ELIANE POZZEBON

Um modelo para Suporte ao Aprendizado em Grupo em Sistemas Tutores Inteligentes

FLORIANÓPOLIS

2008

UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

Um modelo para Suporte ao Aprendizado em Grupo em Sistemas Tutores Inteligentes

Tese de Doutorado submetida à Universidade Federal de Santa Catarina como parte dos requisitos para obtenção do Titulo de Doutor em Engenharia Elétrica.

Eliane Pozzebon

Florianópolis, setembro de 2008

Um modelo para Suporte ao Aprendizado em Grupo em Sistemas Tutores Inteligentes

Eliane Pozzebon

Esta Tese foi julgada adequada para a obtenção do título de Doutor em Engenharia Elétrica, Área de Concentração em *Automação e Sistemas*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.

_____

Prof. Guilherme Bittencourt, Dr.
Orientador

_____

Prof.ª Janette Cardoso, Dra.
Co-Orientadora

_____

Prof.ª Kátia Campos de Almeida, Dra.
Coordenadora do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

_____

Prof. Guilherme Bittencourt, Dr.
Presidente

_____

Prof. Evandro De Barros Costa, Dr.

_____

Prof. Frederico Luiz Gonçalves de Freitas, Dr.

_____

Prof.ª Clara Amélia de Oliveira, Dra.

_____

Prof. Roberto Willrich, Dr.

Aos meus pais, Otto e Lourdes.

AGRADECIMENTOS

Ao meu orientador, Professor Guilherme, minha gratidão pela acolhida como sua orientanda, pela orientação segura, por sua inteligência, por seu incentivo e pelas contribuições valiosas na orientação deste trabalho.

À co-orientadora Janette Cardoso pela oportunidade da vivência na França, com quem muito tive oportunidade de aprender.

Aos membros da banca, em especial ao relator Prof. Evandro de Barros Costa, pelas avaliações e pelas importantes sugestões para o melhoramento deste trabalho.

Aos participantes do projeto MathNet que colaboraram com este trabalho, em especial ao Emílio e João.

Às pessoas com quem tive oportunidade de conviver durante o período de doutorado, pelo companheirismo, convivência intelectual e amizade. Em particular, à Underléa, Marisa, Fernando, Adriana, Fábio, Eduardo, Jones, Edileusa, Jerusa, Flávio, Luciana, Marília e Rui.

À toda equipe da Universidade Toulouse 1, na França, que me recebeu muito bem.

Aos meus amigos e colegas da França, em particular, Pascaline, Matthias, Omar.

Ao meu marido, meus pais, meus amigos e familiares pelo incentivo, apoio, afeto e amor.

Aos professores do departamento de Automação e Sistemas da UFSC, ao Wilson e ao Marcelo da Secretaria da PGEEL.

Ao CNPQ pela bolsa de doutorado no Brasil e à CAPES pela oportunidade de crescimento profissional e também pessoal proporcionando a experiência de viver em outro país.

À todos, que de alguma forma contribuíram para a conclusão deste trabalho, quero expressar os meus sinceros agradecimentos!

Resumo da Tese de Doutorado submetido à UFSC como parte dos requisitos necessários para obtenção do grau de Doutor em Engenharia Elétrica.

Um modelo para Suporte ao Aprendizado em Grupo em Sistemas Tutores Inteligentes

Eliane Pozzebon

Setembro / 2008

Orientador: Guilherme Bittencourt, Dr.
Co-orientadora: Janette Cardoso, Dra.
Área de Concentração: Automação e Sistemas
Palavras-chave: Sistemas Tutores Inteligentes, Sistemas Multiagentes, Aprendizado em Grupo.
Número de Páginas: 144

Este trabalho propõe um modelo, baseado em ontologias e em redes de Petri para gerenciar grupos de estudantes no contexto dos Sistemas Tutores Inteligentes. O modelo proposto explora o modelo de estudante e o módulo de domínio para suportar a adaptação de interação. O modelo inclui uma biblioteca pré-definida de atividades de grupos. Para estabelecer uma atividade do grupo, o professor escolhe um cenário da biblioteca, fornece os parâmetros e o conteúdo da atividade. O tratamento das informações do modelo do estudante é automaticamente incluído nas transições de uma Rede de Petri sob a forma de precondições e condições. O modelo proposto utiliza para a execução uma arquitetura multiagente, que torna operacional o aprendizado em grupo, proporcionando a colaboração entre os estudantes de um mesmo grupo e também entre estudantes de grupos distintos. Uma sociedade heterogênea composta por agentes-aprendizes e agentes gerenciadores de grupos (coordenador de grupos e supervisores de grupos) é definida, onde os agentes-aprendizes são responsáveis por assistir o estudante e representá-los no sistema e o agente coordenador e os agentes supervisores de grupos são responsáveis por gerenciar os grupos de estudantes e acompanhar a interação entre os estudantes. O modelo criado nesta tese foi aplicado em três estudos de casos, sendo um deles implementado e integrado num STI desenvolvido na ferramenta de autoria FAST para gerar sistemas tutores inteligentes.

Abstract of Thesis presented to UFSC as a partial fulfillment of the requirements for the degree of Doctor in Electrical Engineering.

A Model to Support Group Learning in Intelligent Tutoring Systems

Eliane Pozzebon

September / 2008

Advisor: Guilherme Bittencourt, Dr.
Co-Advisor: Janette Cardoso, Dra.
Area of Concentration: Automation and Systems
Keywords: Intelligent Tutoring Systems, Multi-agents Systems, Learning in Group.
Number of Pages: 144

This work proposes a model based on ontologies and Petri nets for the management of student groups in Intelligent Tutoring Systems. The model uses the student model and the domain module to support the interaction adaptation. The model includes a predefined library of the group's activities. To establish a group activity, the teacher chooses a scenario from the library, supplies the parameters and the content of the activity. The information treatment of the student model is automatically included in the Petri Network's transition as preconditions and conditions. The considered model uses for its execution a multiagent architecture, which makes the group learning operational, providing the collaboration among students from a same group and also among students from different groups. A heterogeneous society composed by learner-agents and group-managing agents (group coordinator and group supervisors) is defined, in which the learner-agents are responsible for assisting the students and representing them in the system and the coordinator agent and the group supervising agents are responsible for managing the students groups and for following the interaction among the students. The model created has been applied in three case studies, one of them implemented and integrated into STI authoring tool developed in the FAST system to generate intelligent tutoring system.

Sumário

Lista de Figuras

Lista de Abreviaturas

AA	*Agente Aprendiz*
ASTI	*Arcabouço para Sistemas Tutores Inteligentes*
AT	*Agente Tutor*
CAI	*Computer Aided Instruction*
CG	*Coordenador de Grupos*
CSCL	*Computer Supported Collaborative Learning*
EAD	*Ensino à Distância*
FAST	*Ferramenta de Autoria para Sistemas Tutores*
FIPA	*Foundation for Intelligent Physical Agents*
IAD	*Inteligência Artificial Distribuída*
IAEd	*Inteligência Artificial na Educação*
ICAI	*Intelligent Computer Aided Instruction*
ILE	*Intelligent Learning Environment*
ITS	*Intelligent Tutoring System*
JADE	*Java Agent DEvelopment Framework*
JESS	*Java Expert System Shell*
RdP	*Rede de Petri*
RPO	*Rede de Petri Objetos*
SATA	*Sociedade de Agentes Tutores Artificiais*
S2AG	*Sociedade de Agentes Aprendizes e Gerenciadores de Grupos*
SEH	*Sociedade de Especialistas Humanos*
SG	*Supervisor de Grupo*
SMA	*Sistemas Multiagentes*
STI	*Sistemas Tutores Inteligentes*
UP	*Unidade Pedagógica*

Capítulo 1

Introdução

O campo de Inteligência Artificial na Educação (IAEd) está passando por um processo de evolução, principalmente a partir do surgimento das tecnologias computacionais, como Internet, realidade virtual e multimídia. Entretanto, vários desafios não foram superados na IAEd, especialmente nos Sistemas Tutores Inteligentes (STI) (Pozzebon *et al.*, 2007).

Um Sistema Tutor Inteligente é um ambiente de aprendizagem que permite que um estudante aprenda um determinado assunto com o auxílio do computador. Um sistema para auxiliar na aprendizagem requer uma grande compreensão das várias dimensões envolvidas no processo de ensino. Para o desenvolvimento de Sistemas Tutores Inteligentes (STIs) é necessário integrar diferentes técnicas de Inteligência Artificial (IA).

Um STI, de forma geral, é constituído de quatro módulos: Conhecimento do Domínio, Estratégias Didáticas, Modelo do Estudante e Interface (Wenger, 1987) (Self, 2000). O STI modela o conhecimento do estudante sobre um tópico e à medida em que este realiza determinadas tarefas, o sistema compara este conhecimento com o modelo do conhecimento do domínio. O sistema pode também adaptar os níveis e estilos de aprendizagem ao estudante e apresentar a informação, os testes e as respostas que são mais apropriadas (Frigo, 2007).

A pesquisa e o desenvolvimento de ambientes de aprendizagem com computador no Brasil possuem mais de 30 anos. Os primeiros foram os sistemas CAI (Instrução Auxiliada por Computador), baseados na instrução programada, depois houve o surgimento dos Micromundos, valorizando e abordando uma proposta de aprendizagem através da ação (Self, 1992). Com a inserção da Inteligência Artificial, deu-se origem aos ICAI (Instrução Inteligente Assistida por computador) ou STI (Sistemas Tutores Inteligentes). Os STIs foram criados para prover aos estudantes a aprendizagem personalizada com mecanismos automáticos e inteligentes para

resolução de problemas.

Na década de 90 começou a inserção de modelos computacionais de apoio a aprendizagem com a noção de colaboração. Nesse sentido, os STIs também começaram a ser intitulados de Ambientes Inteligentes de Aprendizagem (*Intelligent Learning Environment ILE*) ou Sistemas Tutores Cooperativos (Costa, 1997). A utilização de agentes e o domínio Web fortaleceu os ambientes de ensino a distância. O termo agente neste trabalho significa o agente artificial.

A tecnologia de agentes inteligentes é um campo abrangente dentro da IA, podendo ser aplicada nos mais diversos tipos de problemas. A construção de sistemas baseados em agentes inteligentes ainda é uma tarefa difícil devido principalmente à falta de recursos, oferencendo facilidades ao seus desenvolvedores, como inclusão de técnicas de IA.

Depois de 2000, o enfoque das pesquisas buscou ambientes computacionais de suporte à aprendizagem colaborativa dotada de recursos ainda mais sofisticados e, conseqüentemente, trazendo mais complexidade no seu desenvolvimento.

Começou-se neste período a investir mais nas ferramentas de autoria para automatizar alguns aspectos envolvidos na construção dos ambientes, proporcionando ao usuário que não possui experiência em programação (Professor/Autor) a possibilidade de construir ambiente de ensino-aprendizagem de forma ágil.

As ferramentas de autoria buscam auxiliar o Professor/Autor no processo de construção do sistema educacional informatizado e surgiram devido à complexidade e esforços exigidos dos desenvolvedores e custos envolvidos no desenvolvimento e manutenção destes ambientes (Murray, 1999). Por exemplo, de acordo Ainsworth (2007) as estimativas do tempo de desenvolvimento de um STI variam de 200-1000 horas para um hora de aprendizagem.

Uma maneira de diminuir este custo é construir um STI com o auxílio de uma ferramenta de autoria. Por exemplo, tais ferramentas permitem que autores sem habilidades técnicas mas com conhecimento pedagógico possam criar ambientes de aprendizagem, além de reduzir o tempo para construção destes ambientes de aprendizagem.

Várias ferramentas de autoria têm sido construídas, por exemplo, as descritas em Murray (1999), Mitrovic *et al.* (2006), Razzaq *et al.* (2005), Ainsworth e Fleming (2005), e Aleven *et al.* (2006).

Dentre os STIs existentes, alguns suportam não apenas a apresentação de domínio para um único estudante, mas também suportam diferentes graus de controle que permitem a interação entre os estudantes e/ou grupos de estudantes.

Dentre os sistemas que suportam a noção de grupos de estudantes existem dois tipos. No primeiro tipo estão os sistemas que apenas disponibilizam as ferramentas de comunicação que permitem a interação do grupo (chat, e-mail, fórum, editores cooperativos, etc.), deixando todas as atividades de solução de problemas e de coordenação sob responsabilidade humana. No segundo tipo, existem os sistemas que controlam os detalhes da interação do grupo seguindo protocolos de interação, comunicação e outros.

No primeiro caso, a tarefa de planejamento instrucional do Professor/Autor é pelo menos tão difícil quanto à tarefa de planejamento de trabalho de grupo tradicional. No último caso, a falta de flexibilidade torna difícil alcançar a adaptação dinâmica aos estudantes e compartilhar e reutilizar componentes do STI entre domínios (Pozzebon *et al.*, 2006).

1.1 Motivação

O foco deste trabalho é a concepção e desenvolvimento de ambientes interativos de aprendizagem voltados para grupos de estudantes. Trata-se de uma área complexa caracterizada pela interdisciplinaridade. A concepção de tais ambientes envolve principalmente aspectos das áreas de Educação, Computação (particularmente inteligência artificial e engenharia de software) e Pedagogia.

Esta área de estudos é profícua para o desenvolvimento de sistemas que visam o aperfeiçoamento da aprendizagem. Tais sistemas vão ao encontro das necessidades de alguns setores da sociedade, como organizações industriais, comerciais e ambientes acadêmicos.

Uma das tendências observadas é o desenvolvimento de ambientes computacionais com ferramentas de autoria que suportem a aprendizagem colaborativa dotada de recursos cada vez mais sofisticados, que permitem inclusive o gerenciamento de aprendizado em grupos de estudantes independente da situação geográfica.

Para proporcionar um melhor aprendizado ao estudante são necessários recursos que lhe ofereçam um aprendizado através da comunicação com seus colegas de estudo, compartilhando conhecimentos e dúvidas. Este enfoque de aprendizado em grupos tornou-se uma tendência na área de Inteligência Artificial na Educação (IAEd), que está buscando novas ferramentas e metodologias que possam ser apropriadas para o ensino e aprendizagem.

O tema deste trabalho é desafiador e complexo devido à diversidade dos conhecimentos necessários para resolver problemas como, por exemplo, o controle dinâmico de um grupo, o

que envolve a Computação Distribuída, o aspecto de Engenharia de Software, e a Inteligência Artificial (*e.g.* agentes inteligentes).

1.2 Contexto e Definição do Problema

Um sistema para auxiliar na aprendizagem requer a compreensão dos vários conceitos envolvidos no processo de ensino-aprendizagem, tais como, professor, estudante, conteúdo pedagógico e estratégias de ensino. Para o desenvolvimento de STIs é possível utilizar diferentes técnicas da Inteligência Artificial (IA), como, Sistemas Especialistas, Redes Bayesianas e Agentes Inteligentes (Wenger, 1987), (Self, 2000) e (McCalla, 2000).

Técnicas de aprendizagem em grupo oferecem vantagens para estudantes e professores, pois contribuem para: o desenvolvimento do pensamento crítico e de habilidades para a resolução de problemas; o aumento dos níveis de motivação quando os estudantes se familiarizam com o trabalho em grupos; melhorar o desempenho dos estudantes levando à redução da ansiedade e ao aumento da auto-estima; a busca dos diferentes estilos de aprendizagem dos estudantes, como o verbal e o visual, entre outras.

O gerenciamento dos grupos de estudantes é uma tarefa dinâmica, que depende de diversos eventos assíncronos (disponibilidade dos estudantes, dados em seus modelos, sua situação pedagógica atual, suas preferências declaradas). Uma vez formado o grupo, sua atividade deve ser gerenciada, dando origem a um protocolo de interação entre os membros e a um modelo do grupo, com as informações sobre suas atividades.

A utilização de sistemas multiagentes possibilita um nível de abstração mais adequado para o tratamento de problemas complexos e distribuídos, tais como ambientes de trabalho e aprendizagem em grupo. Com uma arquitetura de agentes desenvolvida para dar suporte a um processo de formação de grupo, será possível estabelecer a negociação entre os agentes definindo desta forma, encontros ou reuniões entre estudantes do grupo.

O STI mantêm informações sobre o estudante numa base de dados chamada de modelo do estudante. O modelo de um estudante contém informações sobre o mesmo, tais como: preferências, nível de conhecimento, exercícios resolvidos, atividades a serem cumpridas, etc. Com essas informações atualizadas, é possível verificar quais são os estudantes que possuem o mesmo nível de conhecimento e agrupá-los.

Após a criação dos grupos é necessário definir meios para integrar os membros do grupo,

com o objetivo de que se ajudem mutuamente quando estiverem com dificuldades em alguma atividade. O problema de suporte a grupos nos sistemas tutores inteligentes é bastante complexo e trata das interações entre estudantes e o STI, além da apresentação de um domínio do conhecimento de acordo com as estratégias pedagógicas determinadas pelo professor.

1.3 Objetivo Geral

O objetivo deste trabalho é contribuir para a concepção de ambientes interativos de aprendizagem auxiliando os desenvolvedores com a definição de um modelo para gerenciamento de grupos utilizando uma ferramenta de autoria. Desta forma, os professores que não são familiarizados com a tarefa de programação poderão configurar o domínio de ensino, e os estudantes poderão escolher a forma de aprendizado: individualizada ou em grupos com cenários diversificados.

1.4 Objetivos Específicos

Os seguintes objetivos específicos são necessários para o cumprimento do objetivo geral definido na seção anterior:

1. Especificação de um modelo para criação de cursos que permita aprendizado individual e/ou em grupos em STI.

2. Especificação de um modelo que armazene tanto as características dos participantes (estudantes) como os resultados durante as atividades entre os participantes e a ferramenta de aprendizagem.

3. Criar uma arquitetura multiagente para gerenciar as interações inter e intra-grupo de estudantes.

4. Criação de cenários pré-definidos em módulos, objetivando a reutilização das unidades de gestão e conteúdo.

5. Aplicar o modelo proposto numa ferramenta de autoria para STI, de modo que este suporte a atividade de grupos.

1.5 Metodologia

O desenvolvimento do modelo proposto para o gerenciamento de grupos de estudantes em STI seguiu as seguintes etapas:

1. Levantamento e análise dos trabalhos correlatos.
2. Elaboração de um modelo preliminar.
3. Escolha das ferramentas computacionais para a implementação do modelo proposto no item 2.
4. Criação de estudos de caso com base no modelo criado no item 2.
5. Implementação de um protótipo para teste do modelo proposto.

Para a especificação do modelo de gerenciamento de grupos foram utilizados redes de Petri e ontologia; para a implementação foram escolhidas as ferramentas *Java Agent DEvelopment framework* (JADE) como plataforma de agentes, *Java Expert System Shell* (JESS) para a definição de regras utilizadas pelos agentes para gerenciar os grupos, *Tomcat* como servidor de servlets.

A ontologia neste trabalho foi utilizada como uma ferramenta para representação dos modelos dos STI.

1.6 Organização do Documento

Esta tese está dividida em cinco capítulos. Este capítulo apresentou uma introdução sobre o trabalho, a justificativa que levou à escolha do tema e a relevância do mesmo.

No **Capítulo 2** são descritos os principais conceitos envolvidos no desenvolvimento do presente trabalho. Primeiramente são descritos os conceitos de aprendizagem colaborativa assistida por computador e os sistemas tutores inteligentes, bem como uma breve descrição dos módulos que compõem os STI. O capítulo também discorre sobre o modelo MATHEMA e a ferramenta de autoria FAST, que são a base conceitual do modelo desenvolvido nesta tese. Por fim, são descritas as características dos Sistemas Multiagentes, tais como, mecanismos de comunicação e interação entre agentes.

No **Capítulo 3** são apresentados os trabalhos de aprendizado em grupos suportados por computador em Sistemas Tutores Inteligentes. Neste levantamento bibliográfico foi focado o modelo com os agentes, ontologia, protocolo/RPO e cenários.

O **Capítulo 4** descreve o modelo proposto para o gerenciamento de grupos em Sistemas Tutores Inteligentes que está relacionado ao aprendizado colaborativo. Este modelo possui dois níveis: o nível de especificação, que adota as ontologias para representar os modelos e redes de Petri para especificar os protocolos de interações, e o nível de execução, que torna operacional o aprendizado em grupo com a arquitetura multiagente, que proporciona a colaboração entre os estudantes do mesmo grupo e estudantes de grupos diferentes.

O **Capítulo 5** descreve os estudos de casos utilizados para validar o modelo proposto e a implementação, em que foi criado um STI para suportar o aprendizado Colaborativo com o modelo proposto, integrado na ferramenta de autoria FAST que é baseada no modelo MATHEMA.

O **Capítulo 6** aborda as principais conclusões desta tese, juntamente com as perspectivas para trabalhos futuros.

O **Apêndice A** descreve as ferramentas que foram utilizadas para a modelagem e o desenvolvimento do sistema de gerenciamento de grupos proposto nesta tese.

O **Apêndice B** apresenta as regras do estudo de caso implementado.

O **Apêndice C** apresenta as interações do estudante com o STI.

O **Apêndice D** apresenta o compilador para transformar grafo de pré-requisitos em regras.

Capítulo 2

Sistemas Tutores Inteligentes e Técnicas Correlatas

Neste capítulo são descritos os principais conceitos envolvidos no desenvolvimento do trabalho. Primeiramente são descritos os conceitos de aprendizagem colaborativa assistida por computador e os sistemas tutores inteligentes, bem como uma breve descrição dos módulos que compõem os STI. O capítulo também discorre sobre o modelo MATHEMA e a ferramenta de autoria FAST, que são a base conceitual do modelo desenvolvido nesta tese. Por fim, são descritas as características dos Sistemas Multiagentes, tais como, mecanismos de comunicação e interação entre agentes.

2.1 Aprendizagem Colaborativa Assistida por Computador

O conceito do aprendizado colaborativo nasceu no início do século passado, Lev S. Vygotsky, professor e pesquisador russo, lançou o conceito de que o desenvolvimento cognitivo é impactado por interações e relações sociais. Ou seja, aprender é uma atividade social mediada pelos colegas e professores.

Segundo Dillenbourg (1999), a colaboração é uma situação em que duas ou mais pessoas aprendem ou tentam aprender algo em conjunto e está relacionada a quatro diferentes aspectos de aprendizagem: situação, interações, mecanismos e os efeitos da aprendizagem colaborativa. Uma situação pode ser mais ou menos colaborativa, por exemplo, é mais fácil ocorrer colaboração entre colegas do que entre um subordinado e seu chefe. Já as interações também possuem níveis diferentes de colaboração, por exemplo, negociação parece ser mais colabora-

tiva do que dar ordens. Alguns mecanismos de aprendizagem são intrinsecamente mais colaborativos. O último dos aspectos (efeitos da aprendizagem colaborativa) refere-se as diferentes formas de se avaliar a aprendizagem colaborativa. Assim, para entender a aprendizagem colaborativa é necessário entender a relação entre os quatro ítens. Em um primeiro momento a situação gera padrões de interação, essas interações ativam mecanismos cognitivos que por sua vez geram efeitos cognitivos. Contudo essa linearidade é uma simplificação, sendo que a maioria das relações é recíproca.

As palavras *Aprendizado Colaborativo* descrevem uma situação em que formas particulares de interação entre duas pessoas são esperadas e que desencadeariam mecanismos de aprendizagem, mas não há garantias que elas ocorram. Portanto é necessário aumentar a probabilidade de alguns tipos de interação ocorrerem, o que pode ser alcançado apoiando interações mais produtivas pela inclusão de regras de interação no ambiente de aprendizagem (Dillenbourg, 1999).

Harasim *et al.* (2005) também afirma que a participação do estudante é maior em aulas colaborativas e que o sucesso do aprendizado depende muito de regras claras. Em qualquer tipo de sala de aula, o professor deve estabelecer bases de comportamento, esclarecendo as expectativas e o papel de cada estudante. Caso contrário, estudantes desmotivados podem atrapalhar o bom desempenho do aprendizado do grupo.

A aprendizagem em grupo suportada por computador é uma área de pesquisa que visa proporcionar um ambiente de aprendizagem colaborativa utilizando-se de software e hardware que suportam e ampliam o trabalho ou aprendizagem em grupo tradicional (Arriada e Ramos, 2000).

A aprendizagem colaborativa assistida por computador (CSCL - *Computer Supported Collaborative Learning*) pode ser definida como uma estratégia educativa em que dois ou mais sujeitos constroem o seu conhecimento através da discussão, da reflexão e da tomada de decisões, e onde os recursos tecnológicos atuam como mediadores do processo de ensino-aprendizagem.

A CSCL cresceu em torno de um vasto leque de investigações sobre trabalho colaborativo assistido por computador (CSCW *Computer Suported Collaborative Work*). A principal diferença entre CSCW e CSCL é que a CSCW está sendo utilizada principalmente no domínio empresarial enquanto a CSCL está sendo explorada em ambientes educativos. A CSCW tende a focalizar a sua atenção nas técnicas de comunicação enquanto a CSCL tende a concentrar a sua atenção no conteúdo a ser comunicado.

Vantagens da aprendizagem colaborativa

Para os pesquisadores da Universidade de Évora (2008) as vantagens da aprendizagem colaborativa são as seguintes:

a) Na dinâmica do grupo:

- possibilitar alcançar objetivos qualitativamente mais ricos, resultantes de propostas e soluções de vários estudantes do grupo;
- permitir a interdependência positiva entre estudantes: aprender partilhando permite que os estudantes se integrem na discussão e tomem consciência da sua responsabilidade no processo de aprendizagem;
- incentivar os estudantes a aprender entre eles, valorizando os conhecimentos dos outros, tirando partido das experiências de aprendizagem de cada um;
- permitir maior aproximação entre os estudantes e uma maior troca ativa de idéias no seio dos grupos, faz aumentar o interesse e o compromisso entre eles;
- transformar a aprendizagem numa atividade social;
- aumentar a satisfação pelo próprio trabalho.

b) No nível pessoal:

- aumentar as competências sociais, de interação e comunicação efetivas;
- incentivar o desenvolvimento do pensamento crítico e a abertura mental;
- permitir conhecer diferentes temas e adquirir nova informação;
- reforçar a idéia que cada estudante é um professor (a aprendizagem emerge do diálogo ativo entre professores estudantes);
- diminuir o sentimento de isolamento e de temor à crítica;
- aumentar a segurança em si mesmo, a auto estima e a integração no grupo;
- fortalecer o sentimento de solidariedade e respeito mútuo, baseado nos resultados do trabalho em grupo.

Elementos básicos da aprendizagem colaborativa

Alcântara *et al.* (2004) apresenta quatro elementos básicos da aprendizagem colaborativa, quais sejam:

1. a interdependência positiva, é fundamental que haja uma responsabilidade de todos sobre a produção final que está sendo construída. Ou seja, mesmo que haja uma divisão de tarefas é necessário que todos os componentes do grupo se sintam responsáveis por todo o trabalho.

2. a interação face-a-face, o trabalho colaborativo sempre deverá considerar a questão das interações entre os sujeitos.

3. a contribuição individual, é necessário que o sujeito tenha a compreensão da sua participação no trabalho. Assim, torna-se um elemento motivador a referência individual em alguns momentos, o que possibilitará um trabalho colaborativo mais rico.

4. o desenvolvimento de habilidades interpessoais e de atividade de grupo. Além de apropriar-se dos conceitos pertinentes ao trabalho proposto, outra questão que está presente em trabalhos colaborativos é justamente o desenvolvimento de habilidade de relacionamento em grupo. Ou seja, poderá fazer parte dos objetivos do trabalho a forma como o sujeito se relaciona com os demais componentes do grupo.

Teorias relacionadas com CSCL

Muitas teorias contribuem para a compreensão da aprendizagem colaborativa assistida por computador. Estas teorias fundamentam-se na hipótese de que os indivíduos são agentes ativos que intencionalmente procuram e constroem o conhecimento num contexto significativo.

Alguns exemplos destas teorias são: teoria sociocultural (baseada na intersubjectividade e na zona de desenvolvimento proximal de Vigotsky); construtivismo e aprendizagem auto regulada (Piaget); teoria da flexibilidade cognitiva; conhecimento situado, aprendizagem cognitiva, aprendizagem baseada na resolução de problemas (Gallagher *et al.*, 1992); e outras.

Na literatura podemos encontrar trabalhos, como, Inaba *et al.* (2000) e Isotani e Mizoguchi (2008a) que buscam definir padrões de interação baseados em diversos tipos de processos de interação encontrados nas teorias de aprendizagem (por exemplo, Aprendizagem Cognitiva, Tutoria, Aprendizado Situado, além de outras).

Outro exemplo, é o ambiente COLER (Constantino-Gonzalez *et al.*, 2003), baseado na teoria Neo-Piagetiana do conflito sócio-cognitivo, que defende a importância das discussões e conflitos entre as idéias dos estudantes no processo de aprendizagem. O COLER é utilizado na aprendizagem colaborativa de modelagem Entidade-Relacionamento. Inicialmente, os estudantes constroem um diagrama individualmente e um componente do COLER identifica diferenças entre esses diagramas e sua influência no aprendizado, encoraja os estudantes a discutir essas diferenças em pequenos grupos e a chegar a uma solução de consenso.

Na presente tese foi escolhida a teoria de aprendizagem baseada na resolução de problemas (PBL *Problem Based Learning*) porque ensinar com base na resolução de problemas tem muitas vantagens, a primeira é a possibilidade de contextualização dos problemas. Quem escolhe trabalhar com aprendizagem baseada na resolução de problemas é praticamente obrigado a contextualizar, assim o aprendizado fica claro para a maioria dos estudantes.

Classificações de grupos

Kaliannan (1999) considera duas abordagens para classificação de grupos. A primeira leva em conta a dinâmica dos grupos e classifica as aplicações de acordo com alguns critérios.

