representação dos requisitos através de esquemas em Z

Neste segundo passo, considerando os requisitos apontados no primeiro passo, definiu-se os esquemas que representam esses requisitos. Partindo-se de situações mais particulares para, posteriormente, agrupá-las em esquemas mais genéricos, os quais dão uma visão generalizada do Agente Diagnóstico em si.

Inicialmente, pode-se dizer que a especificação dos agentes em geral utiliza a formalização padrão, conforme proposto no capítulo3. Dessa forma, os esquemas utilizados inicialmente, neste estudo de caso, são: *Entidade, Acoes, Objeto* e *Agente*, pois estes são padronizados para qualquer formalização inicial de um sistema.

A Figura 33 apresenta o esquema *Agente de Usuário*, que neste estudo de caso representa o **Agente Diagnóstico**. Também neste estudo de caso o **aprendiz** [AND02], é representando por **usuário**, como segue no formalismo proposto. Este esquema foi criado anteriormente (seção 3.2), no entanto, para o melhor entendimento é mostrado novamente. Este esquema insere os esquemas *Agente, Aprendizagem, Autonomia, Personalização, EstadoAmbiente, ΔEstadoAmbiente* e apresenta um conjunto de ações (variável **A**), um conjunto de crenças (variável **B**), um conjunto de desejos (variável **D**) e um conjunto de intenções (variável **I**). Declara-se ainda as variáveis, **nome** (nome do agente em questão) do tipo STRING e, **usuario** (usuário do sistema) do tipo STRING. Este esquema mostra que a ação **monitorar** usuário e a ação **informar** pertencem ao conjunto de ações.

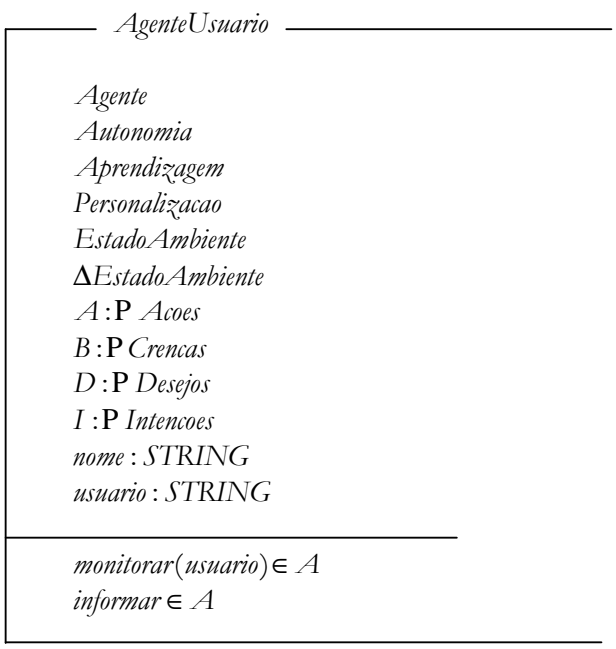


Figura 33 - Esquema *Agente de Usuário*

Em seguida, é exposto o esquema *Aprendizagem*, que insere os esquemas que representam cada requisito associado a esta propriedade: *ObservarUsuario, EstadoDiagnostico, ΔEstadoDiagnostico, Região* e *Δ Região* (vide Figura 34).

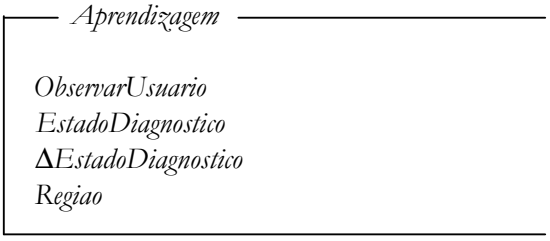


Figura 34 - Esquema *Aprendizagem*

Na Figura 35, é apresentado o esquema *Autonomia* representando a propriedade de Autonomia, que insere os esquemas que representam os requisitos associados a essa propriedade, tais como, *AcoesAgente, AcoesUsuario, AutonomiaUsuario, EstadoZDP, ΔEstadoZDP, EstadoCore* e *ΔEstadoCore*.

A Figura 36, representa o esquema *Personalização* (propriedade de personalização), sendo que, este inclui os esquemas *EstadoUsuario* e Δ*EstadoUsuario*, expressando os requisitos associados à essa propriedade.

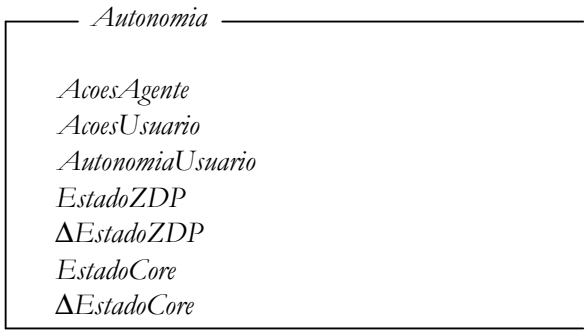


Figura 35 - Esquema *Autonomia*

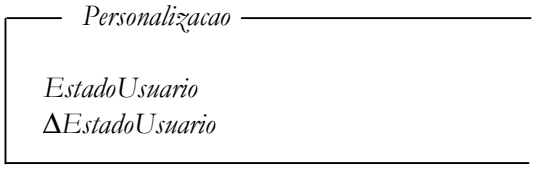


Figura 36 - Esquema *Personalização*

Na Figura 37, é apresentado o esquema *EstadoZDP*, que expressa o requisito *Registrar as dificuldades do aprendiz no módulo ZDP*. Neste esquema é inserido o esquema *AcoesUsuario.* As variáveis declaradas são: **a** como um conjunto de ações, **dificuldades** como um conjunto de dificuldades, **suporte** do tipo SUPORTE e, **usuario** do tipo USUARIO. Como predicados apresentam-se que, para todo usuário, tendo como base o seu conjunto de ações e dificuldades, deve-se atualizar seu estado no nível ZDP, num determinado tempo. Assim o agente é capaz de fornecer algum tipo de suporte ao usuário. Estes pertencem ao conjunto de crenças do agente.

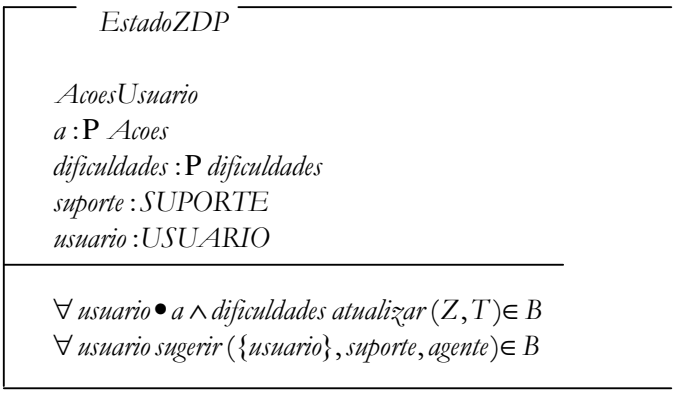


Figura 37 - Esquema *EstadoZDP*

O esquema Δ*EstadoZDP* representado na Figura 38, mostra a mudança de estado do esquema da Figura 37, onde EstadoZDP', representa o novo estado e possui uma função **atualizadif**, que atualiza as dificuldades do usuário, através do esquema *EstadoUsuario* que retorna

as dificuldades deste. Assim, nos predicados tem-se a mudança de estado das dificuldades, onde as dificuldades do usuário são atualizadas no *EstadoUsuário*.

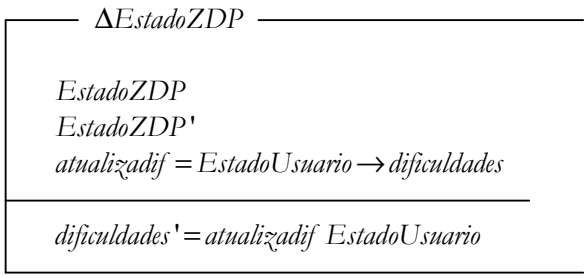


Figura 38 - Esquema Δ*EstadoZDP*

O esquema *EstadoCore* (Figura 39), expressa o requisito *Registrar as habilidades do aprendiz no módulo Core*. Como este esquema possui semelhanças com o esquema *EstadoZDP*, também é inserido o esquema *AcoesUsuario*. As variáveis declaradas são: **A** como um conjunto de ações, **habilidades** como um conjunto de habilidades e, **usuario** do tipo USUARIO. O predicado desse esquema (pertence ao conjunto de crenças do agente) mostra que, para todo usuário, baseando-se no seu conjunto de ações e habilidades, deve-se atualizar seu estado no nível Core, num determinado tempo.

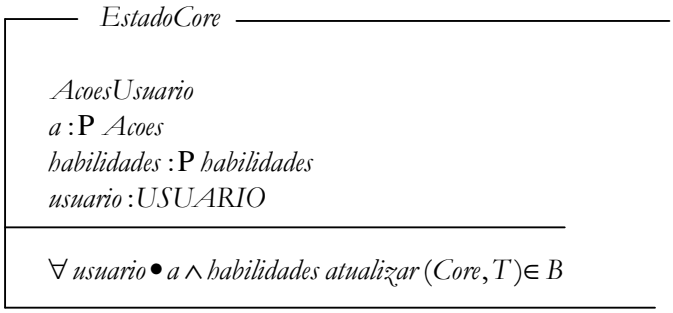


Figura 39 - Esquema *EstadoCore*

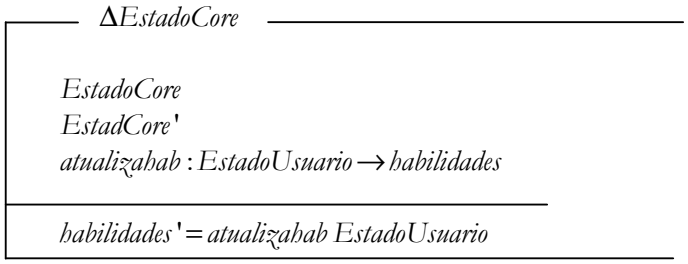


Figura 40 - Esquema Δ*EstadoCore*

O esquema Δ*EstadoCore* representado na Figura 40, mostra a mudança de estado do esquema da Figura 39, onde EstadoCore', representa o novo estado e possui uma função **atualizahab**, a qual atualiza as habilidades do usuário, através do esquema *EstadoUsuario* que retorna as habilidades deste. Também ocorre a mudança de estado das habilidades, onde as habilidades do usuário são atualizadas no *EstadoUsuário.*

Em seguida, são utilizados esquemas considerados como padrão na formalização proposta neste trabalho. Dessa forma, somente são citados os esquemas (da seção 3.2) que constituem a seqüência. Estes esquemas são os seguintes: *EstadoUsuario,* Δ*EstadoUsuario*, que representam os requisitos de **Registrar habilidades ou dificuldades no modelo do aprendiz**; **Se o aprendiz está em ZDP, o agente avalia as habilidades e sugere a ele algum nível de assistência (suporte)** e; **Registrar o histórico de suporte em futuras interações**; *AcoesAgente*, representando o requisito de **Registrar as interações do Agente Diagnóstico com outros agentes**; e os demais esquemas são necessários para a complementação da especificação.

O esquema *ObservarUsuario* (Figura 41), representa o requisito *Verificar se o aprendiz está ou não no nível ZDP a fim de sugerir táticas e escalar o Agente Mediador*. Este insere o esquema Δ*EstadoUsuário*, representando o novo estado do usuário, o qual também representa a observação que o agente faz em relação ao usuário. Consequentemente, o novo conjunto de objetivos atuais do usuário são declarados como pertencentes ao conjunto de crenças e a função **proximo**, que retorna o próximo conjunto de ações do usuário e o ambiente também pertencem ao conjunto de crenças.

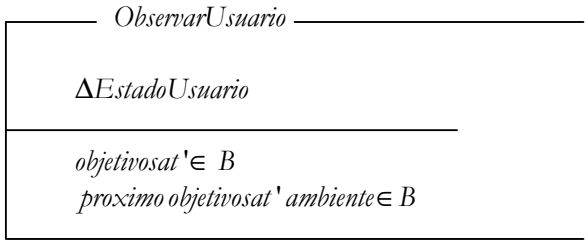


Figura 41 - Esquema *ObservarUsuario*

A Figura 42 representa o esquema *EstadoRegiao*, que representa também o requisito *Verificar se o aprendiz está ou não no nível ZDP a fim de sugerir táticas e escalar o Agente Mediador*. Neste esquema é definida a região em que o aluno se enquadra: ZDP ou Core. Assim, primeiramente, são declaradas as variáveis, **ZDP** (conjunto de dificuldades), **Core** (conjunto de habilidades), **a** um conjunto de ações e**, nome** (nome do agente) do tipo AGENTE. Como predicado tem-se que,

para toda região, representada por **R**, pertencente ao aluno, o agente deve consultar as ações a serem realizadas, que compõe a região ZDP ou a região Core.

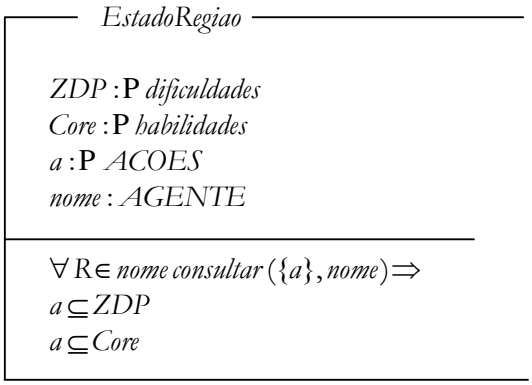


Figura 42 - Esquema *Regiao*

O esquema Δ*EstadoRegiao* representado na Figura 43, mostra a mudança de estado do esquema da Figura 42. Ou seja, as mudanças de região (ZDP ou Core) que ocorrem com o usuário. O EstadoRegiao', representa o novo estado e possui uma função **atualizareg**, que atualiza a região em que está o usuário, através do esquema *EstadoUsuario* que retorna a região desse. Também ocorre a mudança de estado da região (ZDP ou Core), onde a região em que se encontra o usuário é atualizada no *EstadoUsuário*.

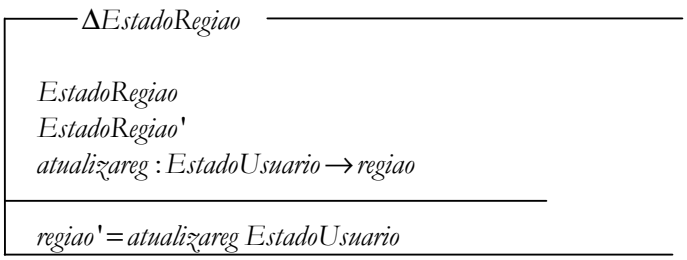


Figura 43 - Esquema Δ*EstadoRegiao*

A Figura 44 (esquema *EstadoDiagnostico)* define o requisito *Registrar o diagnóstico do aprendiz*, que insere os esquemas *AcoesAgente* e *EstadoUsuario*, e declara as variáveis: **nome** do tipo AGENTE e, **usuario** do tipo USUARIO. Expressando que, para todo aluno pertencente a ZDP ou Core, o agente deve processar o diagnóstico (pertencente ao conjunto de desejos do agente).

O esquema Δ*EstadoDiagnostico* representado na Figura 45, mostra a mudança de estado do esquema da Figura 44, que possui uma função **atualizadiag**, atualizando o diagnóstico do usuário, através do esquema *EstadoUsuario* retornando o diagnóstico desse. Ocorre ainda a mudança de estado do diagnóstico, onde o diagnóstico do usuário é atualizado no *EstadoUsuário*.