- Nível do grupo: considera principalmente o número de entidades que formam o grupo;
- Padrão de acesso dos membros: considera como os grupos são formados e distribuídos e como a composição de grupo evolui no tempo;
- Disseminação das informações: considera os padrões de comunicação e interação entre os membros e os tipos de informações trocadas.

A segunda classifica as aplicações de acordo com três critérios de escala:

- Escala Espacial: considera a dispersão geográfica dos participantes;
- Escala numérica: considera o número de entidades em uma aplicação distribuída, tratando normalmente de um mesmo tipo de entidade (estudantes, grupos, etc);
- Escala organizacional: leva em conta as necessidades geradas pela implantação das tecnologias nas organizações, como segurança de dados.

Ellis *et al.* (1991) propõem que um software que apoia o trabalho em grupo pode ser projetado tanto para ajudar um grupo face a face, quanto distribuído, ou para aprimorar a

comunicação e colaboração dentro de uma interação. Estas considerações de espaço e tempo dão origem a quatro categorias, conforme Tabela 2.1.

Tabela 2.1: Matriz Espaço x Tempo.

	Mesmo Tempo	Tempos Diferentes
Mesmo Local	Int.Síncrona	Interação Assíncrona
Local Diferente	Int.Síncrona Distr.	Int.Assíncrona Distr.

Esta classificação, proposta por Ellis *et al.* (1991), divide os grupos em duas dimensões, uma para o espaço, tratando da localização física dos participantes, e outra temporal, tratando do momento em que os participantes trabalham, a qual se divide em mesmo tempo (síncrona) e em tempos diferentes (assíncrona).

A comunicação síncrona é realizada em tempo real, exigindo participação simultânea de todos os envolvidos. Por exemplo, o chat e a videoconferência. A comunicação assíncrona é realizada em tempos diferentes, não exigindo a participação simultânea dos envolvidos. Os participantes não necessitam estar reunidos no mesmo local ou ao mesmo tempo, resultando numa maior flexibilidade de interação e acompanhamento. Por exemplo, o correio eletrônico e fóruns de discussão.

A Tabela 2.2 lista alguns exemplos de ferramentas de apoio para comunicação do grupo, tais como videoconferência e correio eletrônico que são utilizados quando os participantes não estão no mesmo local.

Tabela 2.2: Exemplos de ferramentas de apoio.

	Mesmo Tempo	Tempos Diferentes
Mesmo Local	Tomada de decisão Reuniões Eletrônicas Edição Cooperativa	Gerência projetos Edição Cooperativa
Local Diferente	Videoconferência	Correio Eletrônico

Existem vários métodos para modelagem e análise da comunicação síncrona e assíncrona em grupos de estudantes, tais como os descritos em Avouris *et al.* (2004), Muelenbrock e Hoppe (1999) e Hoppe (1995).

Nesta tese a aprendizagem em grupo de estudantes é aplicada nos Sistemas Tutores Inteligentes, portanto a seguir são descritos os conceitos e apresentados os módulos que compõem tais sistemas.

2.2 Sistemas Tutores Inteligentes

Os Sistemas Tutores Inteligentes (STI) representam uma parte significativa da Inteligência Artificial na Educação (IAEd) e vêm ganhando uma maior importância devido a influência dos recentes desenvolvimentos das tecnologias de comunicação e informação.

Um Sistema Tutor Inteligente (STI) é um sistema computacional para o ensino que tem algum grau de tomada de decisão autônoma em relação às suas interações com os usuários (estudantes) . Esse processo de decisão é necessariamente feito de maneira on-line, durante as interações do sistema com os usuários e, geralmente o sistema precisa acessar vários tipos de conhecimento e processos de raciocínio para habilitar tais decisões a serem tomadas.

Baseado nos princípios da Instrução Assistida por Computador (CAI), estes sistemas de ensino tentam implementar o modelo genérico que possa servir para o ensino a qualquer estudante. Originalmente, a idéia era substituir um tutor humano pelo computador, aproveitando o fato de que o computador é uma ferramenta que permite manipular conhecimento. Com esse propósito, surgiram os sistemas de Instrução Inteligente Assistida por computador (ICAI). Os STIs possuem uma base de conhecimento e são portanto considerados como um tipo de ICAI.

Neste contexto, a IA é utilizada de forma a possibilitar um aumento do potencial de aprendizagem através de técnicas cognitivas. O termo inteligente refere-se então à habilidade que o sistema deve ter de saber o que ensinar, quando e como. Além disso, poderá identificar os pontos fracos e fortes do estudante e usar uma estratégia baseada nessa informação. Poderá procurar informação relevante sobre o aprendizado do estudante (por exemplo respeitando as preferências de aprendizagem), e utilizar os melhores meios de instrução para esse determinado estudante. Ao longo da instrução, o sistema poderá avaliar se o estudante está processando e assimilando de forma correta o conteúdo ensinado.

Mais precisamente, um STI é um sistema computacional que faz o tutoramento de um estudante num dado domínio, como por exemplo matemática. Alguns STI´s modelam o entendimento do estudante sobre um assunto e à medida em que ele realiza determinadas tarefas no sistema, o conhecimento do estudante é comparado com o modelo que ele tem de um especialista naquele domínio. Se existir uma diferença, o sistema pode usar o seu modelo do domínio para gerar uma explicação que vai auxiliar o estudante a compreender o que ficou mal entendido. Além disso, o sistema pode também ajustar os níveis e preferências de aprendizagem do estudante e apresentar a informação, os testes e o *feedback* que são mais apropriados (Pozzebon, 2003).

Um dos objetivos dos STIs é ser capaz de modelar comportamentos de ensino, os quais se adaptam às necessidades do estudante, à situação de aprendizagem e ao assunto da instrução (Murray, 1999).

Wenger (1987) sugere que a função principal de um STI é agir como um veículo de comunicação para ensinar. Trabalhos mais recentes reforçam este ponto dando ênfase sobre o aspecto da comunicação entre tutor e estudantes. Portanto, independente do paradigma de ensino utilizado, o propósito fundamental de todo STI é comunicar o conhecimento e/ou habilidades para o estudante conseguir resolver problemas dentro de um determinado domínio.

2.2.1 Componentes de um STI

Apesar dos STIs diferirem em vários aspectos, a maioria deles segue uma estrutura tradicional (figura 2.1). Classicamente, um STI inclui três componentes: o módulo do domínio, o módulo pedagógico e o modelo do estudante, além da interface com o usuário. A Figura 2.1 exemplifica um STI genérico.

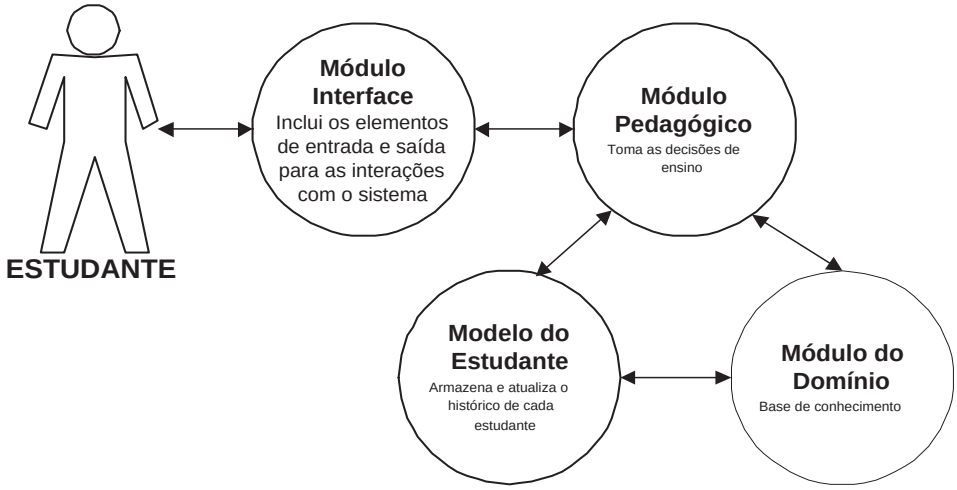

Figura 2.1: Componentes de um Sistema Tutor Inteligente. Adaptado de Mohamed (2005).

Módulo do domínio

Também denominado de Modelo Especialista, o módulo do domínio contém o conhecimento do domínio do sistema e os mecanismos de inferência. É fundamentalmente uma base de conhecimento contendo informações de um determinado domínio, organizada para representar o conhecimento de um especialista ou professor. Segundo Park (1988), o especialista é, geralmente, considerado um dos componentes mais importantes de qualquer STI, pois incorpora a maior

parte da *inteligência* do sistema na forma do conhecimento necessário para solucionar problemas do domínio. O grande desafio para cada novo STI é fornecer uma representação do seu domínio, rica o suficiente para suportar o nível desejado de compreensão proporcionando uma maior flexibilidade no ensino.

O conhecimento de domínio do STI pode ser declarativo e teórico ou procedimental (Rosatelli *et al.*, 2000):

1. Declarativo e teórico: o conhecimento consiste das unidades contendo os conceitos do domínio e suas relações. Para representá-lo são utilizadas qualquer formalismo de representação de conhecimento orientado a domínio, frames, etc. A metodologia utilizada para a aquisição do conhecimento é dividida em três fases: determinar os objetos a serem incluídos no domínio; decidir como os objetos se relacionam entre si; e verificar quais relações estão corretas.

2. Procedimental: é um tipo de conhecimento tipicamente explicativo, nele se explica como fazer uma certa tarefa, como diagnosticar um problema ou recomendar uma ação. Para incorporar o conhecimento em um sistema, se recomenda estabelecer as metas, estabelecer os fatos e estabelecer as relações entre as metas e os fatos.

Assim, o módulo domínio desempenha dupla função:

- age como uma fonte para o conhecimento a ser apresentado. Isto inclui tanto a geração de explicações e respostas aos estudantes, como também tarefas e questões a serem colocadas para resolução;

- serve como um padrão para as avaliações de desempenho do estudante. Para esta função o módulo do especialista deve ser capaz de gerar soluções para problemas no mesmo contexto que o estudante realiza, para que as respectivas respostas possam ser comparadas.

Módulo pedagógico

Também conhecido como Modelo do Tutor ou Instrutor, é o módulo responsável pelas estruturas didáticas e pedagógica e, de certa forma, faz a ligação entre os outros modelos. Cabe a este coordenar e determinar como ensinar, selecionando os tópicos e exemplos a serem dados, planejando o modelo global do tutor e elaborando as estratégias instrucionais.

Um professor dispõe de diversas maneiras de expor um assunto, tornando-o mais compreensível e interessante. A comunicação de um conhecimento para uma pessoa não segue simplesmente um protocolo de transferência de informação como acontece, por exemplo, entre dois computadores. Ela é guiada por estratégias e técnicas que são selecionadas e combinadas dinamicamente reagindo às atitudes e necessidades dos estudantes, por exemplo, as aulas se desenvolvem guiadas pela curiosidade do estudante (Kuyven, 2002).

Existem diversas abordagens pedagógicas empregadas em STIs, mas a maioria dos sistemas tendem a implementar somente uma estratégia pedagógica. Por isso, normalmente estes sistemas não possuem um rico repertório de estratégias de ensino a serem selecionadas. Esta deficiência existe, em parte, porque a maioria das pesquisas concentram-se nos problemas de representação de conhecimento e diagnósticos, ao invés de focar os processos pedagógicos envolvidos no ato de ensinar.

Modelo do estudante

Também conhecido como Modelo do Aluno, do Aprendiz ou do Usuário, é fonte de informação sobre o estudante. O leque de possibilidades e funções deste módulo varia muito de uma implementação para outra. Em geral ele é utilizado para registrar as diferentes atividades de um estudante e assim permitir ao sistema guiá-lo e aconselhá-lo nos momentos certos. Pode ser também um elemento que possui informação incompleta acerca do tópico que vai ser ensinado e que aprende junto com o estudante, servindo assim de parceiro da interação do estudante com o sistema de conhecimento do tutor.

Segundo (Self, 2000), o modelo do estudante é a chave para um ensino personalizado e inteligente num sistema tutorial, e representa o conhecimento que o sistema deve ter de seu próprio usuário. O modelo do estudante deve enfocar mais o processo interativo, ao longo do tempo, levando em conta as ações do estudante, o contexto em que elas ocorreram e estrutura cognitiva do estudante naquele momento.

Este módulo é constituído por dados estáticos e dinâmicos que serão de fundamental importância para o sistema tutor poder comprovar hipóteses a respeito do estudante. Contém uma representação do estado do conhecimento do estudante no momento em que ele interage com um STI. A partir desse modelo e do conteúdo a ser ensinado, o sistema deve ser capaz de inferir a melhor estratégia de ensino a ser utilizada (Desmoulins e Labeke, 1996).

De acordo com (Giraffa, 1999) o modelo de estudante pode ser representado das seguintes

formas:

- Modelo diferencial : consiste da comparação da resposta do estudante com a base de conhecimento. Nesta modelagem, é comparado o desempenho do especialista com o do estudante. O conhecimento é divido em duas classes: aquele que se espera do estudante, e aquele que não se espera.

- Modelo *overlay* : consiste da representação do conhecimento do estudante como um subconjunto da base de conhecimento. Uma crítica ao modelo *overlay* é que este supõe (implícita ou explicitamente) que os erros ou comportamentos anômalos do estudante são sempre devidos à ausência de alguma informação na base do domínio.

- Modelo de perturbação: consiste em supor que os erros do estudante são decorrentes da concepção errônea de algum conceito ou ausência dele. Este modelo também relaciona o modelo do estudante com a base de conhecimento do domínio.

- Modelo estereótipo: um modelo de estudante deste tipo distingue vários tipos de usuários (estudantes). A modelagem do estudante estereotipado pode ter várias dimensões e para cada dimensão o sistema pode ter um conjunto possível de estereótipos. Um exemplo de classificação poderia ser "novato - iniciante - intermediário - especialista".

- Modelo de simulação: consiste em um modelo de como o estudante pode ou deve comporta-se em determinada situação. Por meio deste modelo é possível prever o comportamento futuro do estudante, ou seja, a resposta do estudante, somente com base no seu comportamento durante uma sessão de trabalho.

- Modelo de Agentes: consiste em tratar o modelo do estudante como um sistema de crenças, desejos e intenções (BDI- *Belief, Desires, Intentions*). A interação entre o estudante e o sistema tutor é visto como uma interação entre dois agentes inteligentes, ou pelo menos, dois agentes dotados de algum comportamento cognitivo. Considerar o estudante como um agente, implica considerar o modelo de estudante como um modelo de agente (Wooldrige e Jennings, 1995).

A característica principal do modelo do estudante é a de contemplar todos os aspectos do conhecimento e do comportamento do estudante que tragam conseqüências para o seu desempenho e aprendizagem. Entretanto, a construção de um modelo como este é uma tarefa bastante

complexa para um sistema computadorizado. Os canais de comunicação em um computador podem parecer bastante restritos quando comparados com a capacidade das pessoas em combinar informações em uma grande variedade de meios, como por exemplo o tom de voz ou expressões faciais (Pozzebon, 2003).

Para obter as informações do estudante são utilizadas várias técnicas, por exemplo, questionários, auto-avaliação e histórico da interação.

Quanto à auto-avaliação, Mabbott e Bull (2006) e Kerly *et al.* (2008) utilizam um modelo de estudante aberto, onde os estudantes são autorizados a editar ou negociar suas informações armazenadas no modelo.

Quanto ao histórico da interação para grupo de estudantes é possível explorar as informações do estudante, tais como, informações gravadas em cada interação do estudante, demora para fornecer uma resposta, avaliação da resposta (certa, errada, nula), solicitações de ajuda, tentativas de solução (quantidade), total de respostas certas e erradas (na sessão), seqüência de pedidos de ajuda (antes/depois), estado de espírito (feliz, chateado,indiferente) e repetições.

A construção de um modelo que forneça as informações necessárias é, ainda hoje, um desafio para os sistemas computacionais. Principalmente quando o assunto é considerar emoção e motivação em STI's. Alguns exemplos destes STI's são: Vicente e Pain (1998), Rodrigues e Carvalho (2004) e Andrade *et al.* (2001).

Existe uma linha de pesquisa que estuda especificamente a modelagem de usuários (Conati *et al.*, 2007). Várias técnicas têm sido investigadas para abordar os problemas da modelagem explícita de usuários. Essas técnicas são aplicadas nas três fases do processo de modelagem de usuários: a aquisição; a representação; e a manutenção de modelos de usuário (Langley, 1999).

Módulo de interface

O módulo de interface é responsável pelo fluxo de comunicação de entrada e saída entre o computador e o estudante. A interface pode ser composta por alguns elementos, como janelas, animações, sons, menus, caixas de diálogo, barras, botões e figuras.

É importante salientar que em uma interação com o STI, o estudante não irá somente aprender o conteúdo das lições, mas também terá que aprender como utilizar o sistema. Portanto, a facilidade de uso deve ser uma das considerações principais no projeto destas interfaces. Uma interface consistente ajudará a reduzir a carga cognitiva sobre o estudante (Shneiderman, 1992).

Alguns aspectos devem ser observados numa interface, como a escolha de uma linguagem adequada de comunicação de informações (vindas tanto do sistema quanto do estudante); a escolha dos elementos de interface; a facilidade de uso; e a identificação do usuário. A grande variedade de formas e meios de apresentação existentes fazem com que a interface seja uma das vantagens de uso da computação aplicada ao ensino. Desde a Hipermídia e a Multimídia até a Realidade Virtual, existe uma grande gama de possibilidades de fazer interfaces ergonômicas, amigáveis, eficientes e atrativas para os estudantes.

Para auxiliar na construção dos Sistemas Tutores Inteligentes foi criado por Costa (1997), um modelo de ambiente interativo de aprendizagem baseado no computador denominado MATHEMA.

2.3 Modelo MATHEMA

O modelo MATHEMA, criado por Costa (1997), foi utilizado como uma das bases conceituais deste trabalho, por apresentar um modelo de ambiente interativo de aprendizagem baseado numa arquitetura multiagentes.

O MATHEMA é definido como um modelo de ambiente interativo de aprendizagem baseado no computador, sendo concebido para apoiar atividades que venham a favorecer à realização de interações e seus desdobramentos no processo de aprendizagem. As atividades de aprendizagem são conseqüência do processo de interações envolvendo os seus componentes (Estudante, Tutores) em situações baseadas em resolução de problemas. A aprendizagem é decorrente de atividades provenientes do processo de resolução de problemas.

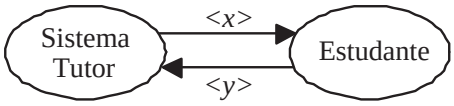

Figura 2.2: Modelo de ambiente de aprendizagem do MATHEMA

De uma maneira mais abstrata, o processo de interação é visto como ocorrendo dentro de um esquema de troca de mensagens (x , y) entre as duas entidades envolvidas (Figura 2.2), obedecendo a um certo protocolo (Costa, 1997) . Neste esquema, o Sistema Tutor formula e envia uma mensagem com conteúdo x para o Estudante, e este reage produzindo e devolvendo-lhe uma mensagem com conteúdo y. Dentre as qualidades necessárias para garantir a efetividade

no processo de interação tornam-se necessárias, do lado do sistema tutor, suporte aos seguintes aspectos:

- conhecimento sobre o domínio;
- conhecimento pedagógico;
- conhecimento sobre o estudante, e
- capacidade de interação.

2.3.1 Modelagem do Conhecimento sobre um Domínio

A modelagem do conhecimento sobre um dado domínio é dividida e organizada segundo duas formas de visualização: uma visão externa e uma visão interna.

A visão externa submete um dado domínio do conhecimento a um particionamento em diferentes subdomínios. Cada subdomínio representa uma visão particular do domínio. Com o objetivo de alcançar um conhecimento com especialidades distribuídas, a busca foi orientada em uma perspectiva tridimensional.

- Contexto: compõem-se de diferentes pontos de vista sobre um domínio de conhecimento, constituindo-se de representações ou abordagens diferentes sobre um mesmo objeto de conhecimento;
- Profundidade: é relativa a um contexto particular, refere-se a um refinamento na linguagem de percepção.
- Lateralidade: refere-se aos conhecimentos de suporte que podem ser apontados para permitir que o estudante adquira conhecimentos relacionados ao domínio da aplicação, proveniente de uma visão particular de contexto e profundidade.

A visão multidimensional mostra a possibilidade de um domínio poder ser enfocado por uma visão contextual, sendo que esta visão pode vir acompanhada de várias alternativas de variações do ponto de vista de profundidade e lateralidade em relação a cada contexto escolhido no domínio (Costa, 1997).

A visão interna equivale a um particionamento do domínio, onde cada subdomínio é composto por um conjunto de unidades pedagógicas (UP) definidas como um currículo, conforme

objetivos de ensino/aprendizagem. As unidades pedagógicas são relacionadas segundo uma ordem definida com base em critérios pedagógicos e a cada uma corresponde um conjunto de problemas. Cada problema está associado a um conhecimento de suporte a resolução, que podem ser conceitos, exemplos, contra-exemplos, dicas, etc.

Simbolicamente um currículo pode ser definido como: Currículo = $UP_1, UP_2, \cdots, UP_n$ onde, UP_1 denota uma unidade pedagógica do currículo estando relacionada segundo uma ordem definida com base nos critérios pedagógicos. A cada UP_i corresponde um conjunto de problemas e a cada problema está associado um conjunto de suporte à resolução, apresentados na Figura 2.3

A Figura 2.3 ilustra a estrutura pedagógica, que é composta por três planos fundamentais, que são denominados de plano pedagógico, plano de problemas e plano de suporte.

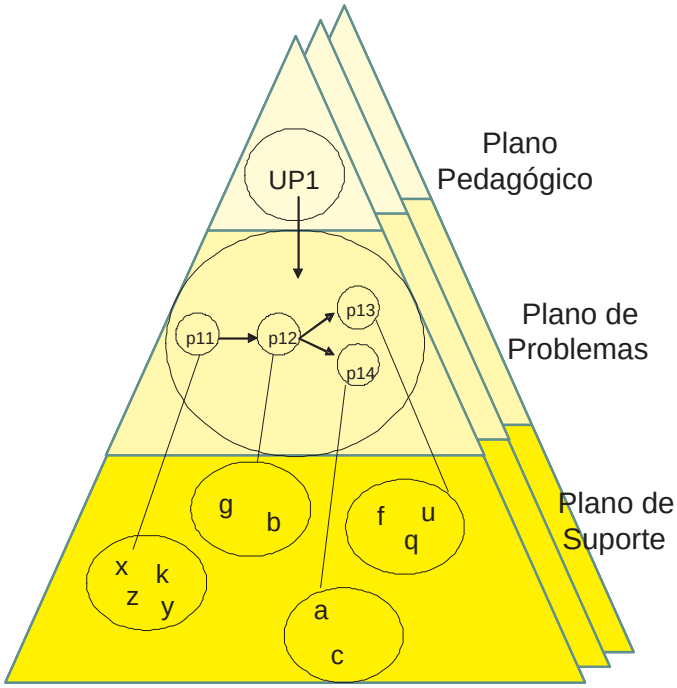

Figura 2.3: Planos

No plano pedagógico (plano superior) estão as unidades pedagógicas relacionadas por uma ordem definida em função de pré-requisitos, servindo para indicar uma ordem de apresentação das atividades pedagógicas.

No plano de problemas (plano intermediário) encontram-se conjuntos de problemas, estando cada um deles associado a uma unidade pedagógica, servindo como atividade pedagógica básica no processo de ensino-aprendizagem. Estes problemas também estão relacionados por uma

ordem parcial, definida de acordo com os pré-requisitos envolvido em suas resoluções.

Finalmente, no plano de suporte (plano inferior) acha-se definido o conhecimento de apoio à resolução dos problemas. Desse modo, tal como já mencionado, incluem-se conceitos, resultados, exemplos (que são as particularidades dos resultados), contra-exemplos, dicas, problemas análogos (da mesma classe), catálogo de erros e mal entendimentos. Neste nível, há os problemas resolvidos e os problemas a serem resolvidos. Quanto aos resolvidos, a cada um deles existe um conjunto de soluções, constando de caminhos diferentes.

2.3.2 Arquitetura MATHEMA

A arquitetura do modelo MATHEMA consiste de três módulos: uma sociedade de agentes tutores artificiais, uma interface de estudante e uma interface de autoria. Na sociedade de agentes tutores artificiais (SATA), cada agente, além das capacidades de comunicação e cooperação, contém um sistema tutor inteligente completo, focado num subdomínio do conhecimento. O fato do sistema consistir de uma sociedade multiagente permite a distribuição dos conteúdos e dados do modelo do estudante entre vários agentes que cooperam no tutoramento.

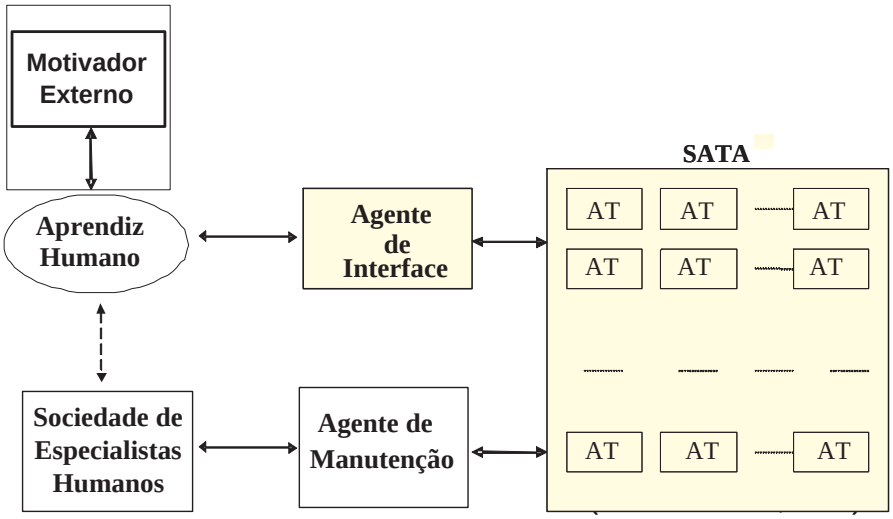

Figura 2.4: Arquitetura do MATHEMA. Extraída de Costa (1997)

A arquitetura do MATHEMA, ilustrada na Figura 2.4, apresenta seis componentes, sendo cada um deles descritos abaixo:

- Aprendiz Humano: agente interessado em aprender algo sobre um dado domínio.

- Sociedade de Agentes Tutores Artificiais (SATA): coleção de agentes que podem cooperar entre si a fim de promover a aprendizagem de um dado estudante em atividade de resolução de problema. Cada agente, denominado Agente Tutor (AT) é especializado em um subdomínio relacionado a um dado domínio de conhecimento. Segunda Costa (1997) essa idéia foi inicialmente inspirada nas reflexões contidas em "Sociedade da Mente", de Marvin Minsky.

- Sociedade de Especialistas Humanos (SEH): fonte integrada de conhecimento externo ao sistema computacional e se comunica com a SATA através dos agentes de manutenção. Dessa Sociedade é requerida a criação e manutenção da SATA (com operações de inclusão, exclusão de agentes, bem como alterações no conhecimento dos agentes) e mais a disposição, em caso de uma falha mais crítica da SATA, de assistir, de um certo modo, os estudantes. A SEH é, portanto, responsável pela incrementabilidade nas capacidades cognitivas da SATA. A SEH mesmo sem ser requisitada por SATA, pode analisar um log com o desenvolvimento das interações entre o estudante e a SATA, avaliando o desempenho de seus agentes tutores para promover melhorias.

- Agente de Interface (AI): representa o elo de ligação entre o Estudante e a SATA. Ele é responsável por desempenhar, primeiramente, o papel de comunicação do agente tutor com o Estudante e vice-versa.

- Agente de Manutenção: representa principalmente um elo de ligação entre a SEH e a SATA, encarregando-se de prover uma interação entre elas, oferecendo meios necessários para percepção, comunicação e manutenção da SATA.

- Motivador Externo: entidades humanas externas que desempenham o papel de quem motiva o Estudante a trabalhar no MATHEMA. Alguns motivadores podem ser um professor do Estudante, seus colegas, etc.

A arquitetura do MATHEMA é utilizada na Ferramenta de Autoria para Sistemas Tutores Inteligentes proposta por Frigo (2007), que é integrada no modelo proposto desta tese.

2.4 Ferramenta de Autoria para STI

Com o objetivo de minimizar os custos em sua construção, vários esforços são conduzidos para a criação de ferramentas de autoria para STI. Segundo Murray (1999) os benefícios a se-

rem alcançados com este tipo de ferramenta são a diminuição de tempo de desenvolvimento e economia de recursos financeiro ou pessoal; organização, relacionamento e estruturação automática dos conteúdo inseridos no curso; deixar de forma transparente para o professor/autor a complexidade de um STI; prototipação rápida e eficiente.

A construção de um STI requer uma integração dos quatro componentes (modelos domínio, estudante, pedagógico e interface) descritos na Seção 2.2.1. A ferramenta de autoria pode ser classificada em sete tipos, conforme suas características:

1. Currículo: Organização e Planejamento.
2. Ensino de Estratégias.
3. Simulação de dispositivo e Treinamento de Equipamento.
4. Domínio de Sistema Especialista.
5. Tipos de Conhecimento múltiplos.
6. Propósito especial.
7. Hipermídia Inteligente/ Adaptativa.

De acordo com Murray *et al.* (2003), as fronteiras entre as classificações não são bem definidas. Existem algumas ferramentas de autoria que possuem características combinadas. Se uma ferramenta de autoria possuir em sua composição características diversas, ela poderá oferecer um nível de interação maior.

Um ambiente de aprendizagem envolve diversos atores, cada qual com um papel definido no ambiente. Estes atores são:

- Autores: são os detentores do conhecimento técnico de domínio da ferramenta, especialistas em artes gráficas e programadores responsáveis pela criação da estrutura básica do curso.
- Professores: são os detentores do conhecimento do domínio do problema, são os responsáveis pelo oferecimento do curso usando o material básico criado pelos autores. Eles podem criar complementos locais ao material básico, tais como listas de exercícios, descrição de ferramentas de apoio disponíveis localmente e sugestões de projetos a desenvolver com tais ferramentas. Devem supervisionar os monitores, podendo nesta tarefa

ter de analisar e responder dúvidas dos estudantes. A partir da análise de dúvidas e das anotações dos estudantes sobre o texto, os professores podem criar revisões do complemento local ou gerar comentários para os autores. Finalmente, estes atores são os responsáveis pela avaliação ou acompanhamento de desempenho dos estudantes.

- Monitores: são auxiliares da atividade de ensino, que têm acesso a dúvidas dos estudantes e podem responder à maior parte destas dúvidas sem recorrer aos professores. O monitor deve oferecer maior disponibilidade para este tipo de tarefa do que o professor, de forma que o estudante tenha um retorno mais rápido às suas dúvidas. Para tanto, diversos indivíduos podem atuar de forma colaborativa para representar o papel de monitores.

- Estudantes: são os atores principais no ambiente colaborativo, com amplo acesso ao material disponibilizado. Possivelmente um estudante terá também acesso a outras dúvidas já manifestadas sobre um tópico, assim como em sala de aula ele pode ouvir questões de colegas e respostas a estas questões. Caso o curso considere trabalhos em grupo, ferramentas de colaboração entre estudantes deverão ser oferecidas. O apoio para atividades de co-aprendizagem é necessário para estudantes dispersos geograficamente.

- Desenvolvedor: Desenvolvedor: são os profissionais da computação (programadores) que especificam o modelo do estudante, especificam os controles para os cenários/estratégias pedagógicas baseadas em informações dos modelos, promovem a manutenção nas sociedades de agentes artificiais no que diz respeito aos processos de comunicação e interação.

Para apoiar as atividades colaborativas será preciso oferecer ferramentas para o trabalho colaborativo entre os membros de cada um dos atores descritos acima (entre autores, entre professores, entre monitores e principalmente entre estudantes) e a cooperação entre estes grupos, por exemplo, a interação professor-estudante, estudante-estudante, etc.

Durante a última década vários trabalhos relacionados à ferramenta de autoria para STI foram projetados e implementados. Murray (1999) lista mais de vinte referências de ferramentas de autoria em seu artigo. Nesta tese é utilizada a FAST (Ferramenta de Autoria para Sistema Tutor) desenvolvida no Laboratório de Controle e Microinformática do departamento Automação e Sistemas da UFSC (Frigo, 2007) que utiliza o modelo MATHEMA.

2.5 FAST - Ferramenta de Autoria para Sistema Tutor

Frigo (2007) propõe em sua tese um modelo formal de adaptação para STI, implementado na ferramenta de Autoria FAST. A definição do modelo é baseada em Ontologia (ver Apêndice A.3) para a representação do conhecimento envolvido no modelo de domínio e do estudante e Redes de Petri Objetos (ver Apêndice A.4), para definir o modelo pedagógico.

As transições nestas Redes de Petri Objetos (RPO) controlam as interações com o estudante, escolhendo os conteúdos do domínio a serem apresentados a cada momento de acordo com as informações armazenadas no modelo do estudante, bem como a determinação dos conteúdos mais apropriados para um determinado estudante, são feitos automaticamente pelo modelo pedagógico. O professor especifica somente o conteúdo referente ao domínio do curso, estabelecendo pré-requisitos e níveis de dificuldades de acordo com a estrutura do modelo de autoria (Cardoso *et al.*, 2004a) (Cardoso *et al.*, 2004b).

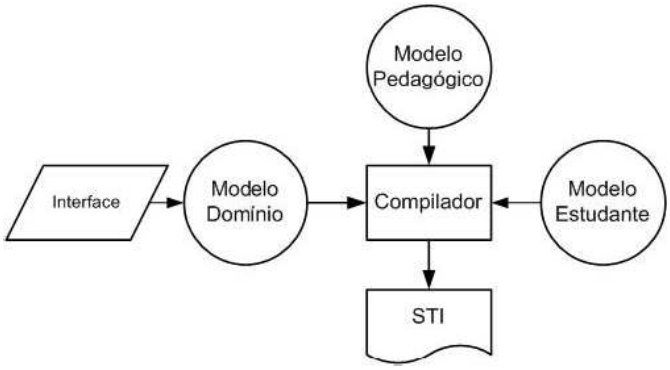

Figura 2.5: Modelo da FAST. Extraída de Frigo (2007).

O modelo formal de adaptação para STI que permite a integração dos modelos, apresentado na Figura 2.5, é formado pelos seguintes ítens:

- Uma interface interativa onde o professor adiciona o conteúdo do curso de acordo com a estrutura do modelo do domínio. Os pré-requisitos associados as UP e aos problemas são definidos em forma de grafos.

- Um compilador que transforma a definição do modelo do domínio do STI, vista como uma instância de respectiva ontologia, em uma rede de Petri que implementa a semântica do curso. Atualização das informações do modelo do estudante são automaticamente introduzidos na RPO. A RPO resultante é usada para controlar um currículo de um agente da SATA.

A FAST é integrada no Arcabouço para Sistemas Tutores Inteligentes (ASTI) para a concepção de cursos, que é composto também pela SATA do Modelo MATHEMA.

2.5.1 Representação dos modelos

As representações dos modelos de estudante e domínio apresentadas por (Frigo, 2007) foram inspiradas no trabalhos de Chen e Mizoguchi (2004). Estes modelos foram incrementados com as informações referentes aos grupos que são abordadas na Seção 4.2.

A representação do modelo pedagógico é feita utilizando RPO e utiliza um controle em dois níveis, do Currículo e das Estratégias.

Nível do Currículo

O nível do Currículo (RPO-CV) especifica os percursos possíveis da disciplina de acordo com suas Unidades Pedagógicas (UP) e problemas (Pb), respeitando os pré-requisitos definidos pelo professor, e o nível das Estratégias especifica a interação entre estudante e sistema tutor durante a resolução de um problema específico. Ambos os níveis são representados por RPO-CV organizadas de forma hierárquica.

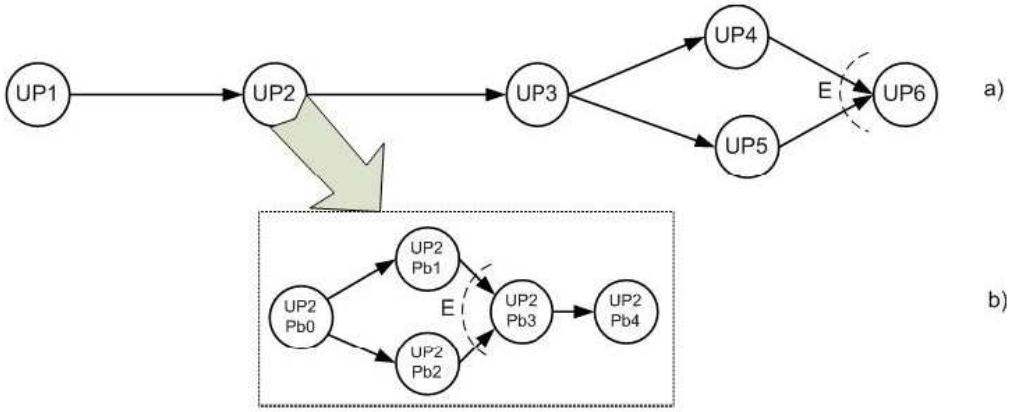

Figura 2.6: RPO-CV organizada de forma hierárquica. Extraído de Frigo (2007).

A RPO-CV é gerada automaticamente por um compilador (ver Figura 2.5), à partir dos dois grafos de pré-requisitos que constituem o modelo do domínio, definidos pelo professor.

- grafo $Gr - UP$, que define as relações entre as *k* unidades pedagógicas, como representado na Figura 2.6 parte (a), onde *k* = 6 (unidades UP1 à UP6);

- grafo $Gr - Pb$, que define as relações entre os n problemas de uma unidade pedagógica, como representado na Figura 2.6 parte (b), que mostra a relação entre os 5 problemas que definem a UP2.

Cada uma das demais UP da Figura 2.6 parte (a) deve ser refinada num grafo definindo as relações entre os seus problemas como o da Figura 2.6 parte (b).

O compilador (ver Seção D) combina o grafo $Gr - UP$ e os *k* grafos $Gr - Pb$ referentes às unidades pedagógicas de $Gr - UP$ em único grafo $Gr - CV$, substituindo cada nó UP_i do grafo $Gr - UP$ pelo grafo $Gr - Pb_i$ correspondente.

A RPO-CV do Currículo é construída seguindo as etapas:

1. Traduzir o grafo do currículo ($Gr - CV$) numa rede de Petri.

2. Transformar a rede de Petri do currículo em uma RPO de modo a considerar diretamente o modelo do estudante.

3. Adicionar lugares de comunicação para permitir a interação com o nível das Estratégias.

Nível das estratégias

No nível das estratégias, chamado de RPO-E, as redes não são definidas pelo professor e sim pelo desenvolvedor. A rede controla as interações na solução de problemas, levando em consideração o retorno e os atributos do modelo do estudante. O objetivo da RPO-E é apresentar ao estudante as unidades de interações associadas com um determinado problema.

Os tipos de interações no nível das estratégias com o estudante podem ser muito mais flexíveis e o objetivo neste nível é unir um conjunto pré-definido de unidades de interação com os protocolos de interação. As unidade de interação podem ser de diferentes tipos, por exemplo, elas podem usar diferentes tipos de mídias ou ainda requerer diferentes tipos de respostas do estudante. A combinação destas unidades com os protocolos de interação que implementam um cenário de aprendizagem é de responsabilidade dos desenvolvedores do sistema.

Um exemplo para o nível das estratégias é representado na Figura 2.7. Esta RPO-E apresenta explicações, exemplos e exercícios em uma ordem e em nível de detalhe determinados pelas informações do modelo do estudante e pelo retorno por ele fornecido. O conteúdo é definido pelo professor e para cada conteúdo deve ser especificado seu nível de dificuldade e eventualmente o público ao qual se destina.

O lugar *Inicio* (figura 2.7) está associado com a unidade de interação introdutória onde, as questões centrais associadas com o problema são mostradas. Os atributos do modelo do estudante usados neste nível de interação são, por exemplo, nível educacional, preferências, mídia, etc.

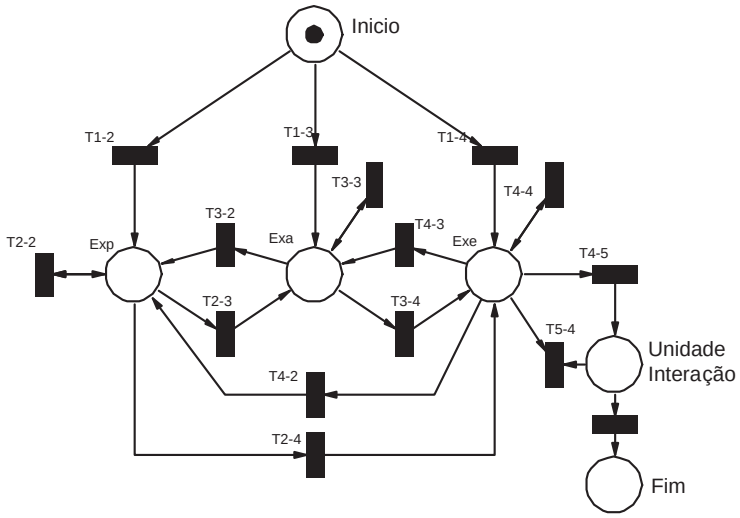

Figura 2.7: Exemplo de RPO-E.

Os lugares *Exp, Exa, Exe* estão associados com as unidades de interação que apresentam ao estudante, respectivamente, explicações, exemplos e exercícios. Cada vez que uma ficha é colocada em um destes lugares, o conteúdo correspondente é apresentado ao estudante. A atividade é diferente de acordo com as interações e com os resultados destas. Todas estas informações são automaticamente armazenadas no modelo do estudante (Frigo, 2007).

A transição é um exemplo de como uma decisão que depende do retorno do estudante pode ser implementada na RPO e representa uma solicitação feita pelo estudante ao sistema responsável pela correção do exercício. A solicitação é representada pela flecha. Depois disso, se o estudante obtém a nota mínima, a transição é disparada e o estudante está apto a continuar suas interações. Caso contrário, a transição é disparada e o estudante deve refazer o exercício.

Controle multi-níveis

O controle multi-níveis permite a comunicação entre o nível de Currículo e as Estratégias. A RPO-CV indica qual o problema de uma Unidade Pedagógica é apresentado ao estudante, e a RPO-E controla as atividades de resolução deste problema, decidindo que tipo de conteúdo e em que ordem ele deve ser apresentado.

O mecanismo que permite a troca de informação entre as RPOs está ilustrado na Figura 2.8, que mostra: a) rede completa e b) é a rede dividida em duas redes N1 e N2. A rede N1 corresponde à RPO-CV e a N2 corresponde às Estratégias.

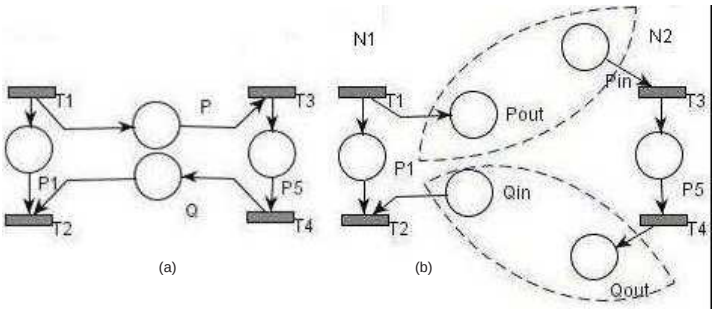

Figura 2.8: Comunicação entre as RPO-CV e RPO-E.

O lugar *P* na Figura 2.8 parte (a) está separado em dois lugares na Figura 2.8 parte (b): P*out* em N1, e P*in* em N2, chamados *lugares de comunicação*. O mesmo processo é feito para o lugar *Q*. Quando uma transição T*1* é disparada na rede completa da Figura 2.8 parte (a), os lugares P*1* e *P* estão marcados, e então a transição T*3* está habilitada. Nas redes correspondentes N1 e N2 da Figura 2.8 parte (b), após o disparo de T*1* os lugares P*1*, P*out* e P*in* são marcados, e T*3* está habilitada. Logo a rede completa e as redes divididas são equivalentes, e este mecanismo de passagem de ficha permite implementar uma rede em um ambiente distribuído ou em um contexto de RPOs hierárquicas. A ficha colocada em P*out* representa uma mensagem enviada para um agente e uma ficha em P*in* representa uma mensagem recebida. Uma ficha em P*1* significa que uma atividade associada foi iniciada.

O uso dos dois níveis de controle também distribui o carregamento do sistema quando diversos estudantes estiverem acessando o sistema simultaneamente.

2.5.2 Funcionamento da FAST

O Autor deve modelar o domínio de acordo com a definição do MATHEMA. Feita a modelagem, ele utiliza a interface de autoria para inserir os grafos de pré-requisitos das Unidades Pedagógicas (UPs) e na seqüência os grafos de pré-requisitos dos Problemas (Pbs). Para cada Problema são fornecidos os conteúdos pelo Professor necessários à resolução dos mesmos. O tipo de informação fornecida, por exemplo, exercícios e explicações, está relacionado com as estratégias pedagógicas utilizadas pelo professor.

As redes de Petri do nível das estratégias (RPO-E) são criadas diretamente pelo desenvolvedor da ferramenta e devem ser suficientemente genéricas para permitir múltiplas estratégias.

A Figura 2.9 ilustra as etapas da FAST a partir dos grafos de pré-requisitos descritos pelo professor e da rede RPO-E fornecida pelo desenvolvedor.

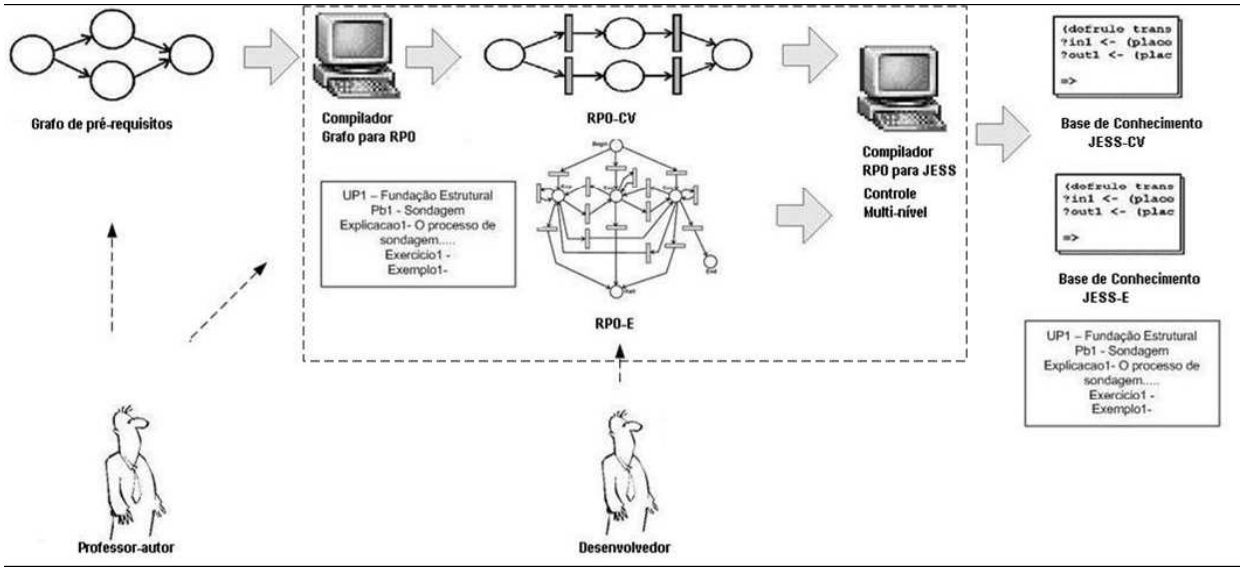

Figura 2.9: Funcionamento da FAST. Extraída de Frigo (2007).

Na FAST, o compilador utiliza as informações dos grafos (UPs e Pbs) juntamente com as informações conceituais do modelo do estudante e constrói as RPOs do nível do currículo (RPO-CV). Para que as RPOs sejam executadas, elas são transformadas em regras que são interpretadas pelo motor de inferência de um Sistema Especialista (*Java Expert System Shell* JESS).

Para executar as RPOs, utiliza-se um shell de Sistemas Especialistas baseado em regras, regras estas que implementam uma RPO. Além disso, o uso de um sistema de regras traz a possibilidade de se adicionar novas capacidades/habilidades, por meio de regras específicas. A opção de tal shell recaiu sobre o JESS (ver Apêndice B), devido sobretudo a sua forte integração com Java.

A saída da FAST é um conjunto de regras JESS que são utilizadas para a construção de um STI, assim como as páginas HTML (*HyperText Markup Language*) com o conteúdo dos problemas fornecido pelo professor.

Maiores detalhes referente o compilador para transformar grafo de pré-requisitos em regras são apresentado no Apêndice D.

A seguir, são abordados os Sistemas Multiagentes que são utilizados na implementação de STI's. Com o uso da tecnologia de agentes é possível desenvolver STI's mais próximos de ambientes reais de ensino, onde cada entidade pertencente ao STI é representada como um agente.

2.6 Sistemas Multiagentes

A IAD (Inteligência Artificial Distribuída) pode ser dividida em duas grandes áreas, que são, a Resolução Distribuída de Problemas e os Sistemas Multiagentes (SMA) (Bond e Gasser, 1988) e (Bittencourt, 1998). Em qualquer uma destas duas áreas, usa-se a designação agente para as entidades que participam das atividades de solução de problemas. A grande diferença pode ser observada pela autonomia destes agentes (Lin, 2005), (Wooldrige, 2002), (Ferber, 2003),(Muller, 1998), (Nwana, 1995) e (Moulin e Chaib-Draa, 1996).

Sistemas Multiagentes (SMA) são sistemas compostos por múltiplos elementos computacionais interativos denominados agentes (Wooldrige, 2002). As características básicas de Sistemas Multiagentes são (Ferber, 1999):

- a comunidade é concebida para cooperar em um ambiente aberto, onde cada agente tem autonomia e pode participar da resolução de problemas;
- eventualmente, um agente pode resolver sozinho um problema;
- os agentes competem entre si pelos recursos e precisam saber lidar com conflitos e coordenar atividades para aumentar a eficiência na solução de problemas;
- os agentes não precisam utilizar a mesma linguagem, implicando a necessidade de traduções e mapeamentos para as representações individuais;
- os agentes possuem apenas uma visão parcial do ambiente, não possuindo informações completas, e
- o controle do sistema é descentralizado, as informações distribuídas e o processamento é assíncrono.

Em um SMA, os agentes devem possuir algumas capacidades específicas para interagirem num mesmo ambiente. Portanto, os agentes devem ter conhecimento da sua existência e da

existência dos outros agentes. Devem ser capazes de se comunicar possuindo, para tanto, uma linguagem específica. Cada agente deverá possuir conhecimentos e habilidades para executar uma determinada tarefa e, portanto, devem cooperar para atingir um objetivo global.

2.6.1 Definição de Agentes

Em um SMA encontram-se normalmente dois tipos diferentes de agentes, os agentes artificiais (módulos de software) e os agentes humanos (usuários).

Existem inúmeras definições de agente, umas mais elaboradas do que outras, introduzindo propriedades mais exigentes e que ao mesmo tempo são mais subjetivas: é o caso da autonomia e da inteligência, por exemplo.

Uma definição abrangente de agente é proposta por Ferber (1999), que define um agente como uma entidade virtual ou real, que está apta a perceber e representar parcialmente seu ambiente. Um agente também possui a capacidade de se comunicar com outros agentes e pode possuir um comportamento autônomo, que é uma conseqüência de suas observações, de seu conhecimento e de suas interações com outros agentes.

O Autor Wooldrige e Jennings (1995) define um agente inteligente como um programa de computador que desenvolve tarefas delegadas por usuário de forma autônoma. São sistemas que apresentam um comportamento determinado de acordo com um processo de raciocínio, e o processo de raciocínio está baseado na maneira pela qual o agente representa suas crenças, desejos e compromissos.

Segundo Demazeau (1990), os agentes devem possuir alguns comportamentos que estão baseados em dois critérios, como a localização da tarefa a ser executada pelo agente, local ou global, e a capacidade do agente de executar sozinho a tarefa, apto ou não-apto.

Assim, de acordo com estes critérios, são definidos alguns comportamentos possíveis para os agentes:

- Coabitação: um agente deve realizar uma tarefa com sucesso e deve ser apto a realizá-la sozinho.

- Cooperação: quando um agente não estiver capacitado para realizar sozinho uma tarefa, ele deve pedir auxílio para outros agentes. Esta cooperação deve ocorrer ainda quando outros agentes podem executar mais eficientemente a mesma tarefa.

- Colaboração: alguns objetivos globais podem interessar a todos os agentes e podem ser realizados individualmente por vários agentes. Deve-se, então, eleger um agente para executar a tarefa.

- Distribuição: algumas tarefas globais podem ser realizadas coletivamente por mais de um agente. O principal problema está em dividir a tarefa global e distribuí-la entre os agentes cooperativos.

Segundo Moulin e Chaib-Draa (1996), um agente é uma entidade autônoma que deve possuir várias habilidades, tais como:

- percepção e interpretação de dados de entrada e mensagens;

- raciocínio sobre suas crenças;

- tomada de decisão (seleção de objetivos);

- planejamento (seleção ou construção de planos de ações, resolução de conflitos e alocação de recursos), e

- habilidades de executar planos incluindo envio de mensagens.

2.6.2 Classificação dos Agentes

Os agentes que compõem um SMA podem ser classificados segundo as seguintes dimensões: Estacionários ou Móveis, Persistentes ou Transientes, Reativos ou Cognitivos e de Software ou de hardware (robôs).

Agentes Estacionários são agentes de software que não possuem a habilidade para se mover de um ambiente computacional para outro através da rede. Agentes **móveis** são agentes de software que podem se mover para outros ambientes através da rede e, quando se movem, levam consigo seus estados internos, ou seja, sua representação mais a memória (Nwana, 1995).

Agentes Persistentes são agentes de software que, uma vez lançados em um dado ambiente computacional, não podem ser excluídos do sistema. Agentes **temporários** são agentes de software que têm uma vida finita, normalmente de duração igual ao tempo de uma dada tarefa, ou seja, ao concluírem sua missão eles são excluídos do sistema, normalmente por eles próprios.

Agentes Reativos realizam uma ação como reação à outra ação efetuada sobre eles. Os agentes reativos se comportam segundo o modo estímulo-resposta, ou seja, não há uma memória

sobre ações realizadas no passado e nem previsão de ações que poderão ser executadas no futuro. Sua capacidade interna realiza apenas associações de entrada e saída.

Agentes Cognitivos são visualizados como sistemas intencionais, ou seja, possuem estados mentais de informação e manipulam o conhecimento. Entre estados mentais estão as seguintes características: crenças, conhecimento, desejos, intenções, obrigações, etc. Estes estados mentais são representados internamente nos agentes. Estes agentes também são ditos sociais porque, além de manipular o seu conhecimento, eles conhecem as crenças, objetivos e motivações dos elementos que os cercam.

Agente de Software define-se como um programa de computador que executa uma determinada tarefa usando informações provenientes de seu ambiente para completá-la. O software pode ser apto a adaptar-se baseado nas mudanças que estão ocorrendo no seu ambiente, gerando, desta maneira, o resultado desejado.

Agente de hardware geralmente é configurado para realizar uma tarefa específica, por exemplo, sensores para medir temperaturas ou robôs para realizar uma determinada tarefa.

2.6.3 Interação entre os Agentes

A interação é uma das propriedades essenciais de um agente e consiste na capacidade de comunicar-se com outros agentes, usuários e sistemas visando atingir seus objetivos. A interação pode ser dividida em camadas de complexidade, tais como comunicação, coordenação e cooperação (Wooldrige, 2002).

A camada de comunicação é básica de qualquer software que precisa interagir. Faz-se necessária uma linguagem de comunicação com uma sintaxe precisa e de conhecimento de todos.

Os SMA podem interoperar com outros sistemas para garantir agilidade na execução das tarefas. Esta interoperabilidade pode ser realizada de agente para agente, agente para usuário, agente para SMA e SMA para sistemas.

- Agente com agente: agentes do mesmo SMA que se comunicam através de um protocolo geralmente proprietário, tornando mais leve a comunicação realizada, pois são transmitidas somente informações relevantes para o sistema.

- Agente com usuário: normalmente realizado através de uma interface gráfica quando o agente precisa da informação de um usuário para completar uma tarefa.

- Agente com SMA: agentes em um SMA que se comunicam com agentes de outro SMA, acessam serviços "exportáveis" pelo SMA, utilizando protocolos padronizados para haver compatibilidade e interoperabilidade entre os sistemas.

- SMA com sistemas: um agente se comunica com um sistema de informação no estilo cliente-servidor. O agente acessa serviços disponibilizados pela API (*Application Program Interface*) do sistema para colher informações necessárias a ele.

- Agente com equipamentos: onde agentes trocam informações com máquinas para que estas realizem tarefas ou para colher informações, por exemplo do estado desta, usando protocolo de chão de fábrica.

Existem três critérios para a classificação de uma sociedade de agentes quanto ao tipo de interação (Oliveira, 1996):

- Quanto ao tipo de agentes: em sociedades homogêneas os agentes são todos do mesmo tipo, ou seja, possuem arquiteturas idênticas. Já nas sociedades heterogêneas existem agentes de diferentes tipos.

- Quanto à migração de agentes: em sociedades fechadas existe um número fixo e único de agentes presentes na sociedade. Nas sociedades abertas o número de agentes pode variar, pois podem entrar novos agentes ou sair agentes da sociedade.

- Quanto à presença de regras de comportamento: nas sociedades baseadas em leis existem regras que determinam o comportamento dos agentes. Em sociedades sem lei não há regras para reger os agentes que fazem parte da sociedade.

Para que uma sociedade de agentes coopere a fim de atingir um determinado objetivo, é necessário que seja bem definida uma arquitetura que possibilite a interação entre estes agentes, (Rodrigues *et al.*, 2003).

Nestas interações ocorre troca de conhecimento, objetivos, planos ou escolhas através da comunicação, que pode ser direta ou indireta. Na comunicação direta, os agentes se conhecem e, por isso, trocam informações diretamente entre si. Na comunicação indireta, os agentes não se conhecem e, desta maneira, a comunicação irá ocorrer através de uma estrutura de dados compartilhada. Um exemplo da comunicação indireta é através de uma estrutura denominada *blackboard*.

2.6.4 Negociação, Comunicação e Cooperação

A negociação é realizada entre os agentes, dividindo a execução das tarefas, de maneira que seja mais organizada e fazendo uso das capacidades e conhecimentos dos agentes. É um processo onde vários agentes buscam um acordo diante de um conflito de interesses, recursos ou habilidades. Este acordo pode envolver troca de informações, recursos ou habilidades (Saywell, 2002).

Moulin e Chaib-Draa (1996) definem negociação como o processo de aperfeiçoar o consenso (reduzindo incerteza e inconsistência) em um ponto de vista, através da troca estruturada de informações relevantes. Para que os agentes possam negociar, eles devem fazer uso de protocolos de comunicação, que servem de estrutura para implementação e são utilizados na difusão das mensagens.

Existem vários protocolos de negociação na literatura de IAD. Um protocolo de negociação especifica os tipos de transações que os agentes podem fazer, bem como as sequências de oferta e contra-ofertas que são permitidas.

Um dos protocolos mais empregados é o Protocolo de rede de contratos (*Contract Net Protocol - CNP*) (Smith, 1980), baseado no processo de licitação em organizações humanas.