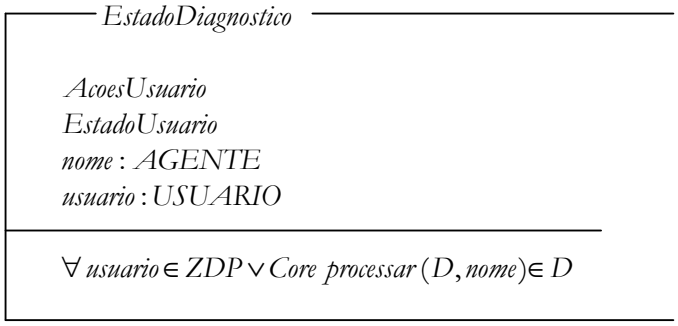


Figura 44 - Esquema *EstadoDiagnostico*

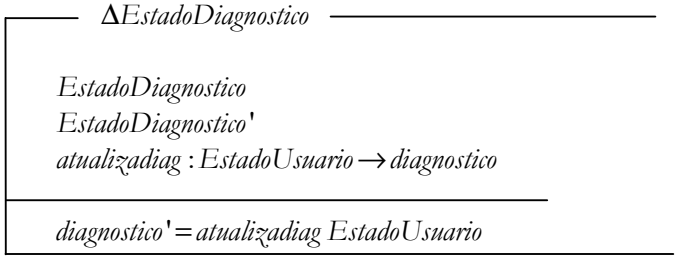


Figura 45 - Esquema Δ*EstadoDiagnostico*

A partir da definição dos esquemas, apresenta-se na Figura 46, a estruturação dos esquemas no estudo de caso, conforme as inclusões realizadas. Os esquemas em grifo são os novos esquemas criados em relação ao formalismo proposto. As linhas (no sentido de cima para baixo) indicam a inclusão de esquemas em outros (herança de características de outros esquemas), conforme já mencionado na Figura 31.

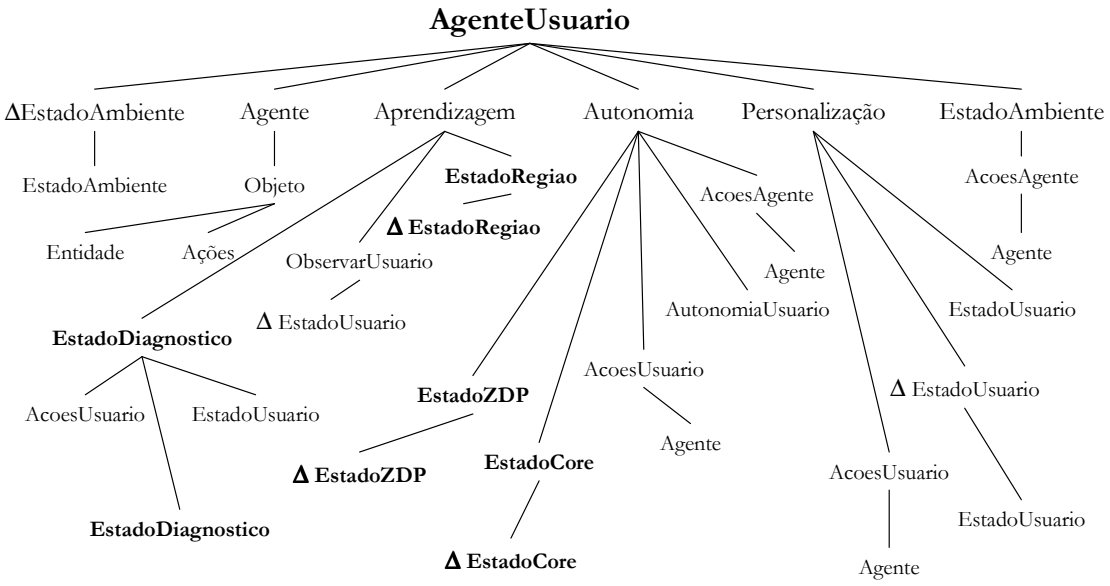


Figura 46 - Estruturação dos esquemas

3º Passo – Agrupamento de esquemas similares (eliminação de redundâncias)

Um exemplo de agrupamento que foi realizado neste estudo de caso ocorreu com as seguintes táticas:

- Registrar as dificuldades do aprendiz no módulo ZDP
- Se o aprendiz está em ZDP, o agente avalia as habilidades e sugere a ele algum nível de assistência (suporte)

Estes requisitos são expressos em um só esquema. O esquema *EstadoZDP*, conforme exemplo mostrado na Figura 47.

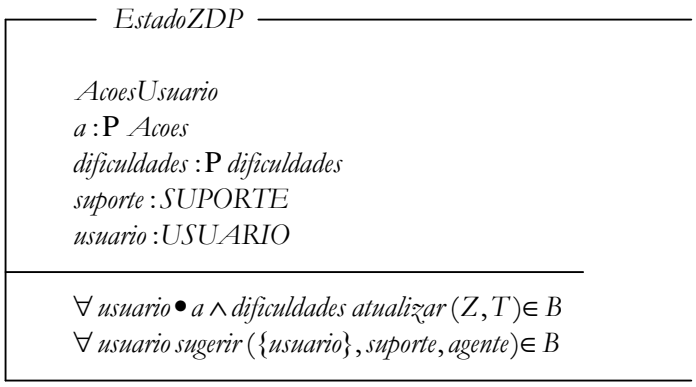


Figura 47 - Exemplo de agrupamento de esquemas

Assim, a medida em que os esquemas são desenvolvidos e trabalhados, pode haver a necessidade de agrupamento, devido o número muito grande de esquemas que um sistema pode gerar e também por facilitar o entendimento e evitar duplicidades.

A Tabela 4, apresenta as variáveis utilizadas no estudo de caso, com seus respectivos tipos utilizados e determinando a funcionalidade que possuem na formalização. As variáveis que constam são aquelas que não foram referenciadas na Tabela 2 (seção 3.2, capítulo 3).

Tabela 4 - Definição dos tipos utilizados no estudo de caso

VARIÁVEL	TIPO	FINALIDADE
atualizar	-	Função
atualizadif	Função total	Função que atualiza as dificuldades do usuário
atualizahab	Função total	Função que atualiza as habilidades do usuário
consultar	-	Função
Core	-	Nível Core
D	-	diagnóstico
dificuldades′	-	Novo estado
estadousuario′	-	Novo estado
dificuldades	Π Dificuldades	Conjunto de dificuldades
habilidades	Π Habilidades	Conjunto de habilidades
habilidades′	-	Novo estado
processar	-	Função
suporte	SUPORTE	Suporte fornecido ao aprendiz
ZDP	-	Nível ZDP

Os símbolos em Z utilizados no estudo de caso com seu respectivo significado são mencionados na Tabela 3, da seção 3.2, capítulo 3.

5 CONSIDERAÇÕES FINAIS

Ainda persistem problemas na questão de especificação de agentes de usuário. Em SMAs, necessita-se definir as propriedades dos agentes a fim de melhor especificar sua arquitetura, decorrente de seu papel dentro da sociedade. Dessa forma, verificamos a necessidade de especificar os agentes de usuário, a partir dos requisitos associados às propriedades identificadas/selecionadas como básicas.

Para este trabalho a questão de pesquisa norteadora determinada foi:

Quais as propriedades básicas do agente de usuário devem ser consideradas quando da definição de uma linguagem para especificação formal de agentes?

Pode-se responder esta questão através da proposta desenvolvida neste trabalho.

Inicialmente, através de levantamento bibliográfico rigoroso, detectou-se a necessidade de determinar algumas propriedades básicas para agentes de usuário: aprendizagem, autonomia e personalização. Levando-se em consideração que os agentes de usuário ainda são uma área com estudos iniciais e, na literatura não existe um conceito clássico nem um conjunto de propriedades definidas especialmente para esses.

Consequentemente, decidiu-se abordar os requisitos associados a estas propriedades. Ou seja, definiu-se para cada propriedade um conjunto de requisitos que melhor representam cada uma. Dessa forma, chegou-se à conclusão de que é necessário o levantamento de requisitos para que seja realizada uma posterior especificação ou modelagem.

A partir dos requisitos, necessitou-se propor uma metodologia inicial a fim de facilitar a especificação dos requisitos associados a cada propriedade. Esta metodologia é composta de uma seqüência de passos que também deixa mais claro o entendimento da especificação. Através dessa, é possível refinar a especificação/formalismo, de tal forma que fique sucinto e de fácil aplicação em diversos sistemas com a tecnologia de agentes de usuário. No entanto, salienta-se que, é uma metodologia inicial e necessita de aprimoramentos para ser considerada completa.

Com as propriedades, requisitos e metodologia definidos, aplicou-se o formalismo Z sobre os requisitos, utilizando-se a metodologia, e chegou-se a um conjunto de esquemas capazes de expressar tais requisitos. Assim, a partir desse conjunto de esquemas propostos foi necessário validá-los a fim de verificar sua aplicabilidade num sistema com a utilização de agentes de usuário.

Neste estudo de caso foi possível detectar que, além dos requisitos inicialmente definidos, cada aplicação possui um conjunto em particular de requisitos, os quais, às vezes, não se aplicam sobre os requisitos apresentados nas propriedades. Assim, são definidos requisitos complementares. A partir daí, cada um desses é representado através de esquemas, e, como mencionado na metodologia, estes podem ser agrupados de acordo com adaptabilidade e semelhança, a fim de evitar redundâncias. Assim, outra contribuição que o trabalho traz está ligada à notação e a utilização do formalismo Z para esta finalidade.

Levando em consideração esses aspectos descritos, acredita-se que todos os objetivos considerados na seção 1.5, do capítulo 1, foram contemplados e, inclusive, este foram complementados com a idéia de propor, além de outras coisas, uma metodologia inicial para a especificação formal de requisitos. Considera-se, também, que se alcançaram os resultados esperados para esta pesquisa. Contudo, as alternativas e propostas aqui apresentadas são promissoras e podem ser ampliadas.

Acredita-se que a proposta é uma contribuição também para o projeto de pesquisa MASP, como já colocado na seção 1.6 do capítulo 1. A proposta apresenta facilidade para unir trabalhos existentes, tanto trabalhos utilizando representação através de estados mentais quanto trabalhos utilizando representação visual. Em trabalhos que utilizam modelagem AUML torna fácil a migração pois, os esquemas em Z são de fácil mapeamento para classes em AUML. E, como pode ser visto, em Z é permitida a inclusão de esquemas em outros esquemas, tendo assim, semelhança com herança em AUML. Já em representação através de estado mentais, a arquitetura BDI descreve o processamento interno do agente utilizando estados mentais, juntamente com lógica multimodal. Esses conceitos também são de fácil adaptação a especificação Z, onde, o conjunto de crenças, desejos e intenções pode ser expresso para especificar um agente, utilizando lógica de 1ª ordem. Dessa forma, a especificação formal realizada possibilita que os trabalhos referenciados com outros trabalhos do projeto MASP que seguem na mesma linha, sejam possíveis de serem migrados para essa especificação.

Uma das maiores dificuldades encontradas foi a não disponibilização de um ambiente próprio para construção dos esquemas em Z. Dessa forma, a construção dos esquemas demandou bastante tempo pois, sem uma ferramenta própria, não foi possível facilitar a construção dos esquemas e verificar com precisão e em pouco tempo a consistência do conjunto de esquemas.

Outra dificuldade apresentada foi em relação aos aspectos funcionais onde é preciso representar como o agente irá se comportar e não dizer qual a função do agente, o que, na

verdade, são os requisitos. Ou seja, dessa forma é possível entender como é difícil entender como é difícil modelar os requisitos dos agentes.

A contribuição ainda não abordada foi a extensão do modelo proposto por Luck [LUC01], o qual especifica agente utilizando o formalismo Z, mas apresenta somente a propriedade de autonomia e, neste trabalho, além dessa, são consideradas as propriedades de aprendizagem e personalização.

Com essas considerações apontadas espera-se que o formalismo seja diferenciado no grupo de pesquisa tanto em novas pesquisas quanto na extensão de pesquisas já existentes.

Trabalhos futuros

Como todo trabalho de mestrado, algumas questões se mantém em aberto e o resultado possui um conjunto de limitações, em função de prazo estabelecido para o desenvolvimento deste.

A validação da especificação exigiria continuar o projeto e implementação do agente especificado, a fim de verificar a utilidade do formalismo como ferramenta para guiar o desenvolvimento e conferir se os requisitos estão satisfeitos. Assim, um trabalho futuro é a continuidade do estudo de caso.

Durante o estudo de caso, não foi modelada a interação entre o agente diagnóstico e o grupo, em função do prazo. Na versão atual do trabalho de Andrade [AND02], as atribuições relativas ao grupo estão vinculadas ao agente diagnóstico, mas é possível que migrem no futuro para o agente social.

Durante o trabalho, pensou-se em especificar como ocorrem as mudanças de estado do agente através de esquemas. No entanto, constatou-se que os requisitos não fazem referência direta a essas mudanças de estado, portanto os esquemas correspondentes não foram incluídos na modelagem final. A questão do quanto se deve detalhar os requisitos é ainda um tema ainda em aberto.

A metodologia proposta ainda é um esboço (metodologia inicial), precisando ser submetida a mais experimentação. No entanto, há indícios de que a proposta pode ser generalizada para outras classes de agentes (agentes de domínio, de infra-estrutura, etc.). Acreditamos que essa extensão pode ser realizada no futuro, ainda no contexto do projeto MASP.

6 REFERÊNCIAS BIBLIOGRÁFICAS

[AND01] ANDRADE, A.F. **Agente pedagógico de diagnóstico para mediar a aprendizagem colaborativa inspirado em uma concepção.** Porto Alegre: CPGCC/UFRGS, 2001. (Proposta de Tese de Doutorado)

[AND02] ANDRADE, A.F. (et al). Cognitive diagnosis to support group learning. In: **Starting Artifical Intelligence Reasearchs Symposium (STAIRS 2002)**, *Proceedings...* July, Lyon, France, p. 129-139, 2002.

[BAR94] BARDEN, R. (et al.). **Z in practice.** New York: Prentice Hall, 1994.

[BAS98] BASTOS, R.M. **O planejamento de alocação de recursos baseado em sistemas multiagentes.** Porto Alegre: CPGCC/UFRGS, 1998. (Tese de Doutorado)

[BON88] BOND, A.H., GASSER, L.(eds.) **Readings in distributed artificial intelligence.** San Mateo: Morgan Kauffman, 1988.

[BRA90] BRATMAN, M. **Intetion, plans and practical reason.** Cambridge: Harvard University Press, 1990.

[BRO87] BROOKS, R. A. Intelligence without representation. In: **Artificial Intelligence**, v. 47, n.2, p.139-159, out. 1987.

[CHA98] CHAIB-DRAA, B. Agent communication language: towards a semantics based on success, satisfaction and recursion. In: **5th International Workshop, (ATAL'98).** *Proceedings...* France, p. 232-245, 1998.

[COH87] COHEN, P., LEVESQUE, H. J. Intention = Choice + Commitment. In: **Sixteen National Conference on Artificial Intelligence**. *Proceedings...* Springer Verlag, 1987.

[COH90] COHEN, P.R., LEVESKE, H.J. Intention is choice with commitment. In: **Artificial Intelligence**. v. 42 , n. 2, p. 231-261, dez. 1990.

[COR94] CORRÊA, M. **A arquitetura de diálogos entre agentes cognitivos dinstribuídos.** Rio de Janeiro: UFRJ, 1994. (Tese de Doutorado)

[DEM90] DEMAZEAU, Y., MÜLLER, J. Descentralized artificial intelligence. In: **First**

European Workshop on Modeling Autonomous Agents in a Multi-Agent World. *Proceedings...* Cambridge: North-Holland, p. 105-120, 1990.

[DIN97] D'INVERNO, M. (et al). Formalisms for multi-agent systems. In: **Journal of Knowledge Engineering Review.** v. 12, n. 3, p. 315-321, set. 1997.