Segundo Oliveira (1996) este protocolo é usado para realizar o processo de negociação entre agentes. Nas redes de contrato, um agente especial, chamado gerente, envia mensagens para todos os agentes da sociedade. Essas mensagens são solicitações de propostas que o gerente envia para os outros agentes, para a realização de uma tarefa. Os agentes que possuam capacidade de realizar a tarefa submetem suas propostas para o gerente, que as avaliará segundo algum critério pré-definido. O agente contratante irá comunicar aos agentes a proposta selecionada, determinando, deste modo, quais são os nodos escolhidos, chamados agentes contratados, que ficarão encarregados de executar a tarefa. Além destes, existem vários outros protocolos que podem ser usados para os mais variados tipos de comportamentos desejados, como por exemplo os Leilões e Estratégias de negociação.

A cooperação ocorre entre dois ou mais agentes, quando eles necessitarem realizar uma mesma tarefa a qual não são capazes de fazer individualmente. Desta maneira, os agentes precisam compartilhar seus conhecimentos, já que possuem apenas conhecimento parcial a respeito do problema.

Existem dois tipos de cooperação, a partilha de informação e a partilha de tarefas. Na partilha de informação um dado agente dispõe ou produz informações parciais que julga serem

úteis a partilhar e as envia para os outros agentes. A partilha de tarefa é efetuada quando um dado agente, ao decompor uma dada tarefa, detecta subtarefas que não poderá realizar, sendo necessário procurar outros agentes que possam auxiliá-lo.

Moulin e Chaib-Draa (1996) propõem quatro objetivos genéricos para a cooperação em um grupo de agentes:

- aumentar a rapidez de conclusão da tarefa através do paralelismo;
- aumentar o conjunto ou escopo de tarefas concluídas, pelo compartilhamento de recursos;
- aumentar a probabilidade de conclusão de tarefas, pelo empreendimento de tarefas duplicadas, possivelmente com diferentes métodos de realização daquelas tarefas, e
- diminuir a interferência entre tarefas, evitando interações prejudiciais.

2.6.5 Organização de SMA

É necessário que algumas questões sejam tratadas para formar um grupo social, como por exemplo a organização e cooperação. A organização que diz respeito a como os agentes interagem entre si, e qual o tipo de organização que eles adotam; e a cooperação quando um agente não estiver capacitado para realizar sozinho uma tarefa, ele deve cooperar com outros agentes.

A organização de um SMA pode ser classificada em três tipos como democrática, federada ou hierárquica (exemplificada na Figura 2.10). Na organização democrática, os agentes não possuem organização e atuam em graus similares de independência e autonomia, sem hierarquia alguma. Neste caso, são necessários procedimentos de coordenação sofisticados para garantir convergência e evitar interações desnecessárias.

Na organização federada existe algum tipo de hierarquia. Há a presença de agentes facilitadores, agentes intermediários entre o cliente e o supervisor. Grupos de agentes da federação têm graus similares de independência e autonomia, e normalmente são bastante heterogêneos. Os procedimentos de coordenação são de média complexidade, devido à própria existência do facilitador, que garante mecanismos de convergência e/ou de distribuição de tarefas (Wooldrige, 2002) (Ferber, 1999).

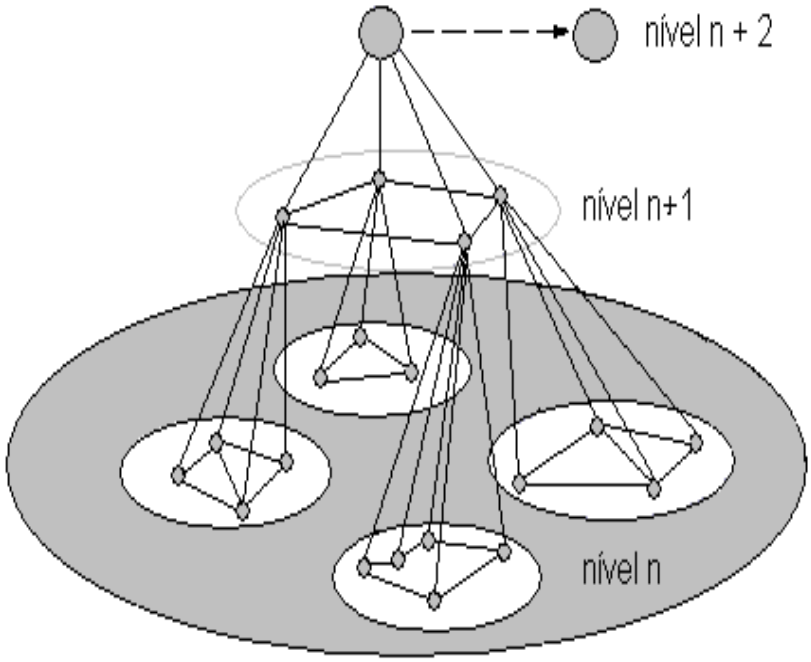

Figura 2.10: Exemplo de organização hierárquica Ferber (1999).

2.7 Conclusão

O aprendizado em grupo colaborativo é uma atividade natural mesmo em ambientes de ensino tradicional, onde os estudantes interagem para entender ou para sanar alguma dúvida sobre o conteúdo pedagógico. É importante que os STI´s ofereçam esse tipo de abordagem como uma ferramenta de ensino-aprendizagem coletiva que visa criar uma dinâmica pedagógica mais rica do que a interação humano-computador.

Neste trabalho o termo aprendizado colaborativo consiste na divisão de um trabalho entre os participantes, em que cada estudante é responsável por uma parte da resolução do problema; os participantes trabalham no sentido de atingir um objectivo comum.

Os STI´s podem ser implementados como Sistemas Multiagentes. Os SMA´s são bastante flexíveis e são formados por diversos e distintos agentes. Com o uso da tecnologia de agentes é possível desenvolver STI´s mais próximos de ambientes reais de ensino, onde cada entidade pertencente ao STI é representada como um agente, por exemplo, o professor/monitor é representado por um agente-tutor.

Os SMA‘s visam solucionar problemas de forma distribuída, onde cada agente possui uma certa autonomia e é responsável por uma determinada tarefa. A utilização de um sistema mul-

tiagente é justicável num processo de aprendizagem em grupo, por ser naturalmente distribuído, sendo composto de três entidades básicas: o estudante que interage com o tutor, professores que elaboram o conteúdo e os agentes especialistas (por exemplo agentes supervisores e coodenador) que monitoram a interação dos estudantes no grupo.

Este trabalho estende o modelo MATHEMA, que é um modelo para concepção e desenvolvimento de ambientes de aprendizagem baseado numa arquitetura multiagentes, e integra a ferramenta de autoria FAST para o aprendizado individualizado. Este aproveitamento do modelo e da ferramenta é importante para proporcionar também o aprendizado individualizado aos estudantes no trabalho de grupo.

É utilizado a Ontologia para a representação do conhecimento envolvido no modelo de domínio e do estudante, e Redes de Petri Objetos, para definir o modelo pedagógico.

Para formalizar e representar as interações, dos estudantes e do grupo, optou-se utilizar as Redes de Petri Objeto. RPO é uma ferramenta gráfica e matemática que se adapta bem a um grande número de aplicações em que as noções de eventos e de evoluções simultâneas são importantes.

Capítulo 3

Trabalhos Relacionados

Neste capítulo são apresentados trabalhos de aprendizado em grupos suportados por computador em Sistemas Tutores Inteligentes. Neste levantamento bibliográfico foi focado o modelo (agentes, ontologia, protocolo/RPO e cenários). A pesquisa não abrangeu trabalhos restritos ao gerenciamento de grupos, por exemplo o acompanhamento de interações entre os estudantes (conversas) ou aplicações específicas de técnicas pedagógicas.

3.1 REDEEM

A REDEEM (Ainsworth, 2007) é uma ferramenta desenvolvida pelo Instituto de Pesquisa de Ensino e Psicologia da Universidade de Nottingham da Inglaterra. A REDEEM é uma ferramenta de autoria que permite aos professores transformar seus materiais pedagógicos em um STI que apresenta uma forma diferenciada e instruções adaptáveis para a sua classe.

Nesta ferramenta uma única disciplina pode ser adaptada para atender as necessidades de uma variedade de estudantes e de aplicações diferentes. Para atender a este conjunto de configurações são utilizadas regras de produção e redes semânticas (Ainsworth, 2007).

O Professor/Autor pode definir as estratégias pedagógicas, conforme a Figura 3.1 extraída de (Ainsworth, 2007), e incrementar o material para permitir seqüencias alternativas, variações no conteúdo do curso, um conjunto de questões, estratégias de interação, uma variedade de estilos de ensino, como alguns aspectos importantes, tais como, prover ajuda, controle de estudante, dificuldade e posição de perguntas.

Na REDEEM, primeiramente o Autor nomeia as páginas e as classifica de acordo com o tipo de material (fácil, geral, introdutório...); após descreve as relações entre as sessões, por exemplo

as relações de pré-requisitos; fornece as questões e o retorno dado ao estudante que explica porque a resposta é correta; define um conjunto de categorias de estudantes em qualquer grau de granularidade, normalmente baseada em desempenho ou tarefas; e poderá definir estratégias como estudo sequencial/guiado ou livre.

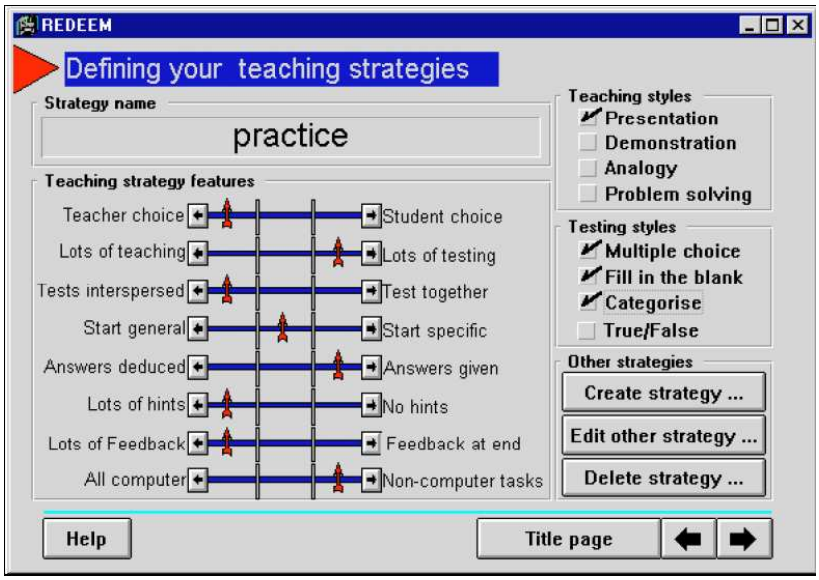

Figura 3.1: Interface de definições de estratégia pedagógica do REDEEM

Segundo Ainsworth (2000), o Professor/Autor classifica os estudantes em classes virtuais e cria estratégias diferenciadas para cada estudante da classe. Porém até o momento, conforme Ainsworth (2007), os autores criaram uma única estratégia para cada grupo ou estudante individual.

Esta ferramenta possui várias limitações, pois não existe suporte para as atividades entre os grupos e o conjunto de ações capturadas dos estudantes são limitadas. Os grupos são definidos na REDEEM como um conjunto de categorias de estudantes.

3.2 WHITE RABBIT

WHITE RABBIT é um sistema desenvolvido no Departamento de Informática e Pesquisa Operacional da Universidade de Montreal no Canadá (Thibodeau *et al.*, 2000). Tem como objetivo aumentar a cooperação entre um grupo de pessoas pela análise de suas conversações. Cada usuário é assistido por um agente inteligente, que estabelece um perfil de seus interesses. Com o comportamento móvel e autônomo, o agente pesquisa agentes pessoais de outros usuários,

afim de encontrar aqueles que tenham interesses comuns. E então os colocam em contato.

Um agente Mediador/assistente é usado para facilitar a comunicação entre os agentes pessoais e para realizar agrupamentos nos perfis que eles tenham recolhido. Esta abordagem usa agentes inteligentes para descobrir os interesses particulares de um grupo de pessoas trabalhando em um domínio particular com a intenção de colocá-los em contato para aumentar o nível de cooperação. Os agentes analisam a conversação entre os usuários através de um chat, com fim de construir para cada um deles um perfil de seus interesses.

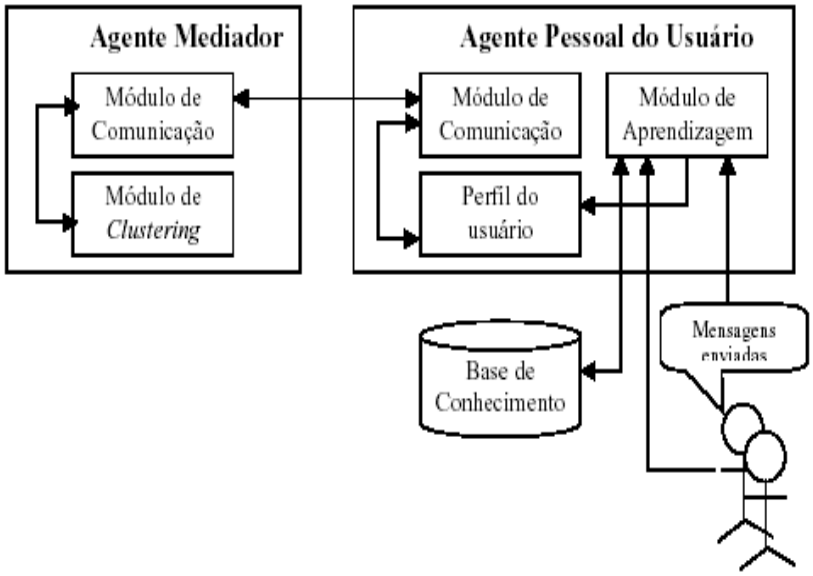

Figura 3.2: Arquitetura do agente Pessoal do sistema WHITE RABBIT.

Nas Figuras adaptadas de Thibodeau *et al.* (2000), a Figura 3.2 ilustra a arquitetura do agente Pessoal e a Figura 3.3 ilustra a arquitetura geral do sistema que é constituída por 6 seções principais, que são:

- Uma camada Voyager dá aos agentes mobilidade e autonomia, onde Voyager é uma arquitetura proprietária para agentes móveis da ObjectSpace. A plataforma Voyager tem como objetivo dar suporte a objetos móveis e agentes autônomos. Desenvolvida em Java, esta plataforma suporta vários padrões, tais como: CORBA (*Common Object Request Broker Architecture*) e RMI (*Remote Method Invocation*). No contexto deste projeto, o módulo de comunicação está baseado inteiramente nesta arquitetura, permitindo aos agentes do sistema comunicar-se uns com os outros.

- Um servidor de chat que organiza o fluxo de mensagens através da rede;

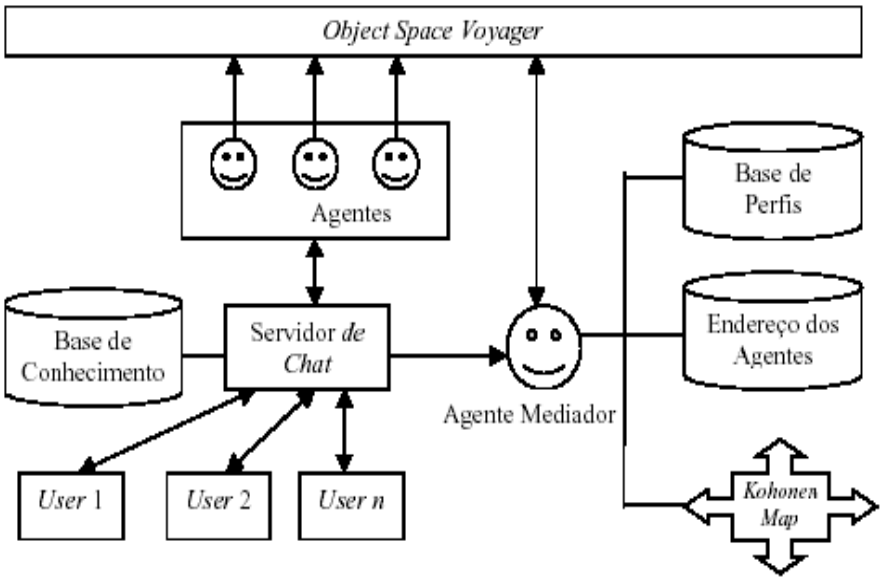

Figura 3.3: Arquitetura geral do sistema WHITE RABBIT.

- Uma interface dedicada ao usuário e ao administrador do sistema. Permite que o usuário envie e receba mensagens, consulte e modifique seu perfil e permite que o administrador observe e ajuste os parâmetros do processo de *clustering* e altere a base de conhecimento;
- Um agente Pessoal para cada usuário que realiza o serviço de análise e apresentação do serviço (ver Figura 3.2);
- Uma base de conhecimento, onde são alocadas diferentes palavras-chave sobre o domínio do conhecimento e seus *links*;
- Um agente Mediador/Assistente para dedicar-se ao processo de *clustering* e facilitar a comunicação entre os agentes.

O perfil do usuário contém todas as informações relevantes sobre o interesse do usuário, na escolha do domínio, o qual permitirá aos agentes encontrar similaridades e conseqüentemente, realizar agrupamentos coerentes.

O módulo de aprendizagem é responsável por modificar o perfil do usuário, fazendo-o mais acurado e mais realista. Este processo é feito em duas etapas:

1. O primeiro passo consiste na aquisição preliminar de informação sobre o usuário através de um questionário. Na primeira vez que o sistema é usado, o usuário é convidado a preenchê-lo dando ao sistema palavras-chave refletindo seus interesses (projetos, realizações e experiências).

2. O segundo passo é a análise da discussão, a qual consiste na extração de palavras-chave do domínio das mensagens enviadas pelo usuário e a atualização do perfil, incrementando o peso dos conceitos associados. O perfil do usuário é composto por um conjunto de informações ponderadas, isto é, cada um deles possui um peso associado ao contexto onde estão inseridos.

Isto permite um ajuste mais personalizado e detalhado do estudante. Dois aspectos demonstram a mobilidade e autonomia dos agentes. Primeiro, a análise da discussão dos usuários pelos agentes pessoais. Depois de ter analisado e atualizado os perfis dos clientes, o agente vai para uma outra máquina conectada à rede e encontra outros agentes.

A segunda importante evidência da mobilidade acontece quando o usuário pergunta para o segundo usuário quem é o membro do mesmo grupo, como determinado pelo agente mediador/assistente.

Neste momento a requisição do agente do usuário (A) usará a sua autonomia para mover-se e encontrar o agente associado ao cliente (B). Então o agente (A) terá possibilidade de questionar mais sobre o agente (B). Se o agente (B) aceitar as requisições, então o agente (B) dará ao agente (A) as informações pessoais (nome real, correio eletrônico, descrição do projeto, etc).

Estas duas situações demonstram a força alcançada pela mobilidade e autonomia dos agentes, a qual permite considerável redução do número de mensagens transmitidas na rede, reduzindo desta maneira, o risco de sobrecarga na rede e a falta de recursos, mesmo quando o número de agentes pessoais é alto (o número de agentes é igual ao número de usuários).

3.3 Um modelo computacional baseado na teoria de Vygotsky

O modelo computacional baseado na teoria de Vygotsky é um modelo desenvolvido pelo Departamento de Pós-Graduação em Informática na Educação da Universidade Federal do Rio Grande do Sul.

O objetivo deste modelo computacional é propor um ambiente que privilegie a colaboração como forma de interação social através do uso de linguagens, símbolos e sinais. Para suportar a aprendizagem colaborativa, é apresentada uma sociedade formada pelos seguintes agentes artificiais: Agentes ZPD (Zona de Desenvolvimento Proximal), Agente Mediador/assistente, Agente Semiótico e Agente Social (Andrade *et al.*, 2001).

Para suportar o modelo coletivo de aprendizagem à distância, foi utilizada a teoria formulada por Vygotsky, como base da fundamentação teórica da proposta. Um importante conceito nesta teoria é que as atividades mentais são baseadas em relacionamentos sociais entre o indivíduo e o ambiente, e este relacionamento é mediado por um sistema simbólico.

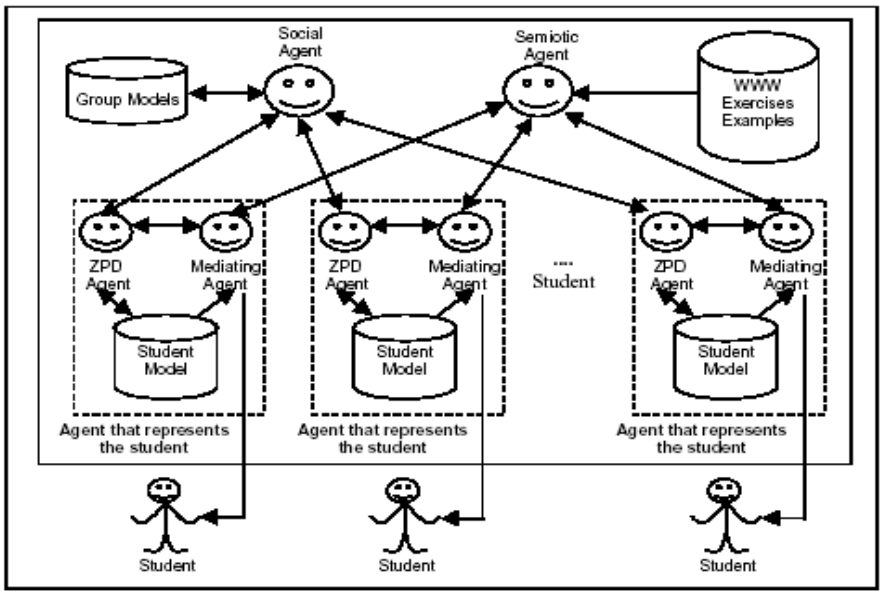

Figura 3.4: Arquitetura modelo computacional baseado na teoria de Vygotsky.

O modelo pedagógico desta pesquisa esta baseado numa forma colaborativa de aprendizagem que é alcançado através da interação social. As interações podem ser de vários tipos, considerando critérios como temporalidade, número de participantes, reciprocidade, hierarquias e até critérios baseados em comportamentos: personalidade, motivação, estado emocional, etc.

A Figura 3.4 extraída de Andrade *et al.* (2001) mostra que o sistema é composto por quatro tipos de agentes artificiais e um agente humano (estudantes):

- Agente ZPD: É responsável por observar o desenvolvimento real do estudante e propõe atividades que tornariam suas capacidades reais. É responsável por estimular aquelas funções que ainda não estão maturadas, mas estão no processo de desenvolvimento. Este agente pode ter funções como: variar o grau de controle das atividades, considerar tarefas gradualmente, ou modificar as formas de ajuda e/ou suporte. Para auxiliar no processo de aprendizagem, um ZPD deve ter um modelo do estudante, identificando suas habilidades e deficiências, o qual será construído pela observação das interações do estudante.

- Agente Mediador/Assistente: É responsável pela interface entre o sistema e o estudante. Além da função de mediar a interação entre o estudante com o ZPD, o Agente Media-

dor/Assistente deve ter acesso ao modelo do estudante ajudando a predição do comportamento do estudante o qual permite determinar as melhores ações para serem executadas para auxiliar o processo de aprendizagem do estudante.

- Agente Semiótico: Para o Agente Mediador/Assistente completar seu papel, é necessário a intervenção de uma estimulação externa (sinais) para o estudante. O Agente Semiótico auxilia a atividade cognitiva do estudante introduzindo esses elementos, por exemplo: figuras, sons, textos, números, etc.

- Agente Social: O Agente Social conhece todos os Agentes ZPD da sociedade e também tem conhecimento da presença de Agentes Sociais. Sua função é estabelecer a integração da sociedade e construir modelos de grupos de estudantes. Uma de suas atividades de Agente social é investigar a existência de estudantes que tenham o conhecimento necessário, crenças e tipos de personalidade que seriam melhor apropriados para uma cooperação entre os estudantes.

- Agentes Humanos: Os agentes humanos são vistos como agentes que estabelecem relacionamento social com cada um conforme suas características pessoais e personalidade. Então é importante que as características pessoais e a personalidade do estudante estejam contidas no modelo, estes traços afetarão a interação diretamente através dos papéis que cada estudante assumirá. Estes papéis determinam a afeição que acontecerá no grupo de estudantes.

3.4 EASE

EASE (*Evolutional Authoring Support Environment*) (Aroyo *et al.*, 2004) é uma ferramenta de autoria desenvolvida pelo Departamento de Matemática e Ciências da Computação da Universidade de Tecnologia de Eindhoven na Holanda em parceria com a Universidade de Osaka no Japão. Esta ferramenta suporta aprendizado em grupo de estudantes e os modelos são baseados nas ontologias.

O Ambiente EASE é evolutivo, possuindo características de raciocínio sobre o próprio comportamento e fazendo uso de regras, para manutenção nas diferentes partes do processo de autoria (dividido em camadas), como ilustra a Figura 3.5.

Três camadas estão presentes no ambiente:

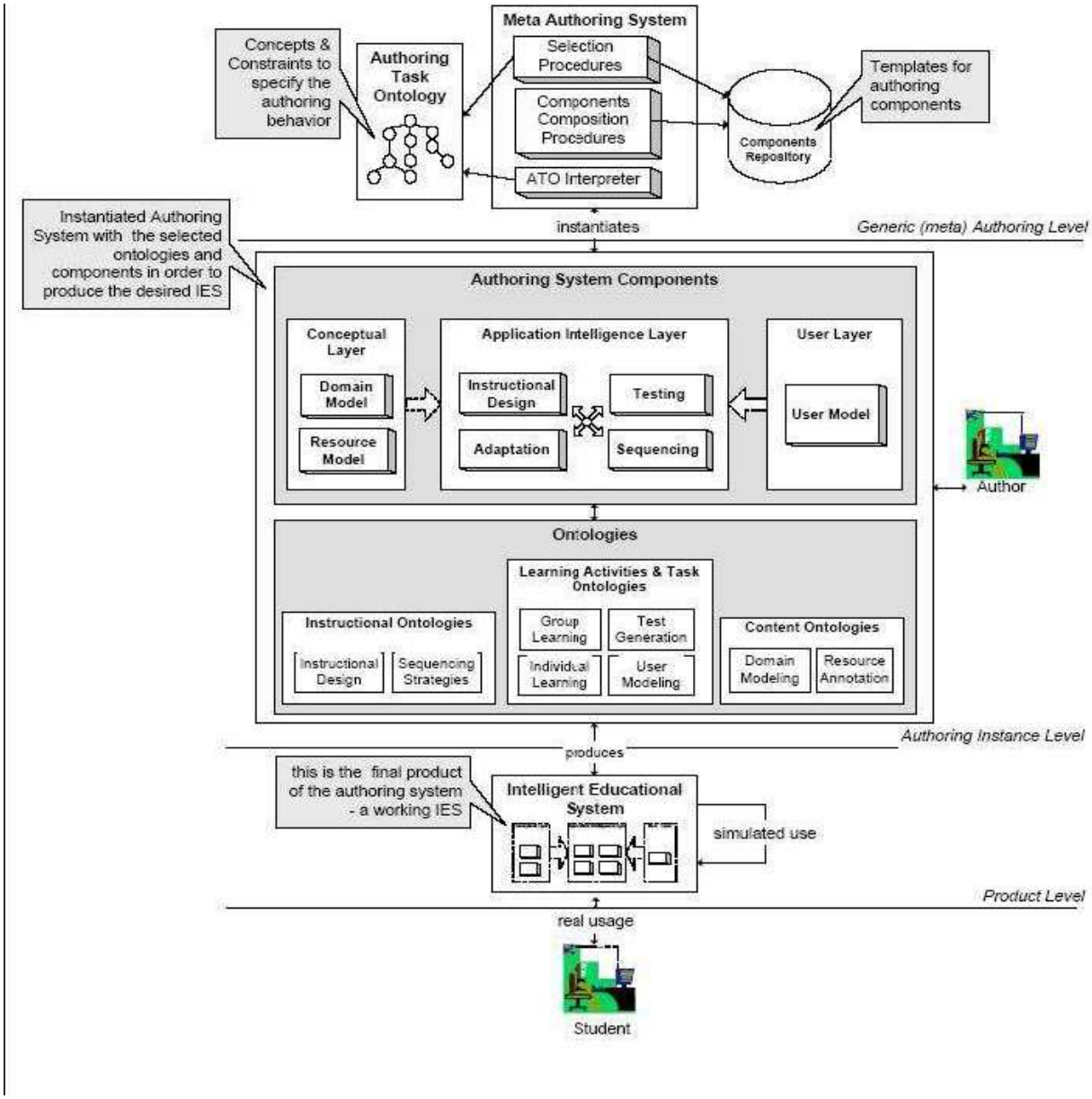

Figura 3.5: Arquitetura do EASE.

- Nível de Meta-autoria: equivale ao meta-conhecimento representado no ambiente educacional instrumental. Esta camada possui a estrutura conceitual, para que o processo de autoria seja iniciado. Este meta-nível foi construído baseado em uma ontologia de tarefas (Aroyo *et al.*, 2004);

- Nível de autoria de instância: nesta camada, o ambiente de autoria é iniciado instanciando um meta-schema com os conceitos, modelos e comportamentos;

- Nível de Produção: equivale ao ambiente educacional instrumental propriamente dito. Ou seja, após a configuração/definição do conteúdo, tem-se um ambiente educacional

instrumental em nível de produção.

Este ambiente é a uma ferramenta de autoria, dando suporte à disponibilização/utilização de regras para configuração da ferramenta de aprendizagem, além da utilização de ontologias.

3.5 COLE

COLE - *Collaborative Online Learning Environment* (Azevedo e Scalabrin, 2005) é um ambiente de suporte à aprendizagem colaborativa desenvolvido pela Pontifícia Universidade Católica do Paraná (PUC-PR) e pela Universidade Tecnológica Federal do Paraná (UTFPR).