[DRO00] DROGOUL, A. Robots as a social species: the MICRobES project. In: **Symposium Of Socially Intelligent Agents: The Human In The Loop (SIA'2000)**, *Proceedings...* AAAI Fall Symposium, AAAI Press (technical report FS-00-04), p. 254-269, 2000.

[DUR96] DURFEE, E. H. Planning in distributed artificial intelligence. In: O'HARE, G.M.P., JENNINGS, N.R. (eds.) **Foundations of distributed artificial intelligence.** New York: John Wiley & Sons, 1996.

[ETZ94] ETZIONI, O., WELD, D. A softbot-based interface to the internet. In: **Communications of the ACM.** v.37, n.7, p. 212-234, ago. 1994.

[FER99] FERNANDES, P.L. Avaliação de desempenho através de redes de autômatos estocásticos. In: **Escola Regional de Informática da Região Sul (ERI 1999)**. *Anais...* São Leopoldo, SBC, p. 52-63, 1999.

[FIN94] FININ, T. (et al). KQML as an agent communication language. In: **The 3rd International Conference on Information and Knowledge Management**. *Proceedings...* ACM Press, p. 149-164, 1994.

[FON97] FONER, Leonard. Yenta: a multi-agent, referral-based matchmaking system. In: **The First International Conference on Autonomous Agents (Agents '97)**, Califórnia, Marina del Rey, p. 321-335, 1997.

[GAS01] GASSER, L. MAS Infrastructure Definitions, Needs, Prospects, In: WAGNER, T., RANA, O. **Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems.** Barcelona: Springer-Verlag, 2001.

[GOU01] GOULART, R.R.V. **Utilizando a tecnologia de agentes na construção de Sistemas Tutores Inteligentes em ambientes interativos.** Porto Alegre: PPGCC/PUCRS, 2001. (Dissertação de Mestrado)

[HAY85] HAYES-ROTH, B. A blackboard architecture for control. **In: Artificial Intelligence,** v. 26, n. 3, p. 229-242, jun. 1985.

[HEW77] HEWITT, C. Viewing control structures on patterns of passing messages. **In: Artificial Intelligence**, v.8, n.2, p.152-170, 1977.

[HUB00] HÜBNER, J.F., SICHMAN, J.S. SACI: uma ferramenta para implementação e monitoração da comunicação entre agentes. In: **Brazilian Symposium on Artificial Intelligence - Sbia-Iberamia'2000.** *Proceedings...* São Paulo: ICMC/USP, p. 117-128, 2000.

[IGL98] IGLESIAS, C., GARIJO, M., GONZALES, J.C. Analysis and design of multiagent system using MAS-CommonKADS. In: **Agent theories, Architectures, and Languages**, v.22, n.3, p.132-141, nov. 1998.

[JAM01] JAMERSON, A., WITTIG, F. Leveraging data about users in general in the learning of individual user models. In: **Seventeen International Joint Conference on Artificial Intelligence.** *Proceedings...* San Francisco, p. 127-141, 2001.

[JEN96] JENNINGS, N.R. Coordination techniques for distributed artificial intelligence. In: O'HARE, G.M.P., JENNINGS, N.R. (eds.) **Foundations of distributed artificial intelligence.** New York: John Wiley & Sons, 1996.

[JEN98a] JENNINGS, N.R., WOOLDRIDGE, M.J. Applications of intelligent agents. In: JENNINGS, N.R., WOOLDRIDGE, M.J. (eds). **Agent technology:** foundations, applications, and markets. New York: Springer, 1998.

[JEN98b] JENNINGS, N., SYCARA, K., WOOLDRIDGE, M.. A roadmap of agent research and development. In: **Autonomous Agents and Multi-Agent Systems**, Kluwer Academic Press, 1998.

[JEN99] JENNINGS, N.R. Agent-oriented software engineering. In: **Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World: Multi-Agent System Engineering (MAAMAW-99)**. *Proceedings...* Germany: Springer-Verlag, p. 99-109, 1999.

[KON00] KONZEN, A.A, FROZZA, R. Clever Tutor: an intelligent tutor system applying a hybrid strategy of teaching. In: **IASTED - International Conference on Computers and Advanced Technology in Education (CATE 2000).** *Proceedings...* Cancun, México, p. 148-158, 2000.

[KON01a] KONZEN, A.A., OLIVEIRA, F.M. **Aplicações de sistemas multiagentes em comércio eletrônico.** Porto Alegre: PPGCC/PUCRS, 2001. (Trabalho Individual I)

[KON01b] KONZEN, A.A., OLIVEIRA, F.M. **Um estudo de caso em modelagem de sistemas multiagentes para comércio eletrônico.** Porto Alegre: PPGCC/PUCRS, 2001. (Trabalho Individual II)

[KON01c] KONZEN, A.A., GIRAFFA, L.M.M.; OLIVEIRA, F.M. **Em direção a uma linguagem de especificação para agentes de usuário.** Porto Alegre: PPGCC/PUCRS, 2001. (Plano de Estudo e Pesquisa)

[KON02] KONZEN, A.A., GIRAFFA, L.M.M.; OLIVEIRA, F.M. **Especificação de requisitos de agentes de usuário em Z.** In: SEMINÁRIOS DE ANDAMENTO 2002. *Anais*... Porto Alegre: PPPGCC/PUCRS, 2002.

[LAM00] LAMPORT, L. **A summary of TLA+.** California, USA: Digital, 2000.

[LAM94] LAMPORT, L. The temporal logic of actions. In: **ACM Transactions on Programming Languages and System.** v. 16, n. 30, p. 156-173, jun. 1994.

[LAM97] LAMPORT, L. **The operators of TLA+.** California, USA: Digital, 1997.

[LAM99] LAMPORT, L. **Specifying concurrent systems with TLA+.** Calculational System Design, 1999.

[LUC01] LUCK, M., d'INVERNO M. A conceptual framework for agent definition and development. In: **The Computer Journal**. *Proceedings*... v. 44, n.1, p. 1-20, out. 2001.

[LUC95] LUCK, M., d'INVERNO M. A formal framework for agency and autonomy. In: **1st International Conference on Multi-Agent Systems, (ICMAS'95)**. AAAI Press, p. 112-122, 1995.

[MAE00] MAES, P., RHODES, B.J. Just-in-time information retrieval agents. In: **IBM Systems Journal**, v. 39, n. 3 & 4, p. 215-232, dez. 2000.

[MAE94] MAES, P. Modeling adaptative autonomous agents. In: **Artificial Life Journal**, v.1, n. 1 e 2, p. 59-70, Mit Press, mai. 1994.

[MEN02] MENEGUZZI, F., GIRAFFA, L.L. **Raciocínio BDI.** Porto Alegre: PPGCC/PUCRS, 2001. (Trabalho Individual I)

[MID02] MIDDLETON, S.E. **Interface agents: a review of the field.** Southampton: University of Southampton, 2000. N. ECSTR-IAM01-001, ISBN: 0854327320 (Technical Report)

[MON00] MONARD, M.C., SICHMAN, J.S. Advances in Artificial Intelligence. In**: Lecture Notes in Artificial Intelligence**, Berlin-Alemanha, Springer Verlag, v. 1952, n. 3, p. 253-271, ago. 2000.

[MOR98] MÓRA, M. (et al). BDI models and systems: reducing the gap. In: **Agents Theory, Architeture and Languages Workshop**. *Proceedings*... Canarias: Springer-Verlag, p. 215-227, 1998.

[MOU01] MOURA, A.V. **Especificações em Z: uma introdução.** Campinas: Editora da Unicamp, 2001.

[MOU96] MOULIN, B., BRAHIN, C. An overview of distributed artificial intelligence. In: O'HARE, G.M.P., JENNINGS, N.R. (eds.) **Foundations of distributed artificial intelligence.** New York: John Wiley & Sons, 1996.

[MUL97] MÜLLER, J., et. al. Methodologies for multi-agent systems. In: SINGH, M.P.; RAO, A.S., WOOLDRIDGE, M. (eds.) **Intelligent agents IV: agent theories, architectures and languages.** *Proceedings*... Rhode Island, USA, p. 274-289, 1997.

[MYL01] MYLOPOULOS, J., KOLP, M., CASTRO, J. UML for agent-oriented software development: the tropos proposal. In: **Fourth International Conference on the Unified Modeling Language: UML 2001**, Canadá, Toronto, p. 54-69, 2001.

[NOG00] NOGUEIRA, L. **Information Retriel: uma abordagem baseada em agentes.** Porto: I. S. de Engenharia do Porto, 2000. (Dissertação de Mestrado)

[O'HA96] O'HARE, G.M.P. Agent Factory: an environment for the fabrication of multiagent systems. In: O'HARE, G.M.P.; JENNINGS, N.R. (eds.) **Foundations of distributed artificial intelligence.** New York: John Wiley & Sons, 1996.

[PAR97] PARAÍSO, E.C. **Concepção e implementação de um sistema multi-agentes para monitoração e controle de processos industriais.** Curitiba: CPGEEII-CEFET, 1997. (Dissertação de Mestrado)

[PRE95] PRESSMAN, R.S. **Engenharia de software.** São Paulo: Makron Books, 1995.

[RAO98] RAO, A.S., GEORGEFF, M.P. Decision procedures for BDI logics. In: **Journal of Logic and Computation**, v. 8, n. 3, p. 110-124, set. 1998.

[RIC93] RICH, E. **Inteligência Artificial.** São Paulo: Makron Books, 1993.

[SHO93] SHOHAM, Y. Agent Oriented Programming. In: **Journal of Artificial Intelligence.** v. 60, n. 1, p.51-92, mar. 1993.

[SIC95] SICHMAN, J.S. **Du raisonnement social chas les agents: une approche fondée sur la théorie de la dépendance.** Laboratoire d'Informatique Fondamentale et d'Intelligence Artificielle, Institut National Polytechnique de Grenoble, 1995. (Tese de Douturado)

[SIL99] SILVA, A. **Agentes de software na Internet**. Centro Atlântico, Lisboa, 1999.

[SMI80] SMITH, R.G. The contract net protocol: high-level communication and control in a distributed problem solver. In: **IEEE Transactions on Computers**. v.29, n.12, p. 134-152, dez. 1980.

[SOU01] SOUZA, J.B. **Em direção a uma abordagem para a Modelagem de um Sistema Multiagente.** Porto Alegre: PPGCC/PUCRS, 2001 (Dissertação de Mestrado).

[SPI98] SPIVEY, J.M. **The Z notation: a reference manual.** Second Edition. Oriel College, Oxford: Prentice Hall International, 1998.

[WIN00] WINCKLER, M.A. (et al). Interação entre aprendiz e computador: métodos para desenvolvimento e avaliação de interfaces. In: **IV Workshop de Informática na Educação.** Porto Alegre, 2000.

[WOO00a] WOOD, M., DELOACH, S.A. An overview of the multiagent systems engineering methodology. In**: First International Workshop on Agent-Oriented Software Engineering.** *Proceedings...* Springer Verlag, Berlin, p. 142-160, 2000.

[WOO00b] WOOLDRIDGE, M. JENNINGS, N.R. KINNY, D. The Gaia methodology for agent-oriented analysis and design. In**: Journal of Autonomous Agents and Multi-Agent Systems.** v. 2, n.1, p. 213-225, out. 2000.

[WOO02] WOOLDRIDGE, M. **An introduction to multiagent systems.** Chichester, England: John Wiley & Sons, 2002.

[WOO96] WOODCOCK, J., DAVIES, J. **Using Z: specification, refinement, and proof.** New York: Prentice Hall, 1996.

[WOO99] WOOLDRIDGE, M. JENNINGS, N.R. KINNY, D. A methodology for agent-oriented analysis and design. In: **Third International Conference on Autonomous Agents (AGENTS'99).** *Proceedings*... Seattle, WA, p. 63-81, 1999.

[YU01] YU, E.S.K. Agent-Oriented Modelling: Software versus the World. In: **The Second International Workshop on Agent-Oriented Software Engineering (AOSE'2001).** *Proceedings*... Montreal, Canadá, p. 45-59, 2001.

[YU95] YU, E.S.K. **Modelling strategic relationships for process reengineering.** Canadá: Departament of Computer Science, University of Toronto, 2001. (PhD. Thesis)

[ZAM01] ZAMBERLAM, A.O. **Em direção a uma técnica para programação orientada a agentes BDI.** Porto Alegre: PPGCC/PUCRS, 2001. (Dissertação de Mestrado)

ANEXO A

ESPECIFICAÇÕES EM Z - NOÇÕES BÁSICAS

ESPECIFICAÇÕES EM Z – NOÇÕES BÁSICAS

Z é uma notação, na medida em que congrega convenções para apresentar certos textos matemáticos, de forma a facilitar a descrição de sistemas de computação. A precisão de uma especificação em Z vem do fato de que a notação Z é baseada na noção matemática de conjunto. A simplicidade do uso da notação Z vem do fato de que essa realmente repousa sobre noções matemáticas elementares. Mais especificamente, a notação Z faz uso de noções de conjuntos e princípios básicos de lógica matemática.

Uma especificação em Z é uma expressão do comportamento do sistema que serve de apoio para se demonstrar propriedades desejáveis do seu comportamento futuro, além de servir como um guia no processo de codificação, que se iniciará em seguida. As especificações em Z são formadas por parágrafos que contém uma especificação formal, alternados com parágrafos textuais, os quais servem de apoio e dão mais clareza à notação formalizada. Ambos os itens são partes integrantes da especificação, um suportando o outro. Em parte, essa mescla de formalismo com texto desmistifica a notação Z, quando idéias pré-concebidas poderiam levar à imagem de que toda especificação em Z fosse constituída apenas por um grande conjunto de fórmulas matemáticas. Os parágrafos textuais são parte importante da especificação. Apenas a seqüência de parágrafos formais de uma especificação em Z, embora descreva o sistema modelado completamente, é, em certo sentido, pobre em comunicabilidade. A especificação completa, com seus parágrafos textuais, tabelas e figuras, é que se constitui num instrumento útil para a compreensão da especificação por outros interlocutores humanos. Criar um balanceamento adequado entre a parte formal e a parte verbalizada de uma especificação é uma arte cujo domínio vem apenas com a experiência.

1. Esquemas

A especificação em Z pressupõe duas partes bem definidas, representadas através de esquemas, conforme figura 1.

- Uma primeira parte declarativa, onde as variáveis de estado são declaradas, e indica-se também as variáveis de entrada e saída.
- Uma segunda parte predicativa, onde constrói-se predicados que exprimem a relação exata que deve vigorar entre as variáveis de estado do sistema no instante anterior e no

instante posterior à execução da operação em foco. É ainda nesse trecho da especificação que indica-se as pré-condições que devem ser satisfeitas pelos valores iniciais das variáveis envolvidas na operação.

A relação entre valores anteriores e valores posteriores das variáveis de estado caracteriza o efeito de cada operação executada pelo sistema. Já a interação do sistema com o meio circundante se dá através das variáveis de entrada e das variáveis de saída. As variáveis de entrada permitem que o sistema altere seu comportamento em função de valores externos. As variáveis de saída são o canal de retorno que permite que o sistema devolva ao meio externo alguns resultados da transformação de estado.

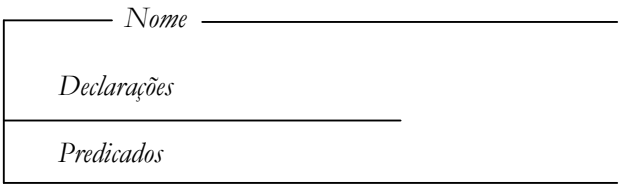


Figura 1 - Definição de um esquema em Z

O esquema da Figura 1, também pode ser representado da seguinte forma:

Nome ≅ [Declarações | Predicados]

A seguir, na Figura 2 apresenta-se um exemplo de esquema em Z, chamado de *Pessoal*. Onde é definida a variável **empregados**, definida como conjunto de **PESSOA**; uma função **chefe_de**, do tipo **PESSOA** que retorna uma lista de pessoas e; uma função **salario**, pertencente ao conjunto dos números naturais que retorna uma lista de pessoas. Este esquema possui como predicado, a relação de que todos os funcionários possuem salário menor do que o chefe.