O COLE é um sistema dirigido à tarefas para aprendizado em grupos. Auxiliam os estudantes a colaborarem enquanto estão resolvendo um problema. Propõem uma nova abordagem para aprendizado colaborativo que identifica as oportunidades de aprendizado baseados nas diferenças entre as soluções dos problemas e as soluções propostas pelos participantes.

O COLE demonstra como os agentes podem orientar a colaboração, no qual a solução de problemas estruturados existem usando um pouco fontes de conhecimentos básicos, e ilustra vários métodos para avaliar conhecimento.

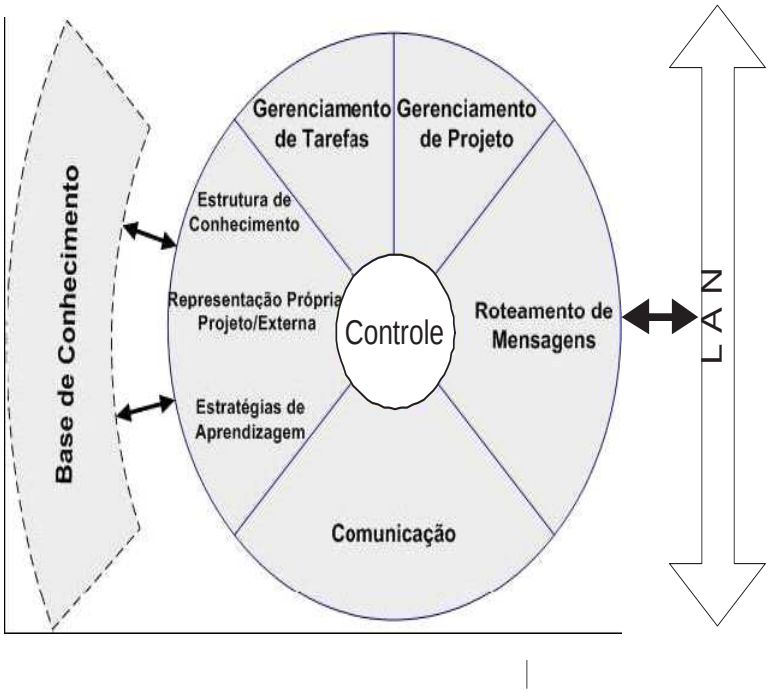

Figura 3.6: Arquitetura do COLE.

O ambiente COLE tem por objetivo aumentar as competências sociais durante o ciclo de vida da aprendizagem. Uma abordagem multiagente foi adotada para permitir ao Professor a

manipulação de uma grande quantidade de dados. Conceitos como interação humana, aprendizagem baseada em problemas e portfólios são recursos utilizados para atingir os objetivos do ambiente (Azevedo e Scalabrin, 2005).

O COLE é baseado numa metodologia para desenvolvimento de SMA que é dividida em oito passos, porém COLE utiliza apenas as fases de cinco até oito. Os oito passos de tal metodologia são:

1. Inserir pontos de vistas: consiste em adquirir informações relevantes, através de entrevistas;

2. Classificar atividades e recursos: organiza as informações adquiridas na entrevista;

3. Obter validação do grupo: grupo determina os serviços em potenciais que devem ser implementados;

4. Descrição dos serviços: a partir do momento em que diversos grupos definiram os serviços que cada um terá, o engenheiro do conhecimento gera uma tabela com os serviços finais que serão produzidos;

5. Escrever cenários: para cada serviço é definido um cenário, descrevendo como será o funcionamento de cada serviço;

6. Construir o modelo: de acordo com o cenário, são definidas telas para simulação do modelo;

7. Identificar Competências: competências são definidas para cada cenário, ou seja, informações como nome, descrição e parâmetros de entrada e saída são especificadas para cada competência, em cada cenário;

8. Sintetizar Competências: nesta fase, as competências são agrupadas, caso haja redundância entre grupos diferentes.

No COLE, os agentes inteligentes podem conter funcionalidades particulares, através do agente genérico, como ilustrado na Figura 3.6. Por exemplo os agentes da camada de Acesso a Dados que são estáticos, os agentes da camada de Negócio que podem estar localizados na máquina do cliente ou em qualquer outro local.

Algumas funcionalidades estão encapsuladas no agente genérico, como o mecanismos de processamento básico, estrutura, habilidades, garantia de comportamento externo e interno e capacidade de se adaptar em novos ambientes (Scalabrin *et al.*, 1996).

Tal ambiente é uma ferramenta colaborativa de autoria, possuindo uma arquitetura aberta para adição e exclusão de agentes no ambiente, porém, o sistema não provê facilidades ao autor, na configuração do processo de tutoria personalizada.

3.6 CHOCOLATO

O CHOCOLATO (***C**oncrete and **H**elpful **O**ntology-aware **CO**llaborative **L**earning **A**uthoring **To**ol*) é um sistema de autoria desenvolvido na Universidade de Osaka no Japão por Isotani e Mizoguchi (2008a). O principal objetivo é construir um modelo baseado em ontologias que auxilie na análise das interações entre estudantes e no planejamento apropriado de atividades para o grupo de estudantes oferecendo recomendações baseadas nas teorias de aprendizagem.

Os padrões de interação entre estudantes são obtidos em diversos tipos de processos de interação encontrados nas teorias de aprendizagem, por exemplo, aprendizagem cognitiva, tutoria, além de outras baseadas em Inaba *et al.* (2003).

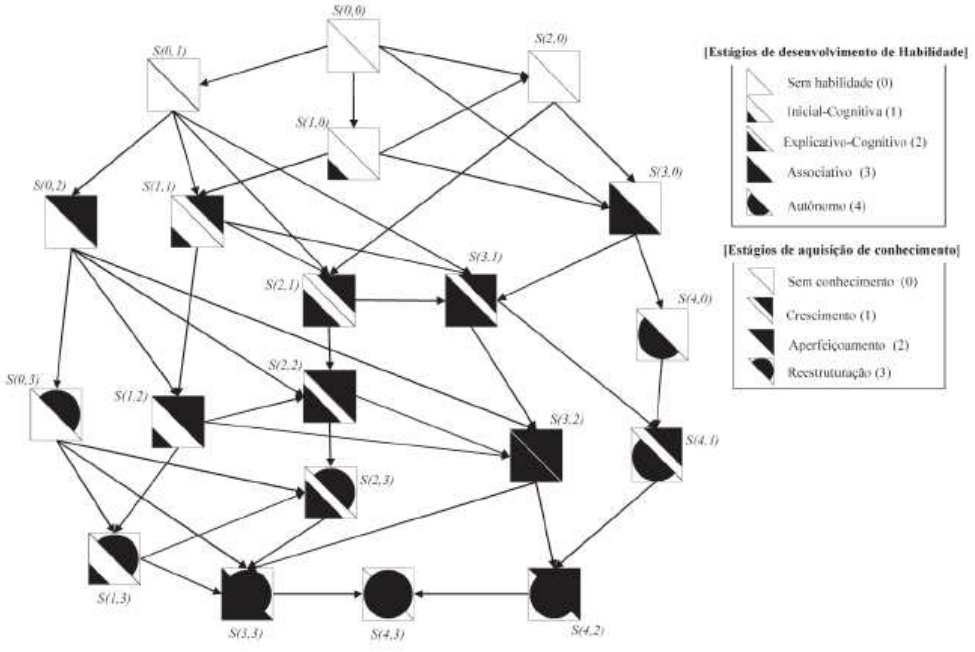

Figura 3.7: Modelo de crescimento do Estudante

Inspirado em Inaba *et al.* (2003) o modelo de crescimento do estudante (LGM) representa o processo de aquisição do conhecimento e o desenvolvimento de habilidades, de forma a esclarecer as relações entre as teorias de aprendizagem e seus respectivos benefícios educacionais.

O grafo representado na Figura 3.7 extraído de Isotani e Mizoguchi (2008b), mostra o modelo de crescimento do Estudante possui vinte nós (estados) que representam os níveis do desenvolvimento do estudante em um determinado período do aprendizado. Cada nó é representado por dois triângulos. O triângulo superior direito representa o estágio do conhecimento adquirido, enquanto o triângulo inferior-esquerdo representa o estágio da habilidade desenvolvida. As setas mostram possíveis transições entre os estados de acordo com as teorias. Para representar o grafo de forma simplificada cada estado é representado por uma tupla $s(x, y)$: x representa o estágio atual do desenvolvimento da habilidade e y representa o estágio atual da aquisição do conhecimento. Por exemplo, s(0,0) representa o estado onde o estudante não possui nenhuma habilidade ou conhecimento; s(0,1) representa o estado onde o estudante ainda não desenvolveu suas habilidades, mas possui conhecimentos em nível de crescimento.

No modelo para representar o processo de interação criado por Inaba *et al.* (2003) utilizam-se dois tipos de vocabulários: diálogo-rótulos e diálogo-tipos, sendo diálogo-rótulos para rotular cada diálogo do estudante e diálogo-tipos para representar o processo de interação de nível abstrato e para distinguir cada tipo de sessão colaborativa. Através da definição destes vocabulários Inaba definiu os padrões de interação baseados nas teorias de aprendizagem da figura 3.8.

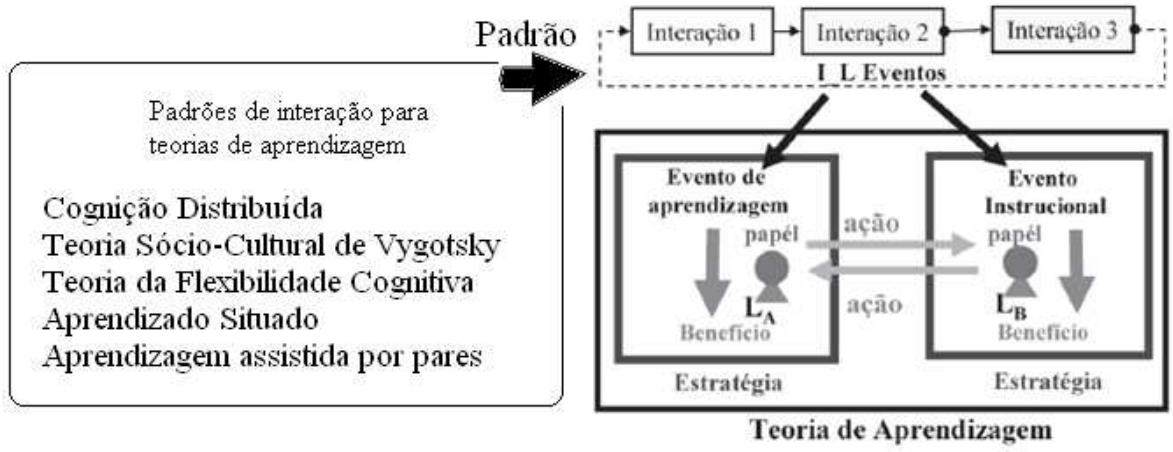

Figura 3.8: Representação do processo de interação. Extraída de Isotani e Mizoguchi (2007)

Em Isotani e Mizoguchi (2007), unificaram-se esses modelos através da extensão da CLO - Ontologia do Aprendizado Colaborativo, que tem como objetivo representar uma sessão cola-

borativa, e criou-se o GMPI (*Growth Model Improved by Interaction Patterns*).

Para facilitar a compreensão das interações contidas num padrão, cada interação foi dividida em dois eventos chamados de eventos $I - L$: um evento instrucional e um evento de aprendizagem. Todo evento instrucional possui uma relação de reciprocidade com eventos de aprendizagem, ilustrados na Figura 3.8 extraída de Isotani e Mizoguchi (2007).

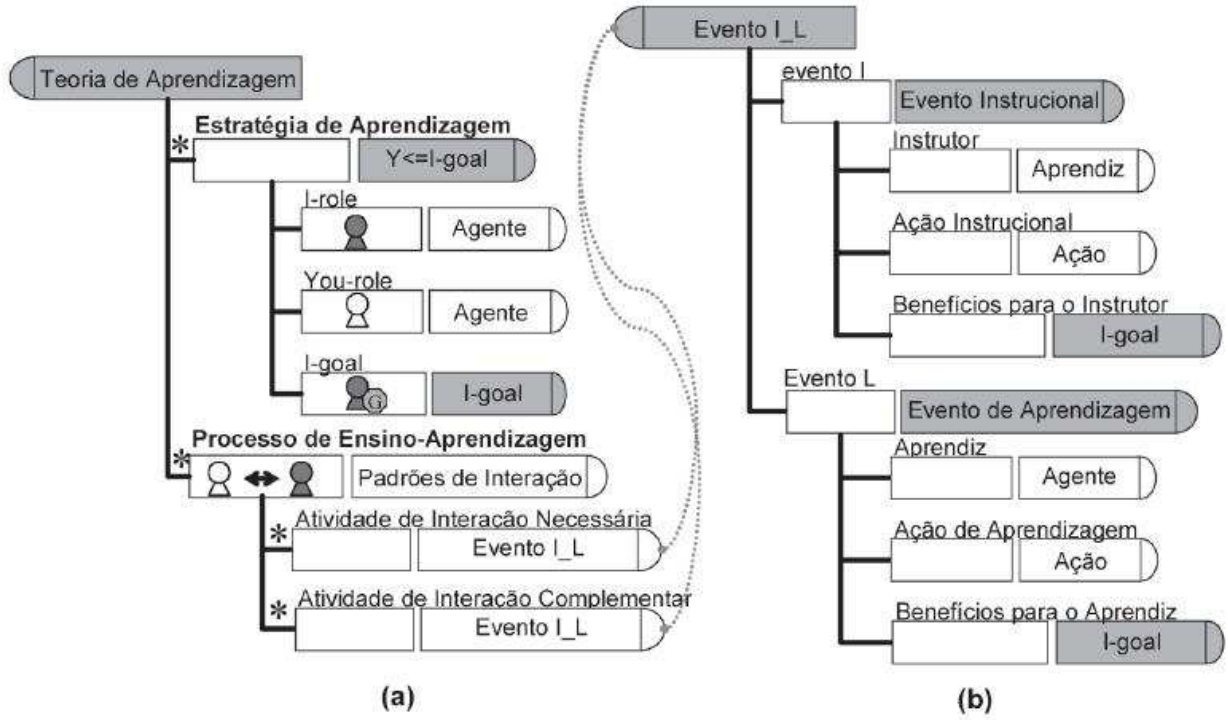

Figura 3.9: Estrutura ontológica para representar a teoria de aprendizagem.

A representação da teoria de aprendizagem na Figura 3.9 parte (a) se subdivide em: a estratégia de aprendizado e o processo de ensino-aprendizagem.

- A estratégia de aprendizagem especifica a forma $(Y <= I - goal)$ como o estudante (I-role) deve se relacionar com outra pessoa (You-role) para que possa atingir seus objetivos (I-goal). Por exemplo, na teoria de aprendizagem cognitiva um estudante se relaciona com outro estudante guiando-o durante a resolução de um problema. Neste caso, a estratégia $(Y <= I - goal)$ utilizada por este estudante é *aprender guiando* sendo seu papel (I-role) do *Professor*, o do outro estudante (You-role) o de *Estudante* e seus objetivos (I-goal) são adquirir habilidades cognitivas (e meta-cognitivas) em nível autônomo.

- O processo de ensino-aprendizagem especifica o padrão de interação de uma teoria de

aprendizagem representado pelas atividades (processos) de interação necessárias e desejadas entre duas pessoas (Instrutor e Estudante).

Dividi-se o processo de interação em dois eventos: evento instrucional e evento de aprendizagem (Figura 3.9 parte (b) Todo evento instrucional possui uma relação de reciprocidade com os eventos de aprendizagem. Ou seja, quando uma pessoal fala, a outra escuta; quando uma pergunta, a outra responde; e assim por diante. Cada evento possui uma ação correspondente e seus possíveis benefícios educacionais ao autor da ação. Estes benefícios educacionais estão relacionados com o contexto (teoria de aprendizagem) nos quais são executados os eventos e as estratégias de aprendizagem.

Com base nestes conceitos foi criado um subsistema do sistema CHOCOLATO denominado MARI para a visualização utilizando o GMPI. A Figura 3.10 extraída de Isotani e Mizoguchi (2008b) mostra a tela do MARI com a teoria de aprendizagem cognitiva, a estratégia e a ontologia.

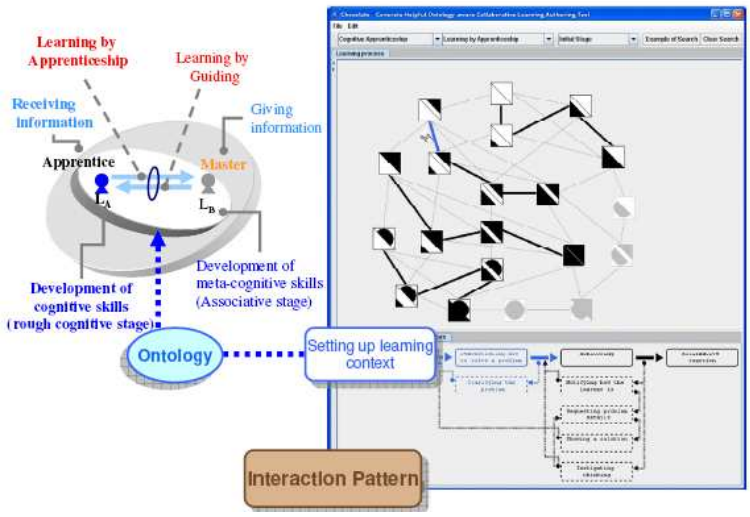

Figura 3.10: Estrutura ontológica para representar a teoria de aprendizagem do MARI

Neste trabalho a estrutura ontológica desenvolvida foi um dos conceitos-chave para esclarecer e representar uma teoria de aprendizagem no contexto de aprendizagem colaborativa.

Outros exemplos de trabalhos que utilizam técnicas de IA são os ASPIRE (Mitrovic *et al.*, 2006), CTAT (Aleven *et al.*, 2006), DEGREE (Barros e M. Verdejo, 2000) e NETClass (Labidi *et al.*, 2006).

3.7 Estudo comparativo sobre as ferramentas selecionadas

Nesta seção é apresentado os resultados do estudo comparativo dos ambientes identificados na seção anterior. Os resultados foram organizados em um quadro comparativo 4.1, onde são destacados o conjunto de aspectos considerados relevantes. A fim de se identificar o conjunto de requisitos utilizados pelos autores para a construção destes STI´s.

Algumas informações da comparação não puderam ser confirmadas, neste caso optou-se por deixar em aberto (?).

Os itens que foram considerados para compor o estudo comparativo são:

- Objetivo: visa identificar o objetivo do ambiente;
- Domínio (Conteúdo): visa identificar o conteúdo a ser trabalhado no sistema;
- Técnicas: visa identificar a técnica utilizada no STI;
- Tipo de comunicação: visa identificar se existe ou não um padrão utilizado para a comunicação entre os agentes;
- Linguagem e/ou Ferramenta de Implementação: visa identificar o tipo de ambiente utilizado para implementar os ambientes;
- Tipo de arquitetura SMA utilizada: visa identificar se os autores utilizaram ou explicitaram o tipo de arquitetura para a sociedade de agentes que está relacionada com a proposta do ambiente;
- Estratégias de Ensino utilizada: visa identificar o tipo de estratégias de ensino utilizada para auxiliar a promover a aprendizagem do conteúdo (domínio) nos usuários dos ambientes;
- Interface Gráfica: visa identificar se o sistema utiliza ou não interfaces gráficas e o grau de adaptabilidade do sistema que ela reflete (exibe) para o usuário;
- Ferramentas grupos: visa identificar quais as ferramentas auxiliares mais utilizadas para a comunicação entre aprendizes ou grupos.

Tabela 3.1: Comparativo dos STI's que utilizam SMA

Nome	WHITE RABBIT	COLE	MOD.BASEADO VIGOTSKY
Objetivo	Aumentar a cooperação entre um grupo de pessoas pela análise de suas conversações.	Aumentar as competências sociais durante o ciclo de vida da aprendizagem.	Propor um ambiente que privilegie a colaboração como forma de interação social através do uso de linguagens, símbolos e sinais.
Domínio	Independente de domínio	Independente de domínio	Independente de domínio
Técnica SMA: Atividades de agentes	Agente Pessoal: responsável por obter informações dos estudantes;gerenciar a interface gráfica, etc. Agente Mediador: facilita a comunicação entre agentes pessoais; gerencia processo de clustering para construção do modelo do estudante, etc.	São utilizados agentes cognitivos independentes que possuem um modelo de sua especialidade do mundo e deles mesmos. Possui vários agentes agentes de serviços especializados. Por exemplo, os agentes da camada de Negócio, da camada de Acesso a Dados e o agente Broker.	Agente ZPD: responsável por observar o desenvolvimento e propor atividades. Agente Mediador: é responsável pela relação do sistema e o estudante. Agente Social: estabelece a integração da sociedade e constrói modelos de grupos de estudantes. Agente Semiótico: auxilia na atividade cognitiva do estudante. Agente Humano: estabelece relacionamento social com cada agente conforme suas características pessoais
Tipo de comunicação entre os agentes	?	KQML/ FIPA-ACL Forma assíncrona	KQML
Tipo arquitetura SMA	Sociedade Heterogênea e aberta	Não hierárquica e Federativa	?
Ling.e Ferramenta	Java, JavaScript, HTML	Java	Java
Estratégias de Ensino			Múltiplas
Interface Gráfica	Gráfica interativa	Gráfica interativa	Gráfica interativa
Ferramentas grupos	Browser e chat	Ferr.para Mensagens instantâneas Intra-grupo.	?

Nome	CHOCOLATO	EASE	REDEEM
Objetivo	Modelo baseado em ontologias que auxilie a analise das interações entre estudantes de um grupo.	Personalizar do STI's para processos específicos e permitir um controle evolutivo.	Apresentar conteúdos nos STI's de forma diferenciada e com instruções adaptáveis para a classe.
Domínio	Independente de domínio	?	Independente de domínio
Técnica: Ontologias e Regras	Utiliza as Teorias de aprendizagem cognitivas, as estratégias e as ontologias. Cria padrões de interação obtidos em diversos tipos de processos de interação encontrados nas teorias de aprendizagem. Unifica-se com CLO/ontologias o modelo de crescimento do estudante e o mod.de processo de interações para criar novos padrões.	utilização de regras e ontologias para a construção do STI. Processo de autoria em camadas: Meta-autoria: possui a a estrutura conceitual para iniciar atividade (ontologia/tarefas). Nível de autoria de instância: Modelos e comportamentos Nível de Produção: STI, configuração e conteúdos	Utiliza as regras para classificar os estudantes em classes virtuais e cria estratégias diferenciadas para cada estudante.
Ling.e Ferramenta			?
Estratégias de Ensino	Múltiplas	Múltiplas	Múltiplas
Interface Gráfica	Gráfica		Gráfica interativa
Ferramentas grupos	?		não

3.8 Resultado da análise comparativa

A seguir apresenta-se a comentários dos itens discutidos na seção anterior.

Quanto ao objetivo, a maioria dos STIs, não apresentam uma proposta para um conteúdo específico, ou seja, suas propostas são independentes do domínio/conteúdo.

Nos STI´s tradicionais, os domínios modelados eram essencialmente domínios lógicos, bem estruturados e muito restritos, tais como: ensino de Matemática, Física, Linguagens de Programação, etc. Com a incorporação da tecnologia de Agentes no projeto e desenvolvimento de STI, verificou-se que o conteúdo do domínio passou a ser variado, ou seja, o conteúdo passou a ser modelado independentemente do domínio.

Quanto as técnicas utilizadas, os STI's analisados utilizam os agentes, as ontologias e as regras.

Nos STI's que utilizam as ontologias, por exemplo, o EASE ou CHOCOLATO, observa-se a utilização de regras para configuração do sistema educacional.

Nos STI's que utilizam os SMA, observa-se que a denominação dada para os agentes é variada.

Ambientes que possuem mais de um agente, praticamente exigem troca de informações. Neste sentido, torna-se claro a necessidade de uma Linguagem de Comunicação comum. A linguagem de comunicação entre os agentes, utilizada na grande maioria dos ambientes estudados foi a linguagem KQML.

A Linguagem de Implementação utilizada determina a portabilidade, performance, bem como recursos audiovisuais que podem ser utilizados no sistema. Dentre os STI´s citados, os ambientes portáveis são White Rabit, Modelo Computacional que foram desenvolvidos em Java.

Segundo Oliveira (1996), existem três critérios para classificação de uma sociedade de agentes:

1. Quanto ao tipo de agentes:

   - Sociedades homogêneas: os agentes são todos do mesmo tipo, ou seja, possuem arquiteturas idênticas;

   - Sociedades heterogêneas: existem agentes de diferentes tipos na sociedade.

2. Quanto à migração de agentes:

   - Sociedades fechadas: há um número fixo e único de agentes na sociedade;

- Sociedades abertas: o número de agentes nesta sociedade pode variar, pois podem entrar novos agentes ou sair agentes da sociedade.

3. Quanto à presença de regras de comportamento:

   - Sociedades baseadas em leis: existem regras que determinam o comportamento dos agentes;

   - Sociedades sem lei: quando não há regras para reger os agentes da sociedade.

A maioria dos trabalhos analisados apresentam características de uma sociedade heterogênea e possuem regras de comportamento para seus agentes.

Arquitetura de agentes refere-se ao modo de organização dos agentes dentro de um sistema e como estão estruturados seus relacionamentos e interações. Assim como existem diversas arquiteturas de software, o mesmo ocorre com relação as arquiteturas de agentes, as quais possuem certas características que permitem a avaliação de sua qualidade e eficácia.

Para exemplificar, tomemos como exemplo a arquitetura de Sistema Federado, ou Federativo. Conforme Thibodeau *et al.* (2000), a interação dos agentes é assistida por um programa especial denominado Facilitador, o qual oferece um conjunto de serviços de coordenação. O Facilitador atua como um mediador, roteando mensagens (solicitações e respostas) de acordo com seu conhecimento interno.

Uma outra arquitetura bastante conhecida e utilizada é a arquitetura *Blackboard* (Quadro negro). *Blackboard* é uma estrutura única e compartilhada entre vários agentes, onde as informações serão escritas e lidas durante o desenvolvimento das tarefas. Como não há comunicação direta entre os agentes, eles devem consultar a estrutura de tempos em tempos para verificar se existe alguma informação destinada a eles.

As Estratégias de Ensino definem formas de apresentar o material instrucional ao estudante Giraffa (1995). Segundo Giraffa (1999), a seleção do conjunto de estratégias de ensino que será utilizada no STI é um aspecto muito importante para garantir a qualidade pedagógica do ambiente de ensino-aprendizagem. A seleção de uma estratégia depende de diversos fatores, tais como: o nível de conhecimento do estudante, o domínio, a motivação e as características afetivas do mesmo e outras.

Os agentes usam as informações do modelo do estudante, para definir qual o melhor conteúdo e/ou exercício a ser exibido. Nos ambientes analisados são utilizadas diversas maneiras para construir o modelo do estudante. Listamos a seguir algumas delas:

- incluir um reconhecimento de padrões aplicados ao histórico das respostas fornecidas por ele;
- comparar a conduta do estudante com a de um especialista e verificar os pontos em comum; - acrescentar as preferências do estudante;
- incluir seus objetivos particulares;
- observar as coisas que o estudante sempre costuma esquecer quando interage com o tutor;

Para qualquer sistema interativo, a Interface Gráfica é de suma importância. Em relação à esse aspecto, todos os ambientes analisados possuem interface gráfica interativa. No desenvolvimento de STI a preocupação com este aspecto não é diferente.

Pode-se citar algumas funções que a Interface do sistema deve propiciar:

- é necessário evitar que o estudante se entedie, ou seja, é preciso riqueza de recursos na apresentação do material instrucional;
- é desejável que haja facilidade para troca da iniciativa do diálogo: o estudante deve poder intervir facilmente no discurso do tutor, e vice-versa;
- o tempo de resposta deve ser rápido;
- a monitoração deve ser realizada o máximo possível em background, para onerar o estudante com questionários excessivos, mas respeitando também a barreira do tempo de resposta; Devido a isto, em muitos dos sistemas existem agentes executores para fazer estas tarefas.

Em relação às ferramentas para grupos que foram incorporadas nos sistemas para auxiliar no processo de ensino-aprendizagem, podemos dizer que as ferramentas tais como: browser e chat são essenciais por tratar-se de ferramentas que dão suporte à comunicação entre estudantes.

3.9 Conclusão

Os STI´s representam uma interessante ferramenta para ambientes de ensino aprendizagem computadorizados. Entretanto, os maiores problemas associados a estes tipos de sistema são: seu alto custo financeiro; elevado tempo de desenvolvimento; complexidade de modelagem; e interação em alto grau entre os membros da equipe interdisciplinar.

O STI é uma aplicação como outra qualquer sob o ponto de vista de Engenharia de Software e, como tal, refletem a questão crucial da área de SMA onde a não padronização da modelagem ou inexistência de metodologias para suporte ao aprendizado em grupos.

Um aspecto importante identificado é a ausência de um modelo para gerenciamento de grupos de estudante que suporte as atividades intra e inter-grupos, sem sobrecarregar o Professor/Autor com a tarefa de especificar as atividades de grupos e os modelos que devem ser considerados e atualizados durante a interação.

A utilização dos agentes artificiais é interessante para operacionalizar a atividade de grupos de estudantes, porém falta a definição de protocolos de interação para monitorar as interações intragrupo e intergrupos de estudantes.

No proximo capítulo, seção 4.4, após a apresentação do modelo proposto para suporte ao aprendizado em STIs é realizada uma comparação do modelo desta tese com os STIs apresentados no presente capítulo.

Capítulo 4

Um Modelo para Gerenciamento de Grupos em STIs

Este capítulo descreve o modelo proposto na presente tese para o gerenciamento de grupos em Sistemas Tutores Inteligentes que está relacionado ao aprendizado colaborativo.