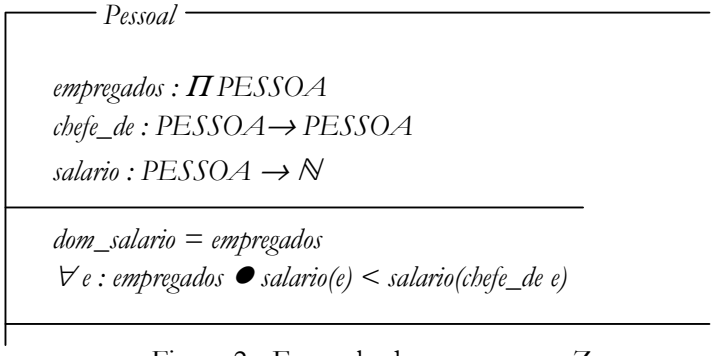


Figura 2 - Exemplo de esquema em Z

2. Descrição de operações

Existem algumas operações básicas utilizadas nos esquemas em Z, as quais são constantemente referenciadas:

? : Caráter inicial quando se que apresentar uma variável de entrada. Exemplo: nome?

! : Caráter final quando se que apresentar uma variável de saída. Exemplo: nome!

′ : apóstrofe representa o estado posterior a operação, ou seja, o novo estado. Exemplo: nome′.

Para representar o estado atual a operação ou o estado anterior, apresenta-se a variável sem utilização de símbolos. Exemplo: nome.

3. Tipos

Toda especificação em Z contará com tipos iniciais. Esse tipos iniciais designam conjuntos pré-definidos cuja estrutura interna não é detalhada. Vale apenas a condição de que os conjuntos são disjuntos dois a dois, ou seja, nenhum deles tem qualquer elemento em comum com os demais.

Um tipo inicial é declarado em Z simplesmente encerrando-se seu nome entre os símbolos **[ ]**. Como exemplo tem-se a declaração:

[PESSOA]

Assim, mostra-se que admite-se a existência desse conjunto e não são fornecidos detalhes sobre sua estrutura interna.

Ainda pode-se dizer que um esquema pode ser o tipo de um objeto:

$$p1; p2 : Pessoal$$

Esse tipo pode ser denotado por:

$$empregados : P\ PESSOA;$$

$$chefe_de : PESSOA \rightarrow PESSOA;$$

$$salarios : PESSOA \rightarrow N$$

Os objetos de um tipo esquema devem satisfazer as restrições desse esquema. E a seleção de uma declaração é realizada com o operador "**.**".

$$p1.empregados \cap p2.empregados$$

As operações sobre conjuntos são definidas só entre conjuntos e elementos de um mesmo tipo. Como exemplo pode-se apresentar: pertence ($\in$), não pertence ($\notin$), união ($\cup$), interseção ($\cap$), contido ($\subset$), subconjunto ($\subseteq$,), entre outros.

3.1. Definição de tipos

Enumeração

Nos tipos enumerados todos os elementos nomeados são distintos e fazem parte do tipo e nenhuma outro elemento faz parte do tipo., conforme exemplos a seguir:

$$SEXO ::= masculino \mid feminino$$

$$GRAU ::= graduacao \mid mestrado$$

Estruturação

Na estruturação os elementos estão estruturados da seguinte forma, de acordo com o exemplo abaixo: toda pessoa ou é aluno ou é professor.

$$PESSOA ::= discente \ll ALUNO \gg$$

$$\mid docente \ll PROF \gg$$

No exemplo a seguir, especifica-se que se for aluno é pessoa do tipo discente; se for professor é pessoa do tipo docente e para toda pessoa existe um aluno do tipo discente ou um professor do tipo docente.

$$\forall al : ALUNO \ (\exists p : PESSOA \bullet p = discente\ al)$$

$$\forall pr : PROF \ (\exists p : PESSOA \bullet p = docente\ pr)$$

$$\forall p : PESSOA \ (\exists al : ALUNO \bullet p = discente\ al)$$

$$\vee (\exists pr : PROF \ p = docente\ pr)$$

Produto cartesiano

$$SEXO \ X \ GRAU = \{(feminino; graduacao);$$
$$(feminino; mestrado);$$
$$(masculino; graduacao);$$
$$(masculino; mestrado)\}$$

$$(feminino; mestrado) \in SEXO \ X \ GRAU$$

Conjunto das partes

P conjunto das partes:

$$P \ GRAU = \{ \{ \}, \{graduacao\}, \{mestrado\}, \{graduacao, mestrado\} \}$$

P1 conjunto das partes menos o conjunto vazio:

$$P_1 \ GRAU = \{ \{graduacao\}, \{mestrado\}, \{graduacao, mestrado\}\}$$

F conjunto das partes finitas:

$$Responsavel \in F \ (PROF \ X \ CURSO)$$

4. Variáveis

Quando tem-se tipos iniciais já declarados, pode-se construir declarações para variáveis. Em z, variáveis são declaradas construindo uma lista de seus nomes, separados por vírgula, terminando a lista com o símbolo de dois-pontos e, escrevendo em seguida o nome do tipo que será associado às variáveis declaradas. Como exemplo, pode-se citar a criação de duas variáveis *u* e *v*, do tipo CODIGOS, como é mostrado abaixo:

$$u, v : \text{CODIGOS}$$

Salienta-se que em Z, uma variável pode assumir um valor genérico dentro de um conjunto de valores possíveis especificados por seu tipo. Mesmo que, em alguns pontos da especificação talvez ainda não se conheça o valor assumido pela variável, esse valor é bem definido e delimitado pelo conjunto de tipos.

5. Operações com esquemas

Existem várias convenções e operadores eu podem ser usadas juntamente com a noção esquemas em Z. Essas operações se comportam como se fosse criada uma álgebra, ou cálculo, envolvendo essas estruturas. Essas convenções e operações visam facilitar a tarefa de construir especificações em Z. Também contribuem para tornar as especificações mais claras, facilitando sua leitura, interpretação e manutenção. A seguir são citadas as operações mais importantes nas especificações em Z.

5.1. Inclusão

A inclusão é uma das operações mais básicas utilizadas nos esquemas em Z. Quando se quer incluir um esquema em outro, simplesmente adiciona-se o nome do esquema que se quer incluir (Figura 3), no esquema em que se está trabalhando (Figura 4). O exemplo abaixo ilustra uma inclusão:

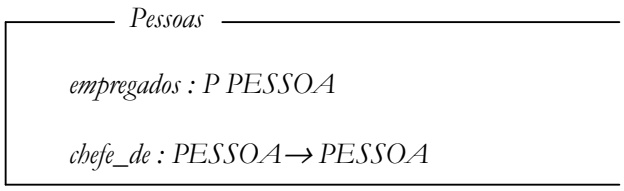


Figura 3 - Esquema a ser incluído

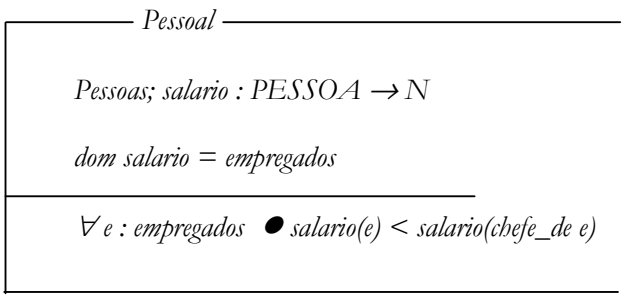


Figura 4 - Esquema atual

Então, as declarações de *Pessoas* são incluídas na declaração de *Pessoal*, e os predicados de *Pessoas* são incluídos (por conjunção) na parte restrição de *Pessoal*.

5.2. Disjunção, conjunção e negação

Essas operações ditam outro comportamento quando as condições são satisfeitas.

$$\mathbf{C \cong A \wedge B;\ D \cong A \vee C;\ N \cong \neg A}$$

Assim, apresenta-se de uma forma simplificada as três operações, onde, A^n e B^n são esquemas, A^n e B^n são esquemas normalizados equivalentes. As declarações de C (D, N) são formadas das declarações de A^n e B^n (ou A^n). O predicado de C (D, N) e a conjunção dos predicados de A^n e B^n, são disjunção de A^n e B^n e negação de A^n.

As operações de disjunção, conjunção e negação poderiam ser combinadas, criando especificações mais complexas. Poderiam ser definidas as noções de implicação e de equivalência entre esquemas. Assumindo que E e F são esquemas em Z, e usando o símbolo $\Rightarrow$ para designar implicação entre esquemas, como uma forma alternativa para um esquema do tipo $(\neg E) \vee F$, obtêm-se:

$$\mathbf{M \ E \Rightarrow F}$$

Ainda, usando o símbolo $\Leftrightarrow$ para denotar a equivalência entre esquemas, pode-se interpretar, como uma forma mais concisa para o esquema $(E \Rightarrow F) \wedge (F \Rightarrow E)$:

$$\mathbf{M \ E \Leftrightarrow F}$$

5.3. Composição

Os esquemas descrevem operações que alteram o estado do sistema. Essa transformação se dá quando mudam os valores de certas variáveis de estado do sistema, por meio da ação do esquema. Portanto, é natural imaginar uma seqüência de transformações de estado promovidas pelas ações de vários esquemas, em sucessão. Então, surge a idéia de composição de esquemas, onde a ação consecutiva de dois esquemas poderia se capturada por um terceiro novo esquema. Ou seja, pode-se dizer que a ação desse terceiro esquema pode ser decomposta na ação dos esquemas componentes, quando aplicados na seqüência original.

Assim, se E e F são esquemas em Z, então a composição de E e F cria um novo esquema C, representado por:

$$C \triangleq E \circ_9 F$$

Salienta-se que para construir o esquema C, é necessário que o efeito deste seja idêntico ao efeito produzido pelo sequenciamento das ações dos esquemas E e F sobre o estado inicial do sistema. Ou seja, as variáveis que representam novo estado em B devem ser as mesmas que variáveis simples de C e, os tipos de variáveis devem corresponder.

5.4. Operador Δ

Δ é usado para especificar operações que podem modificar o estado. Como exemplo, mostra-se o esquema da Figura 5, onde, se S é um esquema descritor de estado em Z, então ΔS designa um novo esquema definido. Ou seja, ΔS inclui tanto o esquema M quanto sua versão decorada pelo símbolo *'*.

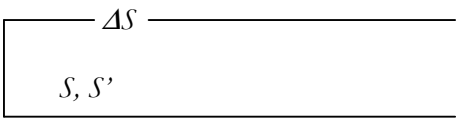


Figura 5 - Esquema do novo estado

A Figura 6 mostra o esquema *AdicionaEmpregado* com a inclusão do novo estado do esquema *Pessoal*, representado por *ΔPessoal*.

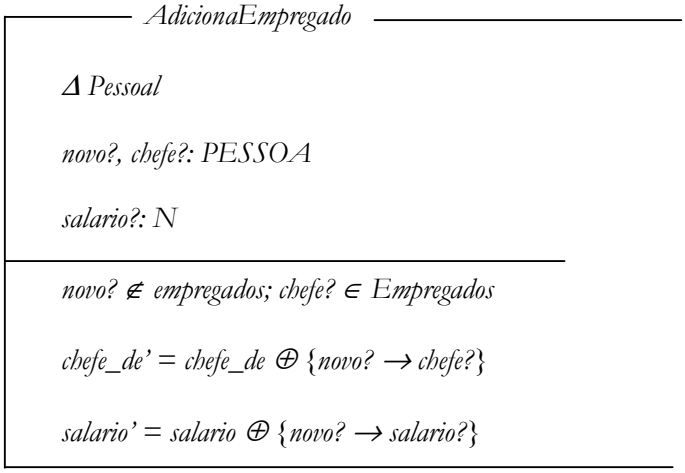


Figura 6 - Esquema AdicionaEmpregado

5.5. Operador Ξ

Ξ é usado para especificar operações que não modificam o estado, por exemplo, realização de consultas. Então se S é um esquema descritor de estado, então a notação ΞS denota um novo esquema, conforme Figura 7, onde v_1 ,..., v_n são todas as variáveis de estado que ocorrem em S. Ou seja, Ξ S incorpora todas as declarações de variáveis de estado de S , pela inclusão da variante ΔS e, além disso, decreta que nenhuma dessas variáveis sofre qualquer alteração em seus valores.

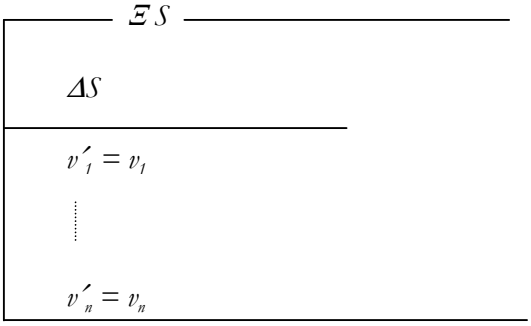


Figura 7 - Novo esquema

7. Glossário da notação

Na Tabela 1 apresenta-se uma lista de conjuntos básicos utilizados nas especificações em Z.

Tabela 1 - Lista de conjuntos básicos

Conjuntos	Significado
∅	Conjunto vazio, um conjunto sem elementos
x ∉ T	Não pertence, x não é um elemento de S
S ⊆ T	Subconjunto, os elementos de S são um subconjunto de T
S ⊂ T	Está contido, S é um subconjunto de T e S não eqüivale a T
S ∪ T	União, o conjunto de elementos estão em S ou T
S ∩ T	Interseção, o conjunto de elementos em comum de S e T
S \ T	Conjunto diferença, o conjunto de S que não está em T

Na Tabela 2 apresenta-se uma lista de relações binárias mais utilizadas nas especificações em Z.

Tabela 2 - Lista de relações binárias

Relações binárias	**Significado**
(x,y)	Par x,y
x ↦ y	X mapeia para y, mesmo que (x,y)
first p	Primeiro elemento do par p
second p	Segundo elemento do par p
X ↔ Y	Relação binária
dom R	Domínio, o conjunto dos primeiros elementos de todos os pares em R
ran R	Imagem, o conjunto do elementos secundários de todos os pares em R
$Q\ ^{0}_{9}\ R$	Composição relacional, Q é composto por R

Na Tabela 3 apresenta-se uma lista de números e operações aritméticas mais utilizadas nas especificações em Z.

Tabela 3 - Lista de números e operações aritméticas

Números/Op. Aritméticas	**Significado**
ℤ	Conjunto dos números inteiros
ℕ	Conjunto dos números naturais, iniciando em zero
$\mathbb{N}_1$	Conjunto dos números positivos, iniciando com um
+, - , *	Operações aritméticas, adição, subtração e multiplicação
i .. j	Conjunto dos número que inicia com i e termina com j
P	Indicação de conjunto não vazio
F	Conjunto finito

Na Tabela 4 apresenta-se uma lista das funções mais utilizadas nas especificações em Z.

Tabela 4 - Lista de funções

Funções	**Significado**
$X \pfun Y$	Função parcial, alguns membros de X são emparelhados com um membro de Y
$X \fun Y$	Função total, todos membros de X são emparelhados com um membro de Y
$X \pinj Y$	Injeção parcial, alguns membros de X são emparelhados com diferentes membros de Y
$X \inj Y$	Injeção total, todos membros de X são emparelhados com diferentes membros de Y
$X \bij Y$	Bijeção, todos membros de X são emparelhados com diferentes membros de Y