Na Seção 4.1 apresenta-se: a estrutura para o gerenciamento de grupos, um sistema multiagente denominado G-Grupo, que utiliza os Cenários pré-definidos e a sociedade de agentes S2AG, além da integração destes com o modelo MATHEMA e a Ferramenta de Autoria para Sistemas Tutores - FAST.

Na Seção 4.2 apresenta-se o nível de especificação, que adota as ontologias para representar os módulos do domínio, do estudante e do grupo, e redes de Petri para especificar os protocolos de interação.

Na Seção 4.3, apresenta-se o nível de execução que torna operacional o aprendizado em grupo com a arquitetura multiagente. Tal arquitetura proporciona a colaboração entre os estudantes do mesmo grupo e estudantes de grupos diferentes.

4.1 Descrição do Modelo para Gerenciamento de Grupos

O modelo de gerenciamento de grupos proposto na presente tese permite a especificação e execução de atividades complexas de grupo, sem sobrecarregar o Professor/Autor com a tarefa de especificar as atividades de grupos e os modelos que devem ser considerados e atualizados durante a interação.

Foi criada uma biblioteca de cenários pré-definidos para atividades em grupos de estudantes.

Lembramos que o termo cenário indica uma estratégia pedagógica. Um cenário consiste num conjunto de definições operacionais para a atividade em grupos, sendo definido pelo desenvolvedor e armazenado numa biblioteca. Estes cenários são especificados utilizando ontologias e Redes de Petri Objetos. Uma das vantagens da biblioteca de cenários são as unidades de gestão e conteúdos que podem ser reutilizadas para a criação de novos cenários.

Para operacionalizar a atividade de grupos de estudantes, foi criada a Sociedade de Agentes-Aprendizes e Gerenciados de Grupos (S2AG). Com a S2AG organizada de forma hierárquica é possível monitorar as interações intragrupo e intergrupos.

Na S2AG, todos os estudantes que fazem parte do sistema são representados como agentes-aprendizes. Os grupos são compostos por agentes-aprendizes e monitorados pelos agentes gerenciadores (Coordenador e Supervisor de grupo). O agente Supervisor é responsável pelo monitoramento das atividades intragrupo e o agente Coordenador, devido à visão geral de todos os grupos, atua nas atividades intergrupos.

O modelo proposto integra a biblioteca de cenários, e S2AG com a Sociedade de Agentes Tutores Artificiais (SATA) do modelo MATHEMA (ver Seção 2.3), e a Ferramenta de Autoria para Sistemas Tutores (Seção 2.5) para criar os STI's. A Figura 4.1 ilustra a interação entre os módulos existentes na ferramenta FAST e no modelo MATHEMA e os módulos novos acrescentados neste trabalho.

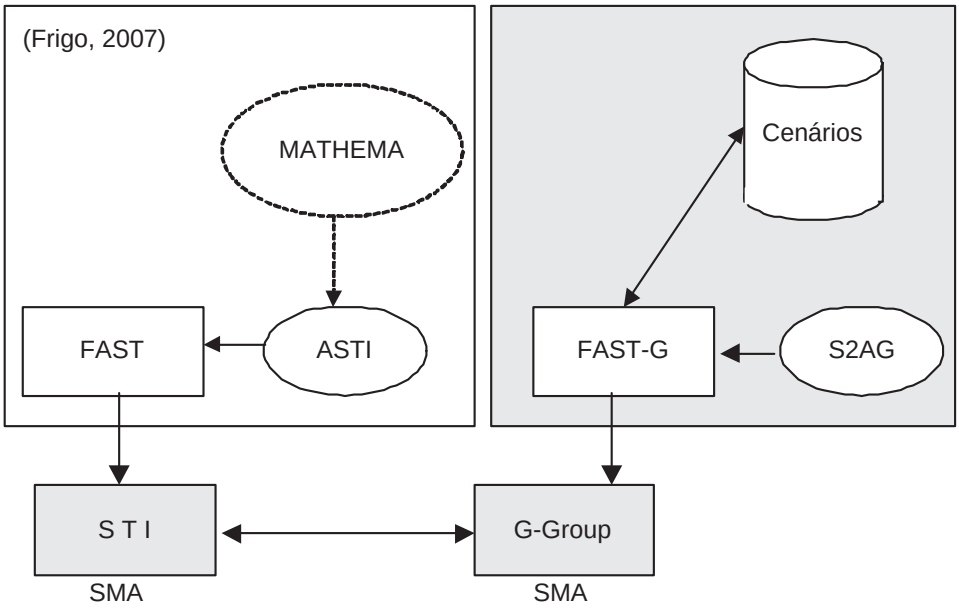

Figura 4.1: Arquitetura para Gerenciamento de Grupos.

Nos módulos já existentes na arquitetura, estão (a) SATA do modelo Mathema que é uma coleção de agentes que podem cooperar entre si a fim de promover a aprendizagem de um dado estudante em atividade de resolução de problema. Cada um desses agentes é especializado em um subdomínio relacionado a um dado domínio de conhecimento. (b) Arcabouço Sistemas Tutores Inteligentes (ASTI). (c) Ferramenta de Autoria para Sistemas Tutores (FAST) que é a ferramenta de Autoria para concepção de Sistemas Tutores Inteligentes para aprendizado individualizado.

Os novos módulos acrescentados são:

- G-GRUPO: SMA construído para suportar a atividade de grupo;
- FAST-G: ferramenta de Autoria para Sistemas Tutores Inteligentes estendida para suportar o trabalho com grupos de estudantes;
- S2AG: é uma sociedade heterogênea composta por todos os agentes-aprendizes que fazem parte do sistema e os agentes gerenciadores de grupos (coordenador de grupos e supervisores de grupos).

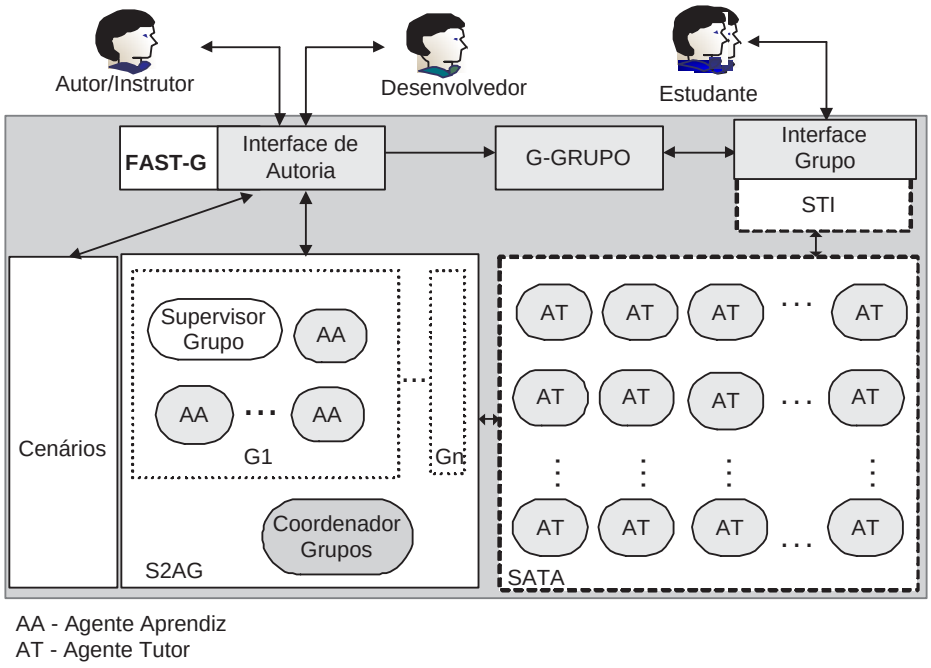

Figura 4.2: Interação entre agentes artificiais e humanos.

A Figura 4.2 ilustra a interação entre as sociedades de agentes artificiais e os agentes humanos (externos) que são descritos a seguir:

- **Desenvolvedor:** pessoa responsável por especificar a biblioteca de cenários, onde as atividades de grupo estão localizadas.

- **Autor:** pessoa responsável por escolher e instanciar um cenário adequado para construir uma atividade de grupo.

- **Instrutor:** pessoa responsável por supervisionar a atividade do grupo, determinando o início e o fim da atividade, e verificando o retorno do estudante.

- **Estudante:** representa o estudante humano, sua interação com o sistema é efetuada através do agente-aprendiz. Este agente representa o elo de ligação entre o estudante humano, a SATA e a S2AG.

Para cada estudante humano cadastrado no sistema existe um agente-aprendiz artificial que o representa. O agente-aprendiz tem acesso ao modelo do estudante que possui, entre outras coisas, as preferências do estudante. A coordenação de grupos é responsável pelo gerenciamento dos grupos.

Cada atividade de grupo é necessariamente baseada num sistema tutor inteligente e apresenta dois níveis, o nível de especificação e o nível de execução, que são descritos nas seções 4.2 e 4.3, respectivamente.

4.2 Nível de Especificação

O MATHEMA fornece um esquema de modelagem para o domínio que é dividido em duas visões, visão externa e visão interna (ver Seção 2.3.1).

Este fato permite a construção de interações de grupo que, embora não sejam dependentes do domínio, podem explorar a estrutura do domínio, indo além do simples suporte de comunicação entre membros do grupo e o instrutor. Isto é possível porque essas interações do grupo podem usar as mesmas atividades de resolução de problemas já definidas no contexto do STI. Por exemplo, o estudante está participando de uma atividade de grupos e utiliza os exemplos ou explanações, do STI para aprendizado individual, para esclarecer uma dúvida.

Uma outra vantagem é que o modelo do estudante, usado no gerenciamento da interação do grupo, pode ser definido como uma extensão do modelo do estudante no STI subjacente, de forma que o gestor de interação do grupo possa explorar as preferências e os resultados ante-

riores obtidos por cada estudante no contexto do aprendizado individual durante sua interação com o STI.

Os modelos de estudante, grupo e módulo domínio são representados usando ontologias e foram inspirados nos trabalhos de Chen e Mizoguchi (2004) e Mizoguchi e Bourdeau (2000). Essas ontologias são descritas nas próximas subseções.

4.2.1 Representação do Módulo de Domínio

O módulo do Domínio contém definições da visão interna do modelo conceitual MATHEMA. Uma disciplina desenvolvida usando o modelo proposto é representado como uma instância do módulo do domínio e contém todas as informações fornecidas pelo autor. Essas informações são de dois tipos: propriedades e conteúdos. Exemplos de propriedades são as relações de pré-requisitos, o nível de detalhe ou dificuldade, etc. Os conteúdos apresentados aos estudantes são páginas HTML.

A ontologia descrita em Frigo (2007) inclui conceitos para definição de grafos de pré-requisitos, que podem ser utilizados para definir as relações entre as unidades pedagógicas ou problemas, e conceitos para representar tipos específicos de unidades de interação. A Figura 4.3 apresenta a ontologia com as classes *Curriculum, Pedagogical-Unit, Problem* e *Interaction-Unit*, *Prerequisite* e *Node*. A Classe *Interaction-Unit* especifica as unidades de interações. As classes *Explanation*, *Example* e *Exercise* representam tipos específicos de unidades de Interações, outros tipos podem ser acrescentados conforme a necessidade.

O conhecimento do domínio utilizado para as atividades de grupos é declarativo e teórico (Seção 2.2.1), onde são determinados os objetos a serem incluídos no domínio e suas relações.

O conceito de Problema e seus sub-conceitos são reutilizados na definição do conceito de Unidade de Conteúdo do modelo de gerenciamento de grupos (Figura 4.6).

4.2.2 Representação do Modelo do Estudante

O modelo do estudante, proposto em Frigo (2007), contém definições de todos os conceitos necessários para caracterizar um estudante e seu histórico de interações com o sistema.

O modelo do estudante é baseado na abordagem do estereótipo (seção 2.2.1) que classifica os estudantes de acordo com o nível de seu conhecimento. As informações são obtidas através

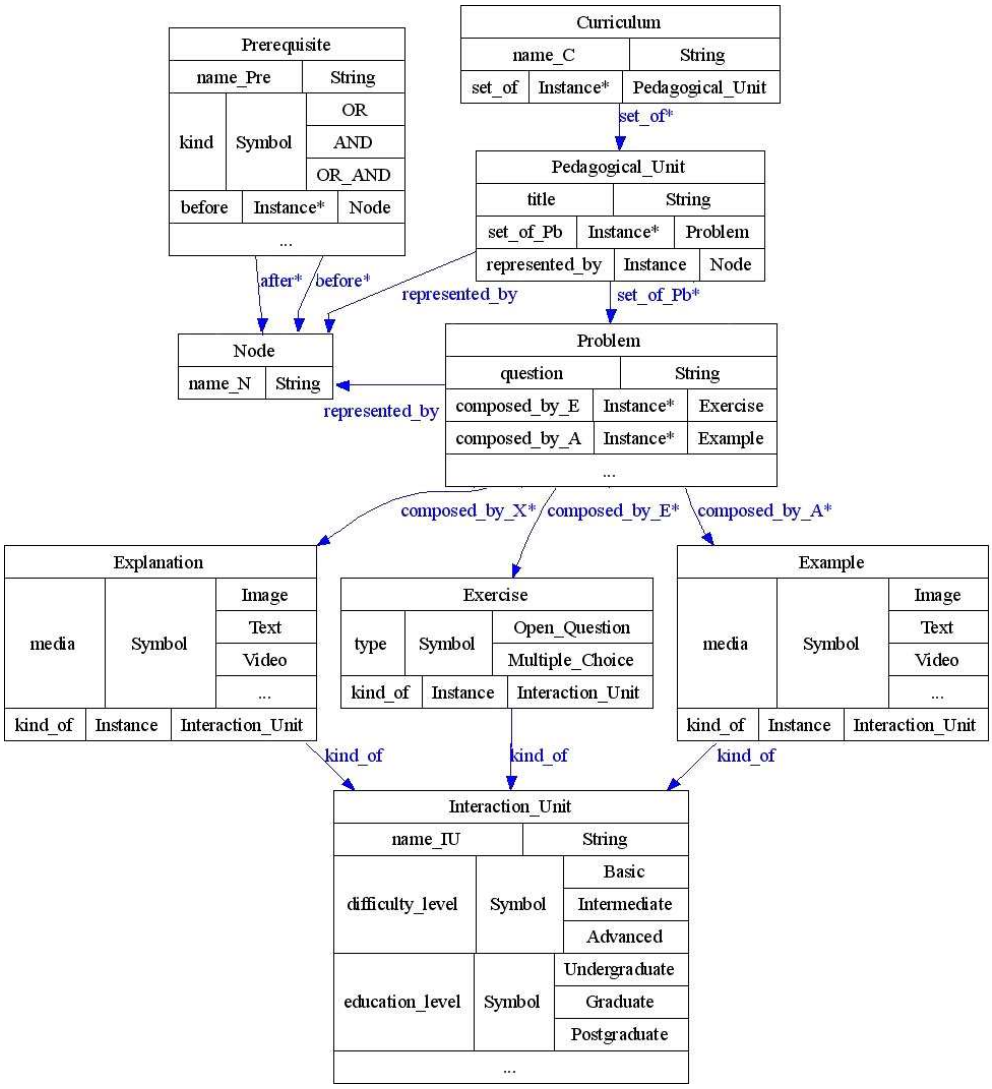

Figura 4.3: Módulo do Domínio do STI.

de um teste preliminar e da atualização, que é feita durante as novas interações, permitindo uma re-classificação caso esta se faça necessária.

Seus conteúdos incluem informação estática, como nível educacional, baseado num teste preliminar, preferências e também informações dinâmicas, que consistem de descrições das atividades dos estudantes durante suas sessões de interação com o sistema.

O modelo do Estudante, ilustrado na Figura 4.4, foi estendido no presente trabalho de tese para incluir a informação necessária à interação de grupo. Esta também inclui informações estáticas, como preferências, e informações dinâmicas, como o registro do desempenho do estudante durante as atividades de grupo.

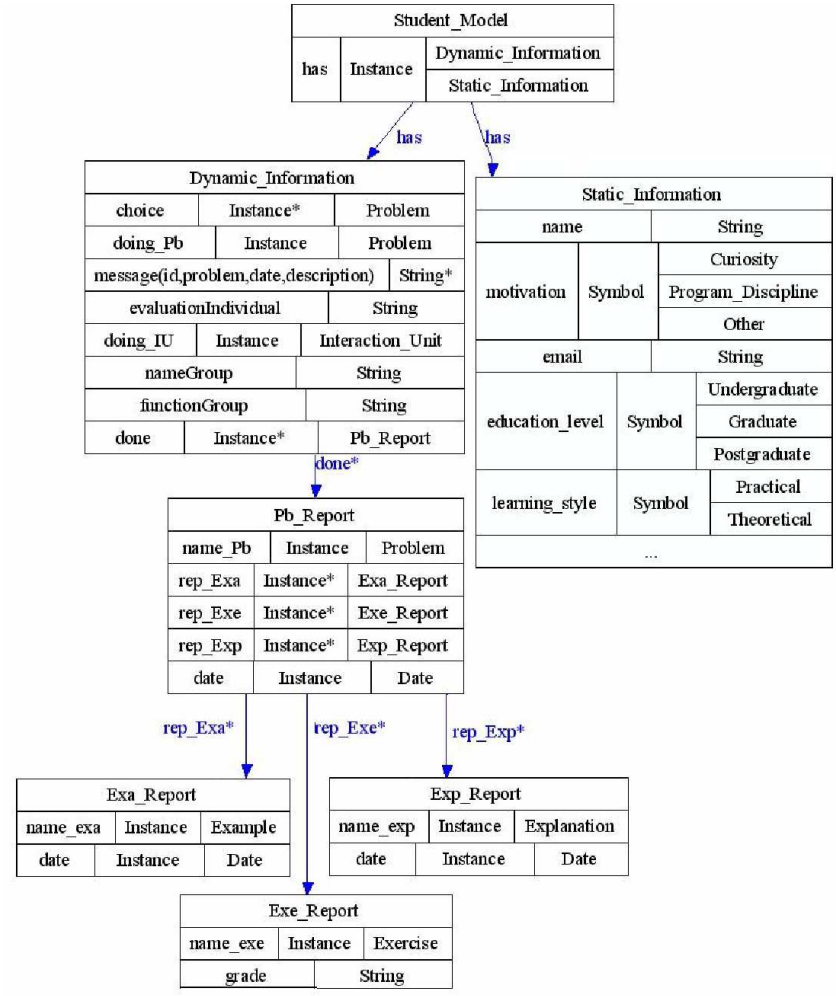

Figura 4.4: Modelo do Estudante.

4.2.3 Representação do Modelo do Grupo

Os conceitos envolvidos no nível de especificação da atividade em grupos incluem o Grupo, o Cenário, a Unidade de Atividade, a Atividade do Grupo e o Protocolo de Interação.

O conceito de **Grupo** é um conjunto de estudantes já inscritos no STI. As informações referentes o grupo de estudantes, por exemplo o número de participantes e papéis dos participantes, são encontradas na classe intragrupo (ver figura 4.5) enquanto as informações individuais dos estudantes são armazenadas no Modelo do Estudante (ver Figura 4.4).

O conceito de **Cenário** consiste de uma definição operacional da atividade do grupo. Cenários são definidos pelo Desenvolvedor e armazenados numa biblioteca de cenários. Eles são construídos para instanciar as atividades em grupos de estudantes que usam as unidades de atividades pré-definidas.

O conceito de **Unidade de Atividade** (*Activity Unit*) define as diferentes atividades que

ocorrem num dado cenário. Existem dois tipos de unidades de atividade, conforme a ontologia apresentada na Figura 4.6.

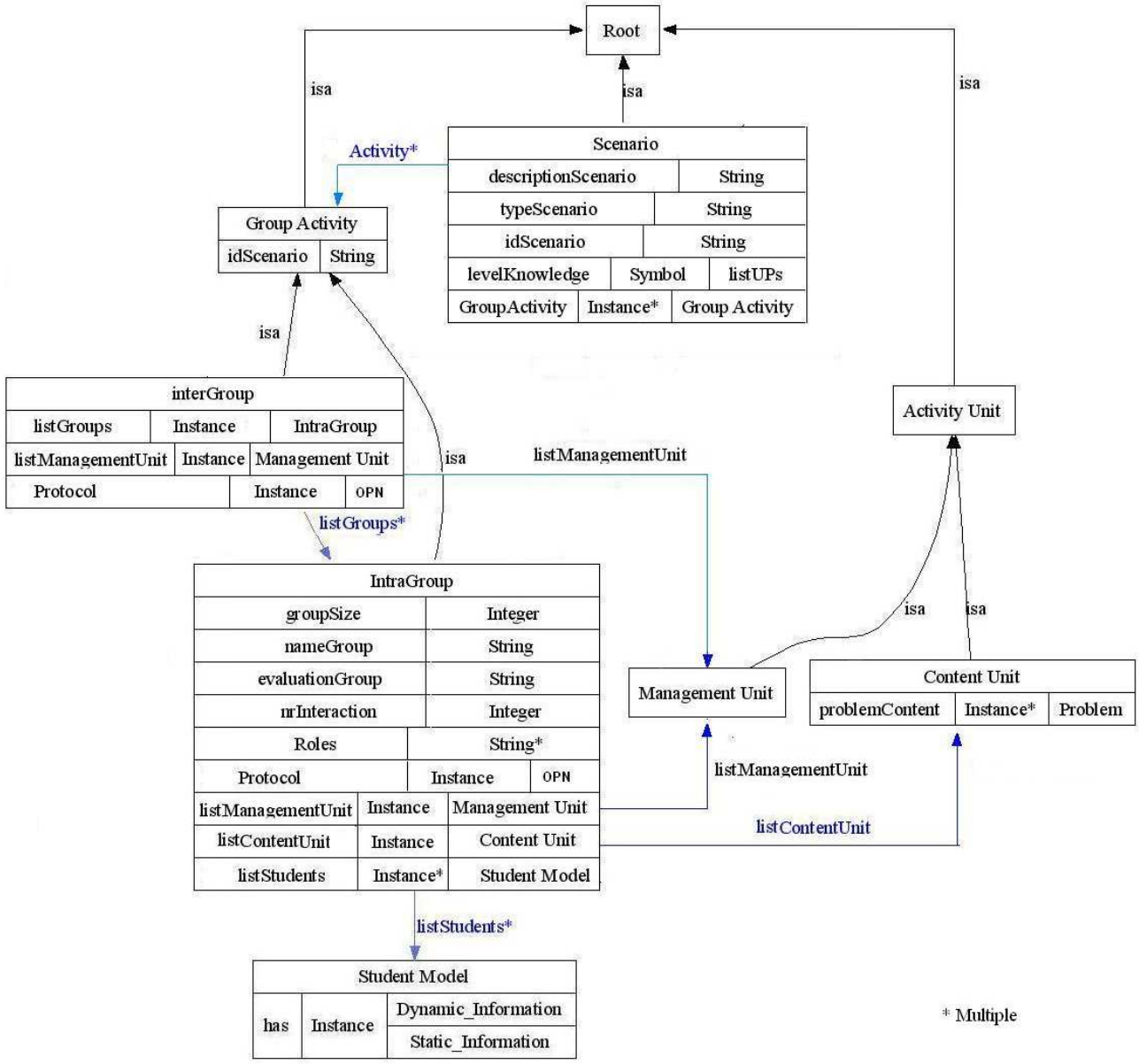

Figura 4.5: Ontologia das Atividades de Grupos.

- **Unidades de gestão** (*Management Unit*): usadas para definir as atividades típicas das interações de grupos, como formação do grupo, distribuição de problemas, negociação, competição, integração das soluções, instrução para membro do grupo, etc.

- **Unidades de conteúdo** (*Content Unit*): usadas para definir as tarefas de resolução de problemas associadas a um dado cenário. Essas tarefas são definidas usando a especificação de Problema (que inclui Unidades de Interações) do módulo de domínio do STI (ver

Seção 4.2.1). As definições de conteúdo são fornecidas pelo Autor, usando a interface da ferramenta de autoria FAST.

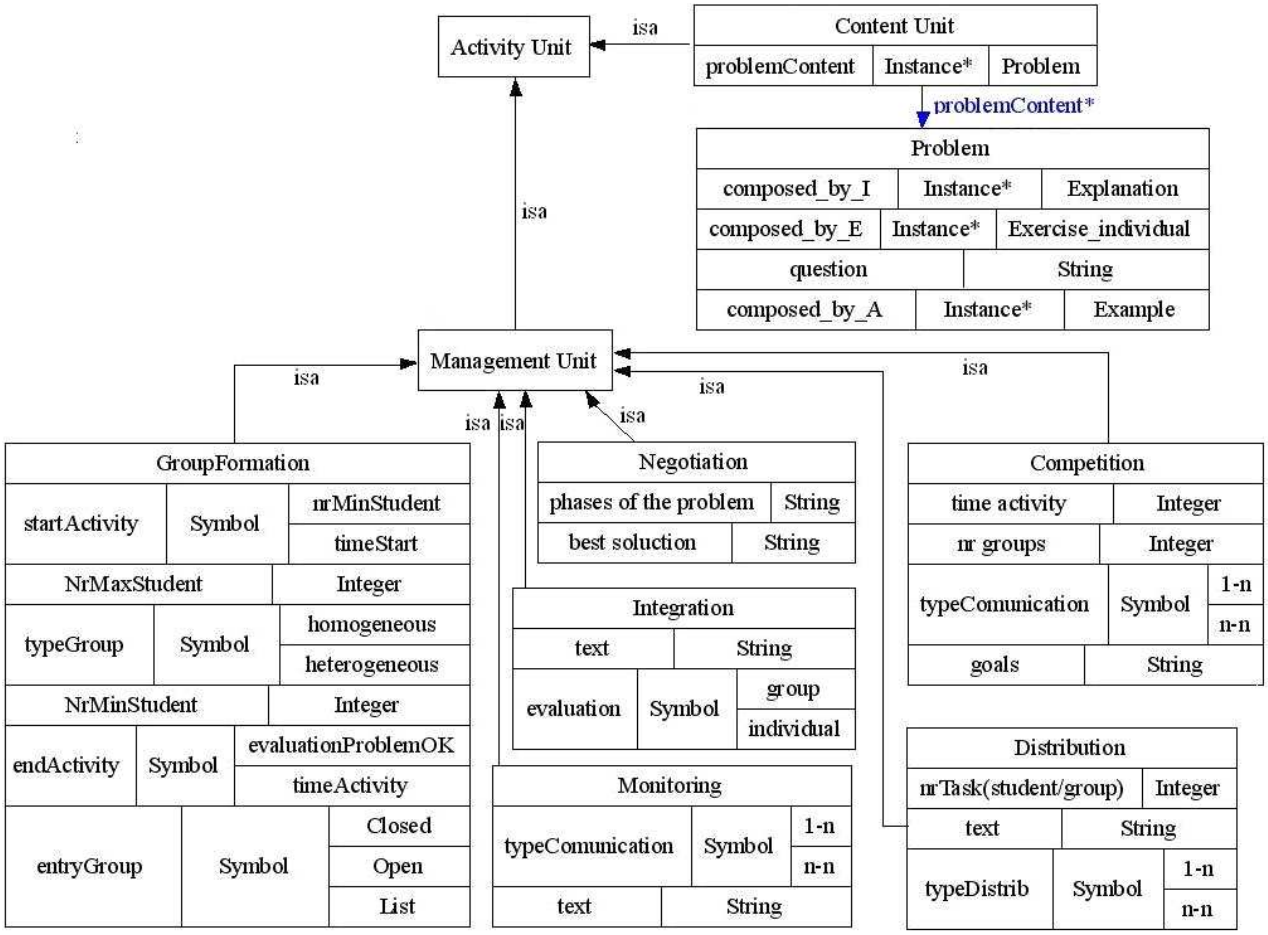

Figura 4.6: Ontologia Unidades da Atividade.

O conceito de **Atividade do grupo** (*Group Activity*) é o controle das atividades nos grupos. O tipo de atividade do grupo é denominada intergrupos (*interGroup*) ou intragrupo (*intraGroup*). A atividade do grupo inclui as unidades de atividades e o protocolo de interação.

Intergrupos:

Nas atividades intergrupos as interações ocorrem entre vários grupos de estudantes, por exemplo, uma competição, enquanto na atividade intragrupo as interações são restritas a um grupo de estudantes.

A classe intergrupos da figura 4.5 contém as listas dos grupos (*listGroups*) participantes da

atividade e a classe Intragrupo contém as informações resultantes das interações, por exemplo, a lista de participantes do grupo (*listStudents*), avaliação do grupo e outros.

O Desenvolvedor é responsável por criar os cenários para as atividades em grupos de estudantes enquanto o professor/Autor é responsável por incluir o conteúdo a ser ensinado e os parâmetros para as atividades em grupos.

A atividade intergrupos contém uma lista de unidades de gestão (*Management Unit*). Estas unidades de gestão são utilizadas para definir as atividades típicas, como formar os grupos, distribuir as tarefas e monitorar os estudantes. Cada uma das unidades de gestão da figura 4.7 parte (a) pode ser refinada como no grafo da figura 4.7 parte (b), o compilador combina o grafo da figura 4.7.a com o grafo da Figura 4.7 parte (b) formando uma única RPO para o agente Coordenador que criará os grupos.

Segundo nível da RPO intergrupos, na Figura 4.7 parte (b), é o detalhamento da formação do grupo.

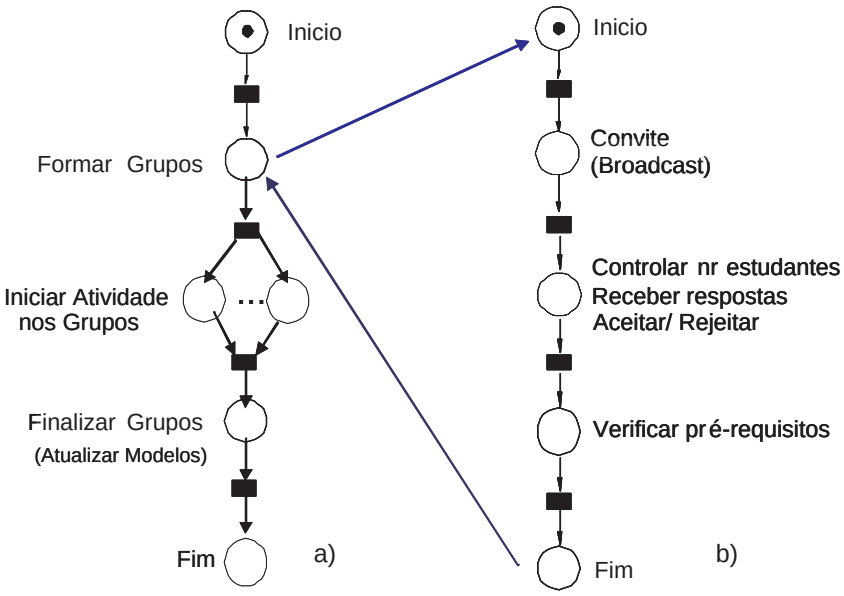

Figura 4.7: RPO Intergrupos organizada de forma hierárquica

Intragrupo:

A RPO intragrupo especifica o percurso para um grupo de estudantes realizar uma atividade. Esta rede é composta de unidades de gestão (*Management Unit*), definidas pelo Desenvolvedor, e por unidades de conteúdo (*Content Unit*), definidas pelo Autor/Professor. As unidades de

gestão e as unidades de conteúdos são representadas no modelo de grupo (ver figura 4.6). A cada unidade de conteúdo está associado um problema representado no modelo do domínio (ver figura 4.3).

Na Figura 4.8 ilustra a RPO-Inter que dispara o protocolo RPO-Intra, está transição ocorre quando, por exemplo, inicia a atividade intra-grupo e é finalizada quando conclui a atividade.

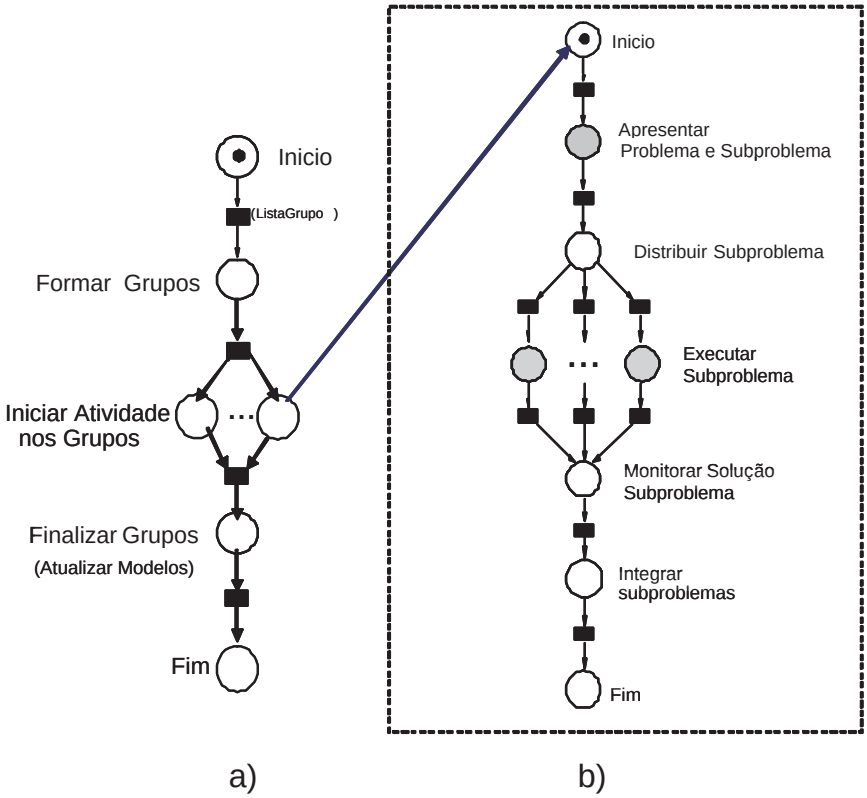

Figura 4.8: RPO-Inter dispara o protocolo RPO-Intra.

As interações entre estudantes e tutor durante a resolução de problemas numa atividade de grupo, representadas nas unidades de conteúdo, são definidas pelo Autor/Professor utilizando a interface da FAST. Todas as informações referentes aos estudantes participantes do grupo são obtidas e atualizadas no modelo do estudante (*Student Model*) (ver figura 4.4).

Nas interações do estudante com o tutor para solução de problemas é utilizado um conjunto de unidades de interação (*Interaction Unit*), que podem ser de diferentes tipos, por exemplo, explicações (*Explanation*) , exemplos (*Example*) e exercícios (*Exercise*) em uma ordem ou em nível de detalhe determinados pelas informações do modelo do estudante e pelo retorno fornecido pelo estudante.

Os participantes do grupo podem receber diferentes papéis (*roles*). O conceito de Papel

estrutura os membros do grupo em categorias, de acordo com sua participação num cenário. Alguns exemplos de papéis podem ser Líder da Equipe, Membro Participante, representado pelos estudantes, e Instrutor. Estas informações são armazenadas no modelo do Estudante.

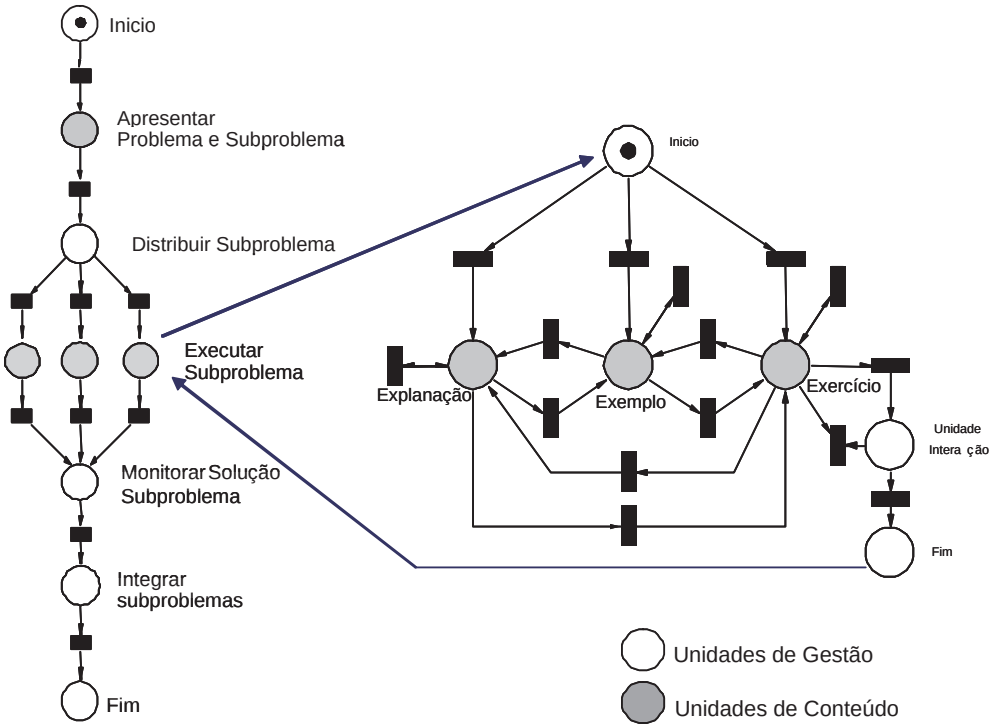

Figura 4.9: RPO Intragrupo que utiliza o protocolo de resolução de problemas

A figura 4.9 ilustra um exemplo da RPO intragrupo, onde é utilizado o protocolo para resolução de problemas proposto na tese de Frigo (2007).

A RPO intragrupo da figura 4.9 parte (a) mostra um conjunto de unidades para apresentar, distribuir problemas e integrar soluções para um grupo de estudantes. Na Figura 4.9 as unidades de gestão são ilustradas no nó/lugar na RPO com a cor branca e as unidades de conteúdo com a cor cinza.

Na unidade execução do problema da figura 4.9 parte (a), é utilizado o protocolo de resolução de problemas (figura 4.9 parte (b)) para oferecer aos estudantes maior flexibilidade para o aprendizado com diferentes tipos de mídias. Durante a resoluções destes problemas os estudantes podem interagir para compartilhar conhecimentos com outros estudantes.

Protocolo de Interação

O conceito de **Protocolo de Interação** (*Protocol*) contém a especificação operacional da atividade do grupo, isto é, a ordem na qual as unidades de atividade são executadas num dado cenário. A especificação de uma instância de um protocolo de interação é um processo de dois passos, conforme ilustrado na Figura 4.10.

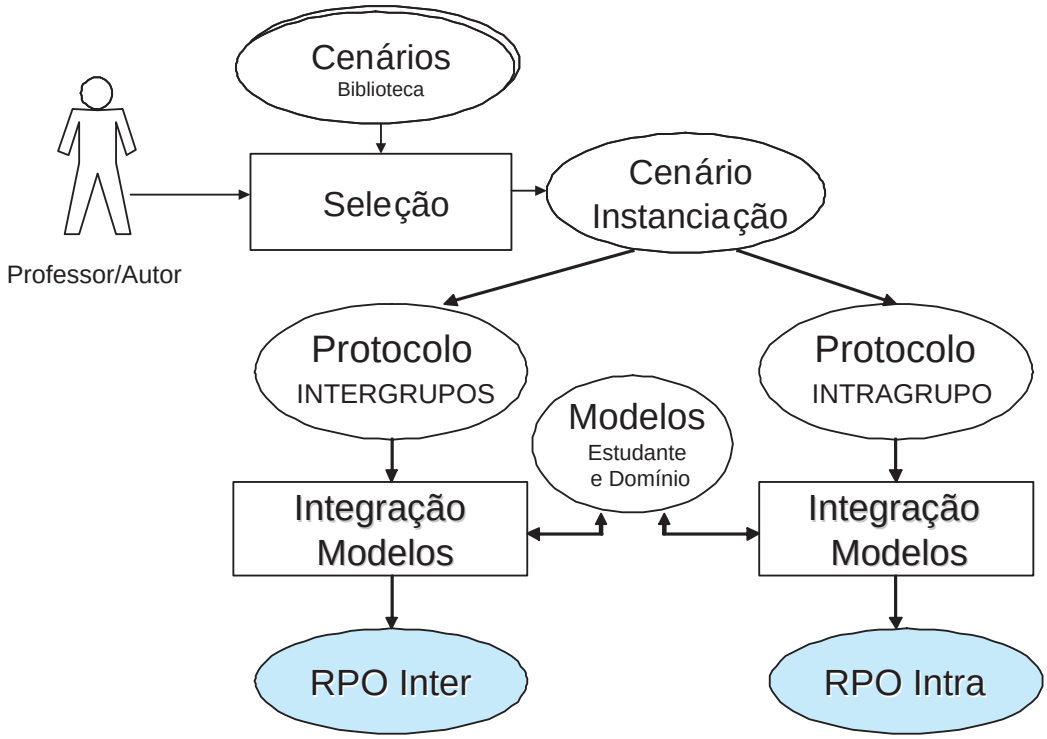

Figura 4.10: Especificação da Atividade de Grupo.

O primeiro passo consiste da seleção de um cenário da biblioteca de cenários e da instanciação de todos os seus atributos.

O segundo passo compila essa informação para produzir duas rede de Petri que definem os protocolos de interação para as atividades intergrupos e intragrupo. O processo de compilação integra automaticamente os modelos nas condições das transições das Redes de Petri. Essa integração fornece o caráter adaptativo dos protocolos de interação.

4.3 Nível de Execução

O nível de execução consiste de um sistema multiagente que gerencia uma atividade de grupo baseada numa instância de uma atividade de grupo definida no nível de especificação. A Figura

4.11 ilustra a instância de uma atividade de grupo.

Para definir tal instância, o desenvolvedor cria uma biblioteca de cenários e os modelos a partir das ontologias. O autor escolhe o cenário mais adequado da biblioteca de cenários, fornece os conteúdos utilizando a FAST e personaliza os parâmetros do cenário (ex.: requerimentos de nível dos estudantes, número máximo e mínimo de membros no grupo, etc.). Essas informações são compiladas numa rede de Petri. Os agentes artificais gerenciadores de grupos e agentes-aprendizes interagem com a SATA e utilizam a rede de Petri para operacionalizar a atividade em grupo no STI. Estes agentes são descritos com mais detalhes nas subseções seguintes.

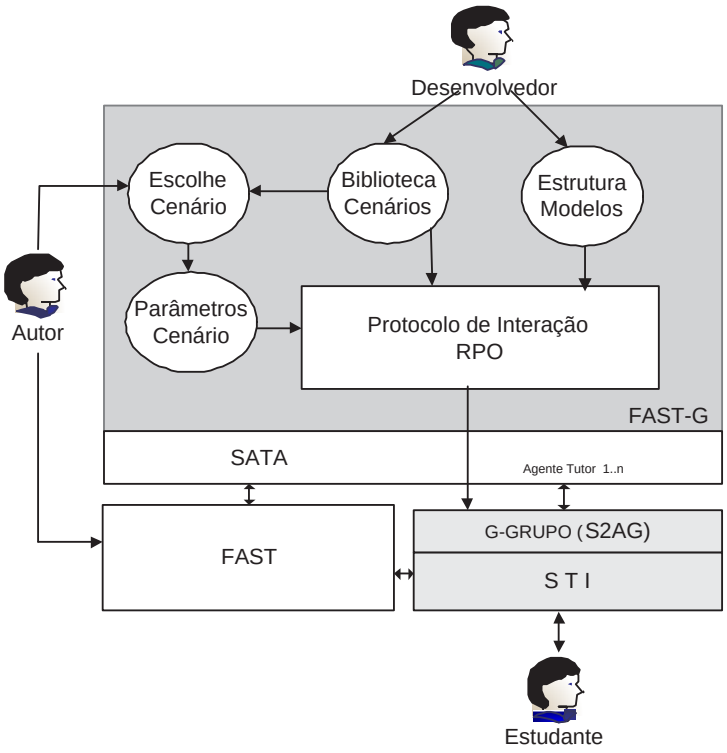

Figura 4.11: Instância de uma atividade de grupo.

4.3.1 S2AG Sociedade de Agentes-Aprendizes e Gerenciadores de Grupos

Para tornar operacional o aprendizado em grupo foi criada uma Sociedade de Agentes-Aprendizes e Gerenciadores de Grupos (S2AG) que contém Agente-Aprendiz (AA), Agente Supervisor de Grupo (SG) e Agente Coordenador de Grupos (CG).

A Figura 4.12 ilustra a arquitetura proposta e as respectivas formas de interação entre os

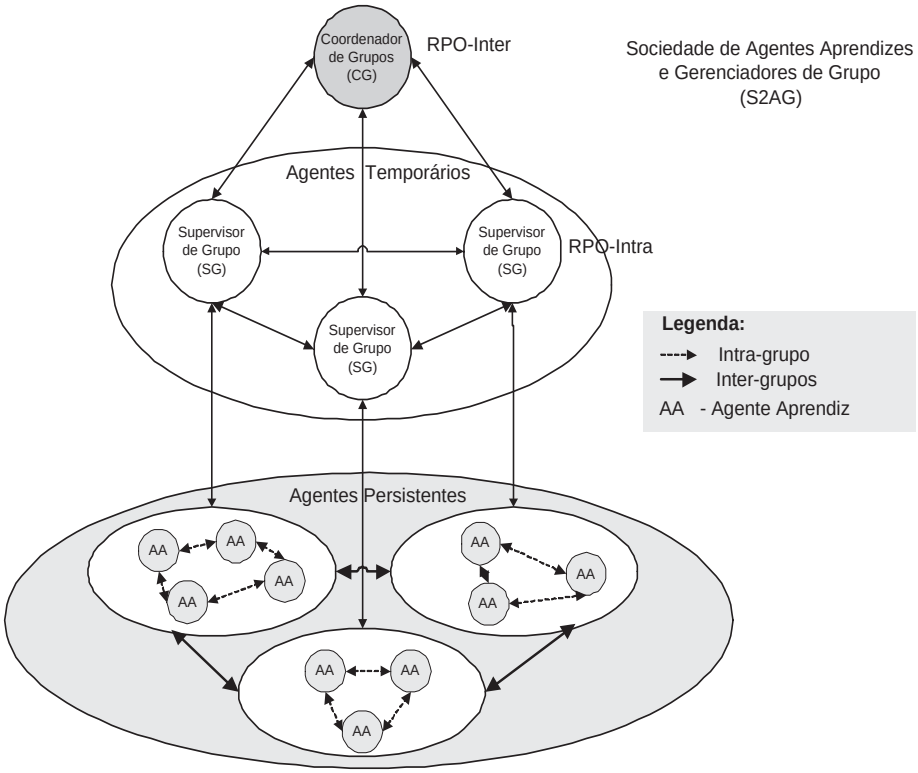

Figura 4.12: Arquitetura Multiagentes S2AG.

agentes-aprendizes, que são: intragrupo, com agentes-aprendizes do mesmo grupo, e intergrupos, entre agentes-aprendizes de grupos distintos.

A seguir são descritos os papéis dos agentes coordenador (AC) e supervisor de grupos (ASG), bem como os processos de interação entre os agentes-aprendizes (AA).

4.3.2 Agente Coordenador de Grupo

O papel do Agente Coordenador de Grupos (ACG) é criar os grupos e gerenciar as interações entre os grupos de acordo com a rede de Petri associada ao protocolo de interação intergrupo definido na instância do cenário. É responsável pela criação e destruição dos Agentes Supervisores de Grupo em tempo de execução e pelo armazenamento permanente de todas as informações relevantes sobre as atividades de grupo. O Agente Coordenador também fornece uma interface para o Instrutor/Professor, através da qual ele pode monitorar a atividade do grupo.

Os resultados do grupo são armazenados pelo agente Coordenador no modelo do grupo e do estudante. Os modelos contêm informações sobre o grupo, tais como: números de estudantes que fazem parte do grupo, nível de conhecimento, desempenho do grupo, informações sobre as interações entre estudantes, etc.

4.3.3 Agente Supervisor de Grupo

O papel do Agente Supervisor de Grupo (ASG) é supervisionar uma atividade de grupo de acordo com a rede de Petri associada ao protocolo de interação intragrupo definido na instância do cenário. O agente Supervisor é criado pelo agente Coordenador e durante a atividade em grupo, antes de encerrar, envia os resultados de seu grupo ao agente Coordenador e então é destruído.

O agente supervisor pode sugerir atividades didáticas que visam facilitar o entendimento da matéria e encorajar a colaboração entre estudantes do mesmo grupo por meio de desafios, jogos, dicas, etc.

4.3.4 Agente-Aprendiz

O papel do Agente-Aprendiz (AA) é representar um estudante. Cada Agente-Aprendiz armazena internamente as informações do modelo do estudante relevantes à gestão do grupo, por exemplo, as atividades de grupo das quais o estudante participou, estatísticas sobre a situação do estudante nesses grupos (líder ou não), o número de comunicações de grupo no qual esteve envolvido, etc. Também pode consultar o modelo do estudante armazenado na SATA do STI.

4.3.5 Processo de interação Intragrupo

No nível intragrupo a colaboração somente é possível entre agentes-aprendizes que fazem parte do mesmo grupo e pode acontecer a qualquer momento enquanto o sistema estiver ativo. Nesse caso os agentes podem interagir utilizando ferramentas síncronas de comunicação, por exemplo, chat ou mural.

O responsável pela gestão é o agente Supervisor, segundo uma rede de Petri cujas fichas contêm a lista de estudantes.

Estas unidades podem ser usadas para identificar os agentes-aprendizes pertencentes ao grupo, ou para permitir que uma mensagem seja emitida, ou consultar os modelos do estudante armazenados, ou ainda para verificar o desempenho dos estudantes numa dada unidade de conteúdo.

4.3.6 Processo de interação Intergrupos

No nível intergrupos a colaboração é possível entre agentes-aprendizes que fazem parte de grupos diferentes.

O objetivo do processo de interação intergrupos é permitir atividades entre os grupos, por exemplo, uma competição.

O responsável pela gestão é o agente Coordenador, que utiliza uma rede de Petri cuja ficha contém a lista de grupos. Esta rede de Petri contém as Unidades de gestão pertencentes ao Cenário.

4.4 Comparação com os trabalhos relacionados

Nesta seção é apresentado o comparativo entre modelo proposto neste trabalho e os trabalhos descritos no capítulo 3. Os resultados foram organizados em um quadro comparativo (Tabela 4.1), onde são destacados o conjunto de aspectos relevantes.

A idéia de instanciação de cenários de atividades para grupos de pessoas foi encontrada na ferramenta *COLE* (Seção 3.5). Nesta ferramenta é definido um cenário para cada serviço, após adquirir as informações relevantes sobre as pessoas através de entrevistas; estas são organizadas e o grupo de pessoas determinam os serviços que devem ser potencialmente implementados. A partir do momento em que diversos grupos definiram os serviços que cada um terá, o engenheiro do conhecimento gera uma tabela com os serviços finais que serão produzidos; cada serviço é definido num cenário, descrevendo como será o seu funcionamento; de acordo com o cenário, são definidas telas para simulação.

A diferença entre a *COLE* e o modelo proposto na presente tese está no armazenamento das informações numa biblioteca, que poderá ser reutilizado pelos professores. As informações relevantes adquiridas sobre as pessoas/estudantes, são armazenadas no modelo de estudante, que também poderão ser adquiridas através de entrevistas.

Ao invés de tabela de serviços finais que é gerada estaticamente, nesta tese foi implementado o protocolo com redes de Petri que permite representar o aspecto dinâmico das interações e pode ser reconfigurado durante a aprendizagem. Para estabelecer uma atividade do grupo, o professor escolhe um cenário da biblioteca, fornece os parâmetros e o conteúdo da atividade. Esta informação é compilada numa rede de Petri que monitora a atividade do grupo.

A idéia do protocolo com as redes de Petri foi inspirada no trabalho de Frigo (2007), que

Tabela 4.1: STI que suportam trabalhos em grupo

Modelo	Modelos	Protocolo/RPO	Gerenciamento de grupos	Biblioteca Cenários
COLE				Instância Cenários
REDEEM	Explora Modelos Domínio e Estudante	Regras de Produção		
EASE	Baseado Ontologias	Regras de Produção		
WHITE RABBIT			Agente Mediador e Agentes Pessoais	
M.B.Vygotsky			Agentes ZPD, Mediador Semiótico e Social	
Chocolato	Baseado Ontologias			

propõe um modelo formal de adaptação para STI, implementado na ferramenta de Autoria FAST (ver seção 2.5). A definição do modelo é baseada em formalismo de Ontologias para a representação do conhecimento envolvido no modelo de domínio e do estudante e em Redes de Petri Objetos (RPO), para definir o modelo pedagógico.

Na REDEEM (Seção 3.1) o modelo do domínio e as informações do modelo do estudante são explorados de maneira semelhante. Na REDEEM, os grupos são definidos como conjuntos de categorias de estudantes, enquanto que, neste trabalho, a definição de grupos fica a critério do professor, que utiliza parâmetros como, uma lista de nomes de estudantes.

A ferramenta EASE (Seção 3.4) é a uma ferramenta de autoria com a utilização de regras para configuração do sistema educacional e a utilização de ontologias nos modelos.

Após a definição de um grupo, ocorre a execução da atividade em grupos, onde é utilizada uma arquitetura multiagentes. A arquitetura torna operacional o aprendizado em grupo, propor-

cionando a colaboração entre os estudantes de um mesmo grupo e também entre estudantes de grupos distintos.

A utilização de arquitetura multiagentes para o gerenciamento de grupos de estudantes pode ser encontrada nos trabalhos *WHITE RABBIT* (Seção 3.2) e o *Modelo Computacional baseado na teoria de Vygotsky* (Seção 3.3).

A idéia de cada usuário ser assistido por um agente foi inspirada no *WHITE RABBIT*. A diferença é que no *WHITE RABBIT* o agente possui um comportamento móvel e autônomo enquanto no S2AG o agente-aprendiz é um agente estacionário que não possui a habilidade para se mover de um ambiente computacional para outro através da rede (ver Seção 2.6.2). No *WHITE RABBIT* o agente Mediador/Assistente equivale ao agente-supervisor apresentado nesta tese, usado para facilitar a comunicação entre os agentes pessoais (nesta tese denominados agentes-aprendizes).

No *WHITE RABBIT* o enfoque é para a formação do grupo, a diferença está na utilização de Algoritmo Genético para detectar os estudantes com afinidades e os agentes móveis que analisam a conversação (chat) entre os usuários.

No *Modelo Computacional baseado na teoria de Vygosky* a sociedade de agentes mediadores é composta por quatro tipos de agentes artificiais: o Agente ZPD, o Agente Mediador/Assistente, o Agente Social e o Agente Semiótico, bem como agentes humanos (estudantes). Comparando com o modelo proposto da presente tese, o agente social representa o modelo do grupo e o agente *Mediating* equivale com as funcionalidades no agente-aprendiz. A interação do agente social (denominado nesta tese de Agente Supervisor) com o grupo são semelhantes às adotadas neste trabalho.

No CHOCOLATO o objetivo é construir um modelo baseado em ontologias que auxilie na análise das interações entre estudantes e no planejamento apropriado de atividades para o grupo de estudantes oferecendo recomendações baseadas nas teorias de aprendizagem. As diferenças deste trabalho com o proposto na presente tese são as estratégias que no CHOCOLATO são baseadas nas teorias de aprendizagem enquanto nesta tese são definidos cenários pré-definidos pelo Autor/Professor.

4.5 Conclusão

Este capítulo descreve o modelo proposto na presente tese para suporte ao aprendizado em grupo em Sistemas Tutores Inteligentes. Este modelo possui dois níveis: o nível de especificação, que adota as ontologias para representar os modelos e redes de Petri para especificar os protocolos de interação, e o nível de execução, que torna operacional o aprendizado em grupo com a arquitetura multiagente, que proporciona a colaboração entre os estudantes do mesmo grupo e estudantes de grupos diferentes.

Na literatura não foi encontrado nenhum trabalho relacionado ao aprendizado em grupos em Sistemas Tutores Inteligentes que utiliza o conjunto das técnicas proposta neste modelo (agentes, ontologia, protocolo/RPO e biblioteca de cenários). Foram encontrados alguns trabalhos parcialmente semelhantes ao modelo proposto nesta tese, por exemplo, o WHITE RABBIT e o Modelo Computacional Baseado na teoria de Vygosky que utilizam um agente como representante do estudante e agentes mediadores para o trabalho de grupos, a EASE que utiliza as ontologias, e a COLE que utiliza a idéia de instanciação de cenários.

Uma das vantagens e diferença do modelo proposto é que ele contempla tanto a aprendizagem intragrupos, através da comunicação síncrona entre os agentes-aprendizes do mesmo grupo, quanto a aprendizagem intergrupos, entre agentes-aprendizes de grupos distintos, podendo ser realizada de forma síncrona ou assíncrona.

No capítulo 5 serão descritos três estudos de casos para validar o modelo criado, sendo um deles implementado e integrado a um STI desenvolvido na ferramenta de autoria FAST para gerar sistemas tutores inteligentes.

Capítulo 5

Aplicação do Modelo Proposto

Neste capítulo são descritos os estudos de caso utilizados para testar o modelo proposto.

Foram criados cenários diferenciados com problemas resolvidos individualmente e/ou em grupos. No primeiro cenário o problema, é dividido em subproblemas e as soluções devem ser combinadas para solucionar o problema original. No segundo cenário, o enfoque é a competição, o problema é o mesmo para todos os estudantes e ocorre a interação intergrupos, onde o Coordenador é responsável pelo controle geral e o supervisor é um mero agente intermediário no grupo. E no terceiro, ocorre a integração de vários cenários numa mesma atividade e o estudante poderá ter o papel do aprendiz e/ou especialista.

O capítulo também discorre sobre a implementação de um cenário, onde foi criado um STI para suportar o aprendizado Colaborativo com o modelo proposto.

5.1 Estudo de caso intragrupo: dividir para conquistar

Esta atividade de grupo desenvolve a estratégia "dividir para conquistar" para a resolução de problemas. Ela supõe um problema que pode ser repartido em um certo número de subproblemas. Cada subproblema pode ser solucionado independentemente e suas soluções devem ser combinadas para solucionar o problema original.

A técnica "dividir para conquistar" foi escolhida por ser um modelo clássico de uma técnica para fixação de conhecimento e utilizada como uma técnica aplicável em salas de aula. Ajusta-se a qualquer conteúdo técnico, artístico ou científico e pode ser aplicada em vários níveis de escolaridade.

5.1.1 Descrição Geral

De acordo com o nível de especificação de uma atividade de grupo, definido na Seção 4.2, os seguintes conceitos são definidos:

- **Grupo:** a atividade precisa de pelo menos um estudante por subproblema.
- **Papéis:** a atividade inclui três papéis: supervisor, solucionador de subproblema e integrador da solução. O papel de supervisão pode ser dado tanto ao instrutor do curso quanto ao agente supervisor. O papel de integrador da solução deve ser dado a um ou mais estudantes que serão responsáveis pela integração das soluções dos subproblemas. A escolha desses estudantes pode ser feita dinamicamente. Por exemplo, o primeiro a completar a solução de um subproblema ou o que tem a melhor nota num STI. Finalmente, o papel de solucionador do subproblema é dado a todos os estudantes que participam da atividade. Os membros dos grupos envolvidos na atividade devem ter a experiência necessária (verificadas no modelo do estudante) para solucionar o problema em consideração.
- **Cenário:** é definido pelas unidades de gestão e unidades de conteúdo.
- **Unidades de gestão:** as unidades de gestão são necessárias para controlar o cenário:

  Nível intergrupos (Coordenador):

  - Formar o grupo: convida os estudantes, recebe aceites e controla o número de membro por grupo. Esta unidade foi detalhada na RPO ilustrada na figura 5.2.
  - Iniciar atividade Grupo: escolha do cenário e criação do agente supervisor responsável pelo grupo.
  - Receber resultado grupo: atualiza modelo do grupo.

  Nível intragrupo (Supervisor):

  - Distribuir subproblemas.
  - Monitorar as soluções dos subproblemas.
  - Integrar e Coordenar os membros do grupos que implementaram incorretamente a interface entre suas soluções.
  - Integrar dos subproblemas: resultados.

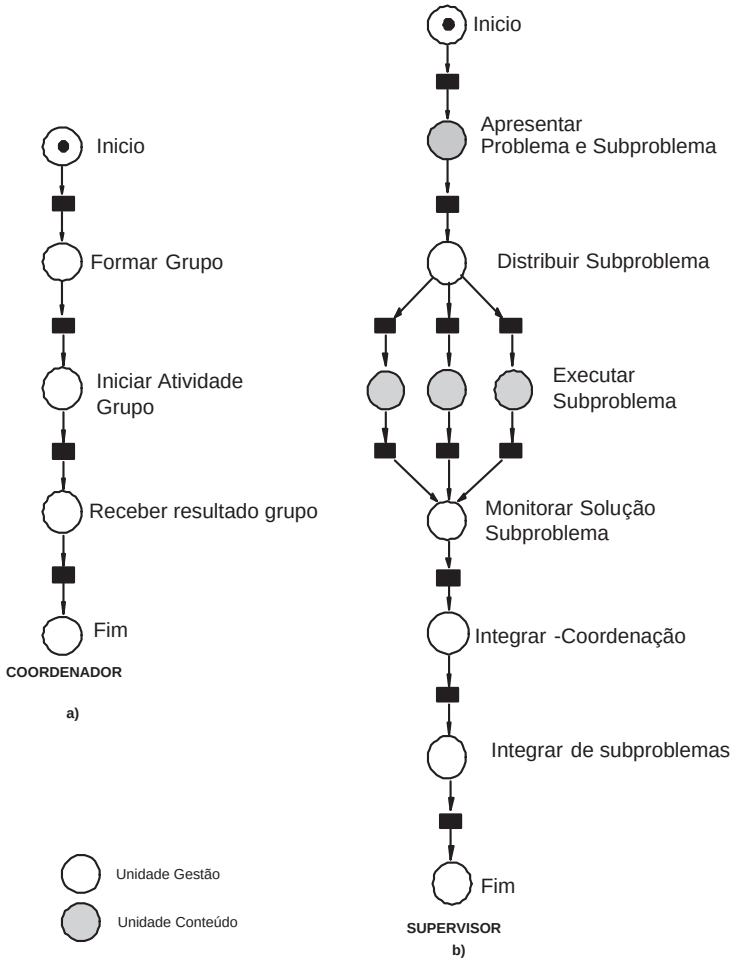

Figura 5.1: Protocolo de interação do cenário Dividir para Conquistar.

- **Unidades de conteúdo:** os conteúdos do cenário, a serem fornecidos pelo autor através da interface de autoria FAST (ver Figura 4.1), consistem das seguintes descrições de problemas:

  1. Apresentar problema e subproblema: uma explicação geral do problema e de seus subproblemas.
  2. Executar subproblema: é utilizado o protocolo de resolução de problemas, onde para cada subproblema possui:

  – uma explanação detalhada;
  – um ou mais exemplos de soluções de problemas similares; e,
  – exercícios: um que teste a solução do subproblema e um que verifique a solução implementada.

Deve-se notar que esses conteúdos são instâncias do conceito do problema (e da unidade de interação) da ontologia do modelo do domínio, e podem também ser usados no contexto de uma interação individual com o STI subjacente.

- **Protocolo de interação:** o protocolo de interação é representado pelas RPO do Coordenador e Supervisor (ilustrada na Figura 5.1), na RPO do Supervisor é acrescentado a RPO para resolução de problemas com as unidades de interação apresentadas aos estudantes, ilustrados na figura (ilustrada na Figura 5.3)

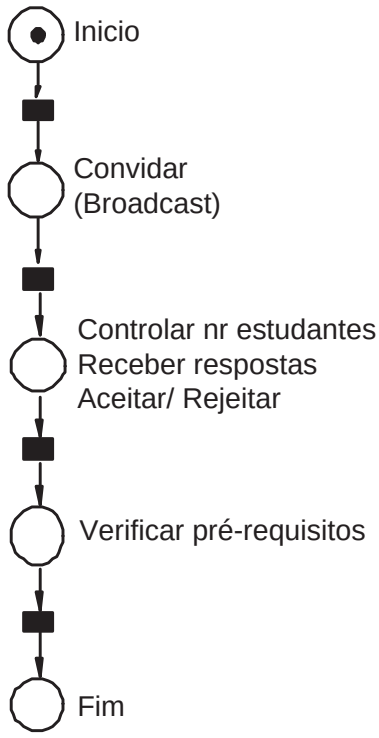

Figura 5.2: Protocolo da Formação de Grupo

Neste contexto, as fichas contêm uma instância do conceito do grupo (Seção 4.3.5). Por razões de legibilidade, foram omitida as pré-condições, ações e as regras de emissão das transições que atuam verificando nas instâncias de objeto representadas pelas fichas. A rede de Petri resultante apenas mantém os aspectos relativos à estrutura comportamental do protocolo. Entretanto, a partir desta estrutura de contrôle, diversas propriedades da rede de Petri podem ser provadas, como a presença de ciclos (seqüências de transições que podem ser infinitamente repetidas), blocagem ou impasse (estado de bloqueio a partir do qual nenhuma transição pode ocorrer), a (in)acessibilidade de um estado (final ou inicial), *boundness* (crescimento infinito do número de fichas) ou a perda de fichas num lugar/estado.

5.1.2 Instância de Cenário

Para instanciar a atividade de grupo para a estratégia de solução de problemas "dividir para conquistar", foi implementada uma atividade de grupo baseada num STI existente de aprendizado individual para o domínio de Estrutura de Informação (Frigo, 2007), em uma disciplina de graduação do curso de Engenharia de Controle e Automação da Universidade Federal de Santa Catarina.

O problema a ser solucionado durante a atividade do grupo é definido como segue:

- Descrição do problema: dada uma linguagem de programação que suporta operações aritméticas sobre inteiros, como ela pode ser estendida para suportar operações para outros tipos de números (racionais, ponto flutuante e complexos).
- Subproblemas: pacotes de operação aritmética para cada um dos três novos tipos de números, incluindo funções de conversão.
- Integração: um pacote de função que integre todos os quatro tipos de números.

A atividade de grupo implementada deve ser desenvolvida durante uma disciplina presencial. As instâncias dos conceitos relevantes envolvidos na definição da atividade de grupo são definidas como segue:

Grupo: Os Estudantes que participarem da atividade devem ter completado as unidades pedagógicas necessárias, neste caso, programação básica e tipos abstratos de dados. As informações referentes as unidades pedagógicas realizadas pelo estudante são obtidas do modelo do estudante (Seção 4.2.2).

Papéis: o papel de supervisor é dado ao professor da disciplina. O papel de solucionador do subproblema é dado a todos os estudantes da sala e o papel de integrador de soluções é dado aos Estudantes que forem membros do primeiro grupo a solucionar com sucesso o subproblema dado.

Unidades de gestão: para formar os grupos é enviado um convite à todos os estudantes da sala. Os Estudantes devem responder com a identificação de seu subproblema preferido. O sistema controla o tamanho máximo de cada grupo automaticamente. Os pré-requisitos necessários também são verificados para cada Estudante. O local da Formação de Grupo na rede de Petri (ver Figura 5.1) é detalhado na rede de Petri ilustrada na Figura 5.2.

A distribuição de problemas é gerada automaticamente. O monitoramento de soluções dos subproblemas é baseado em exercícios incluídos nas unidades de conteúdo. A coordenação

de problemas de interface é realizada sob a responsabilidade do supervisor, através de uma ferramenta de chat e um mural.

Unidades de conteúdo: a rede de Petri de intragrupo implementa as unidades de subproblema e integração de problema é implementada usando a ferramenta FAST. Sua forma geral é ilustrada na Figura 5.3, onde os lugares Exp, Exa e Exe são unidades de interação que apresentam aos estudantes, respectivamente, explicações, exemplos e exercícios.

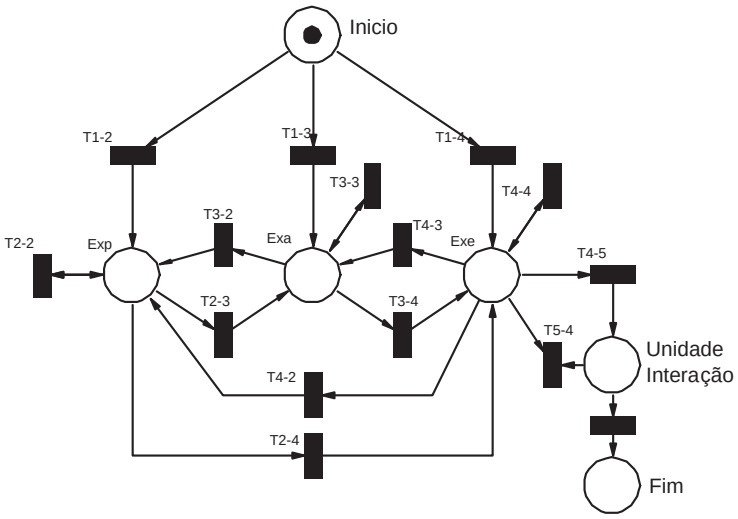

Figura 5.3: Protocolo Resolução Problemas.

5.1.3 Implementação de Estudo de Caso: dividir para conquistar

Para a implementação deste estudo de caso foi utilizada a linguagem de programação JAVA, o servidor de *servlets* TOMCAT, a plataforma para criação e gerenciamento de agentes JADE, e *Java Expert System Shell* (JESS) para o mecanismo de inferência baseados em regras. Estas ferramentas são apresentadas em detalhes no Apêndice A.

Para transformar o grafo de pré-requisitos/RPO em regras para o JESS, foi utilizado o compilador criado por Yamane (2006) apresentado no Apêndice D.

Considerando a existência da estrutura da FAST para implementação do tutor criada por Frigo (2007), a atribuição do Desenvolvedor é a instanciação do cenário e a criação da arquitetura multiagentes S2AG para suportar os grupos de estudantes.

Para incluir o cenário é necessário:

- Definir o cenário, baseado nas ontologias.

- Incluir as RPO/Regras do cenário escolhido.
- Construir os agentes S2AG para suportar grupos e interagir com a SATA.

A atribuição do Autor/Professor é escolher um cenário pré-definido no sistema para o trabalho em grupo, definir os parâmetros, incluir os conteúdos e disponibilizar para os estudantes.

A Figura 5.4 apresenta o protocolo de interação (RdP) e as regras/JESS geradas para os agentes de S2AG gerenciar os grupos de estudantes.

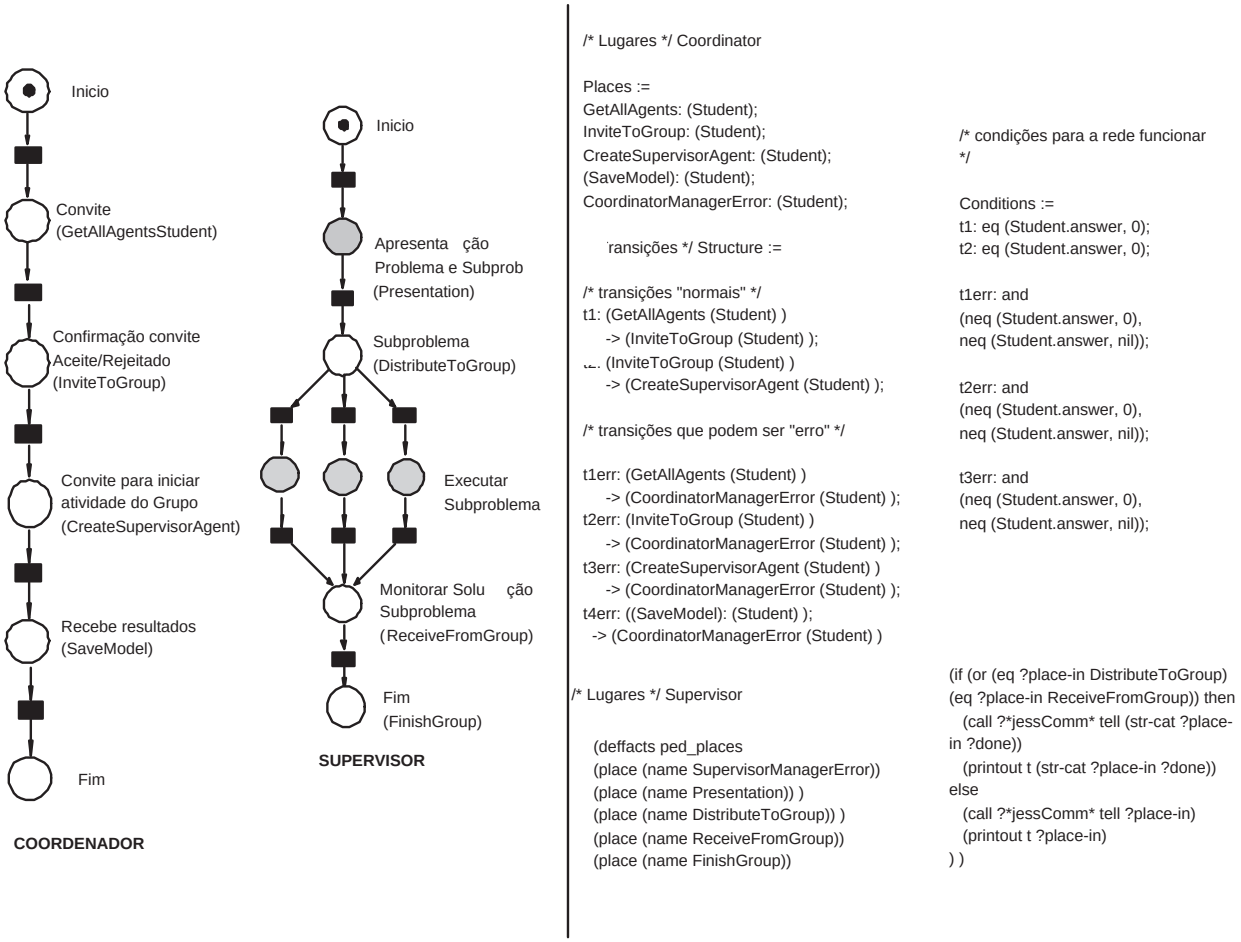

Figura 5.4: Protocolo de interação (RPO) e as regras/JESS.

Maiores detalhes das regras utilizadas pelos agentes Coordenador, Supervisor, e Aprendiz são apresentadas no Apêndice B.

Neste exemplo ocorrem as interações de agentes com os usuários (estudante ou professor) e agentes com agentes (ver Apêndice C).

Como visto na Seção 4.1, há uma sociedade de agentes artificiais SATA, sendo que cada um destes agentes contém dentro de si, um sistema tutor completo. Esta sociedade de agentes se

comunica com o módulo dos agentes gerenciadores de grupos S2AG.

Dentre os agentes implementados existem os que residem dentro do servidor Tomcat. Estes fazem parte da Interface do Estudante, por exemplo os agentes-aprendizes que são o elo de ligação entre o Estudante e os agentes da SATA e S2AG.

Um Agente-Aprendiz é criado a primeira vez que o estudante entra no sistema, conforme a Figura 5.5, e estas informações são armazenadas no modelo do estudante.

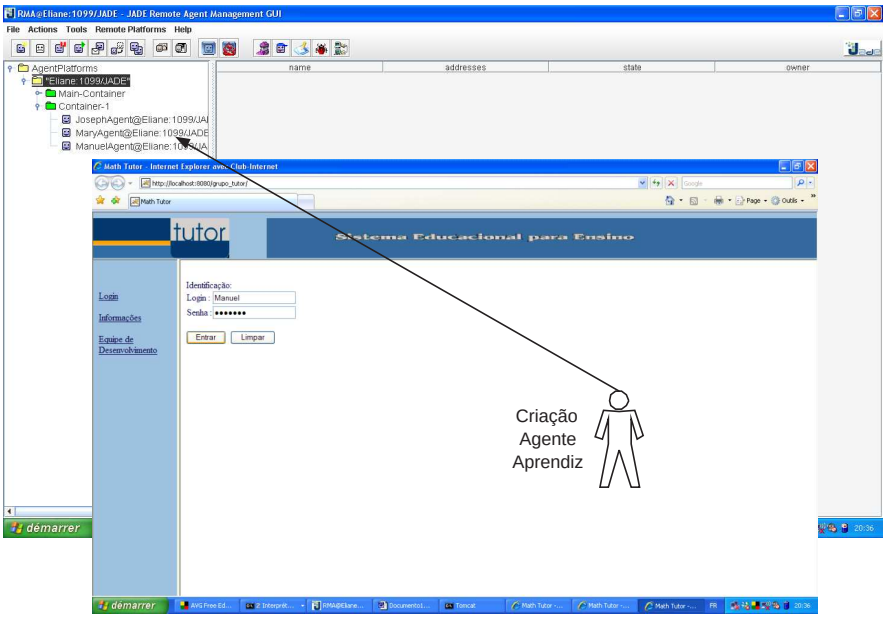

Figura 5.5: Interface para o estudante e criação do agente-aprendiz.

O Agente Coordenador é um agente do tipo persistente (ver Seção 2.6.2) e é responsável pela formação dos grupos de estudantes quando solicitado por um professor ou quando determinados parâmetros são definidos. Por exemplo, quando existir um número mínimo de estudantes para uma determinada atividade.

O Agente coordenador faz o convite (*broadcast*) aos estudantes para participarem do grupo e os Estudantes representados pelos agentes-aprendizes confirmam a presença (conforme a Figura 5.6). Os confirmados serão a lista de nomes dos participantes (ficha) para o agente Supervisor iniciar e acompanhar a atividade em grupo.

O agente supervisor, definido como agente temporário (Seção 2.6.2), é criado pelo agente coordenador para o acompanhamento de um determinado grupo. O agente supervisor distribui uma nova tarefa para o grupo, conforme Figura 5.7.

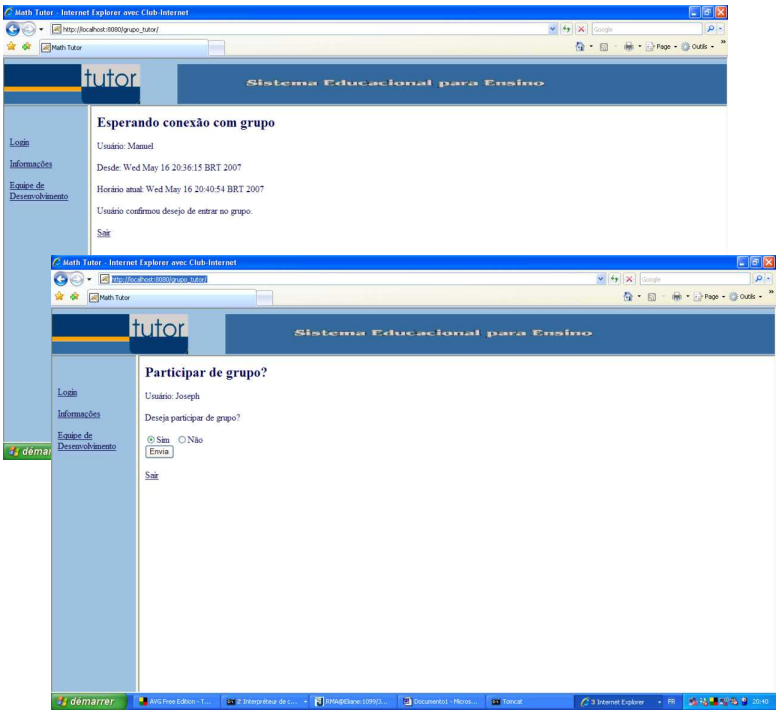

Figura 5.6: Interface para iniciar a atividade em grupos.

O Agente supervisor é responsável também por inicializar a atividade em grupo (Figura 5.8) e quando finalizada ele informa o resultado do grupo para o agente coordenador atualizar o modelo do grupo e é destruído depois da atividade em grupo.

O Agente supervisor envia uma mensagem (*broadcast*) convidando os estudantes para iniciar atividade em grupo. Os estudantes recebem o convite. Através do agente-aprendiz o estudante confirma o recebimento da mensagem e responde com os parâmetros (UP, Problema a fazer) que são enviados para o Agente supervisor.

Durante a atividade do grupo, os Estudantes podem interagir entre si. O agente supervisor interage com os agentes-aprendizes que representam os Estudantes (Figura 5.9). Por exemplo, o Agente Supervisor comunica a todos os participantes do grupo quando entra um novo integrante ou quando um Estudante consegue concluir uma determinada tarefa. Também pode enviar mensagens para motivar a colaboração entre o membros do grupo.

Neste exemplo são distribuídos três problemas diferentes para cada participante com objetivo que os mesmos interajam entre si para buscar as soluções. Quando todos os Estudantes concluírem suas tarefas, o trabalho em grupo acaba, caso contrário o grupo continua ativo até que as tarefas pendentes sejam concluídas. Caso o estudante abandone o sistema, então considera-se concluída a tarefa do mesmo no grupo. A decisão de concluir a tarefa foi escolhida

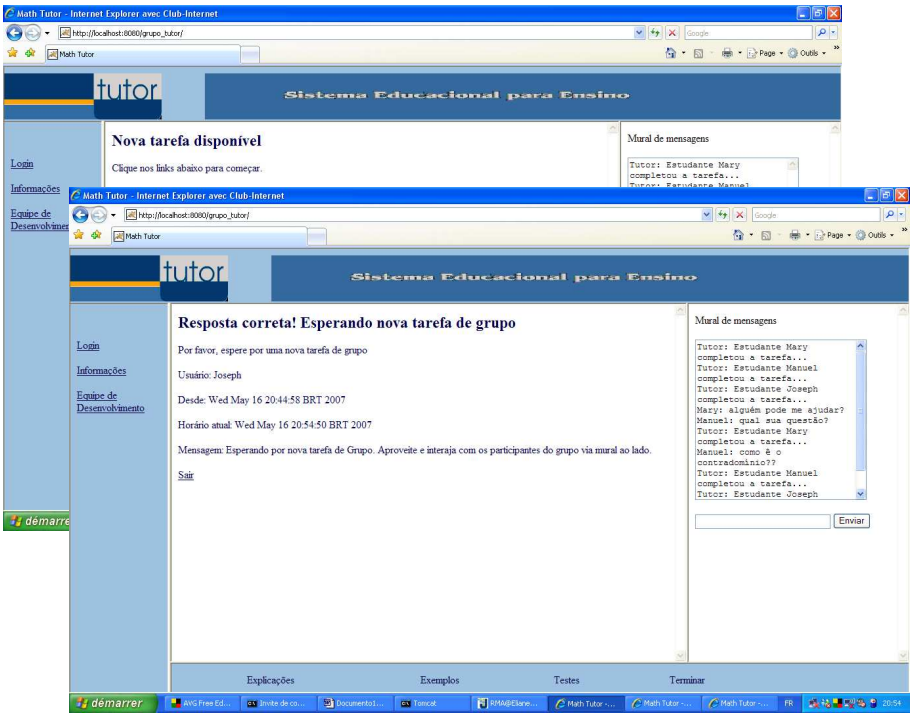

Figura 5.7: Agente Supervisor distribui nova tarefa para o grupo.

para não estender muito o problema, mas nada impede de ter novas outras opções, por exemplo, continuar a atividade com os demais membros do grupo.

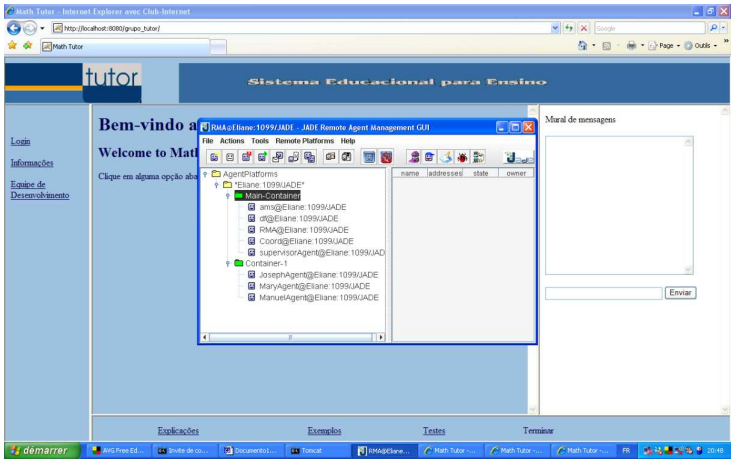

Figura 5.8: Inicializada a atividade em grupo

Neste estudo de caso implementado “dividir para conquistar”foi possível mostrar que o modelo criado para o gerenciamento de grupo é funcional para gerenciar uma atividade de grupo de estudantes. Uma avaliação mais detalhada é necessária para a validação com interações síncrona, isto é, com estudantes numa sala de aula, e assíncrona, com estudantes em locais e

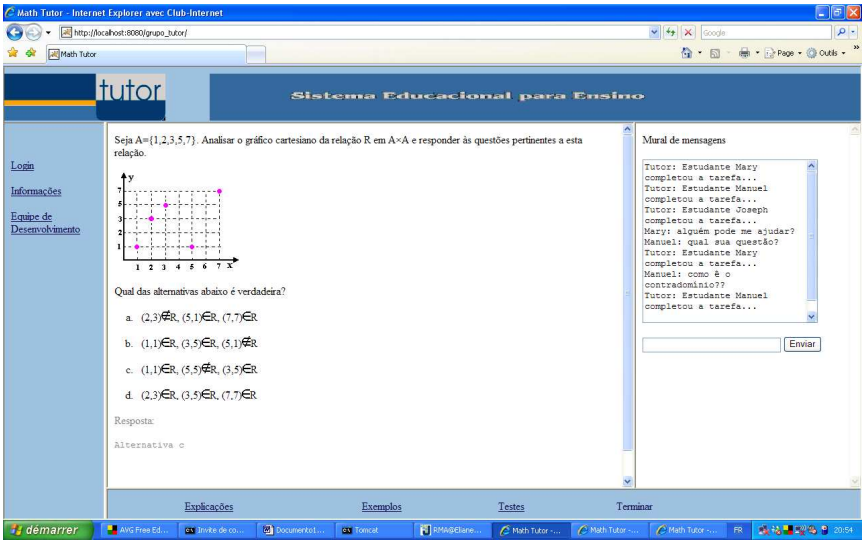

Figura 5.9: Agente Supervisor interage com grupo.

tempos diferentes.

Integração com a SATA

A SATA é composta por Agentes tutores que interagem com a S2AG.

A especificação do cenário contém as Unidades de conteúdo que são usadas para definir as tarefas de resolução de problemas. O Agente Tutor é especializado em subdomínios relacionados a um dado domínio de conhecimento.

O Agente Supervisor necessita destes subdomínios, por exemplo na unidade Distribuição de problemas. Então ocorre a interação entre os agentes Supervisor e Tutor.

Esta interação ocorre sempre que existir Unidades de Conteúdo no protocolo de interação.

Interface com os Estudantes dos Grupos

A interface com o estudante se faz via Web, através de páginas dinâmicas implementadas com Servlets. Portanto, existe a necessidade de se fazer com que o sistema multiagente e os Servlets se comuniquem. Foi implementado o agente-aprendiz que reside no mesmo Container que os agentes do SG2AG. Esta solução foi escolhida para facilitar a comunicação entre os estudantes participantes dos grupos com os Servlets.

Ao entrar no sistema é criado pelo servidor Tomcat uma instância da Classe *StudentAgent*, que representa o usuário do sistema multiagentes. Esta classe contém referência a um Container do Tomcat, que é criado dinamicamente caso ainda não exista, e se conecta ao Container

principal, no qual residem os demais Agentes do Tutor.

Enquanto não começa a atividade no grupo, o estudante poderá utilizar o tutor para aprendizagem individual. Foi criado un Servlet simples que exibe uma página de espera enquanto não começa a interação de grupo.

5.2 Estudo de caso intergrupos: competição entre grupos

Nesta Seção é apresentado um segundo estudo de caso de uma atividade de grupo que denomina-se Autódromo. Esta atividade simula uma corrida de carros e se presta à fixação de um conteúdo com resolução de problemas. Esta Técnica Pedagógica foi adaptada do livro Manual de Técnicas de Dinâmica de Grupo de Sensibilização de Ludopedagogia (Antunes, 1987).

Esta atividade de grupo supõe que o problema é resolvido por equipes de estudantes, o que difere do estudo de caso Dividir para Conquistar (Seção 5.1) em que o problema é divido em subproblemas e resolvido individualmente. Neste cenário ocorre a interação intergrupos enquanto no estudo de caso anterior é intragrupo. Neste cenário o Coordenador é responsável pelo controle geral (de distribuição e integração de soluções) e o supervisor é um mero agente intermediário no grupo.

5.2.1 Descrição Geral

De acordo com o nível de especificação de uma atividade em grupo, os seguintes conceitos são definidos:

- **Grupo:** a atividade precisa de no mínimo dois grupos de estudantes.
- **Papéis:** a atividade inclui três papéis: coordenador, supervisor e líder do grupo. O papel do coordenador é coordenar a competição intergrupos. O papel do supervisor é supervisionar a atividade intragrupo, e o papel do líder do grupo é representar o grupo nas interações intergrupos e o papel de solucionador do subproblema é dado a todos os estudantes participantes.
- **Cenário:** o tipo do cenário é uma competição e os membros envolvidos nesta atividade devem preferir as atividades em grupos (informações do modelo do estudante, ver Seção 4.2.2).

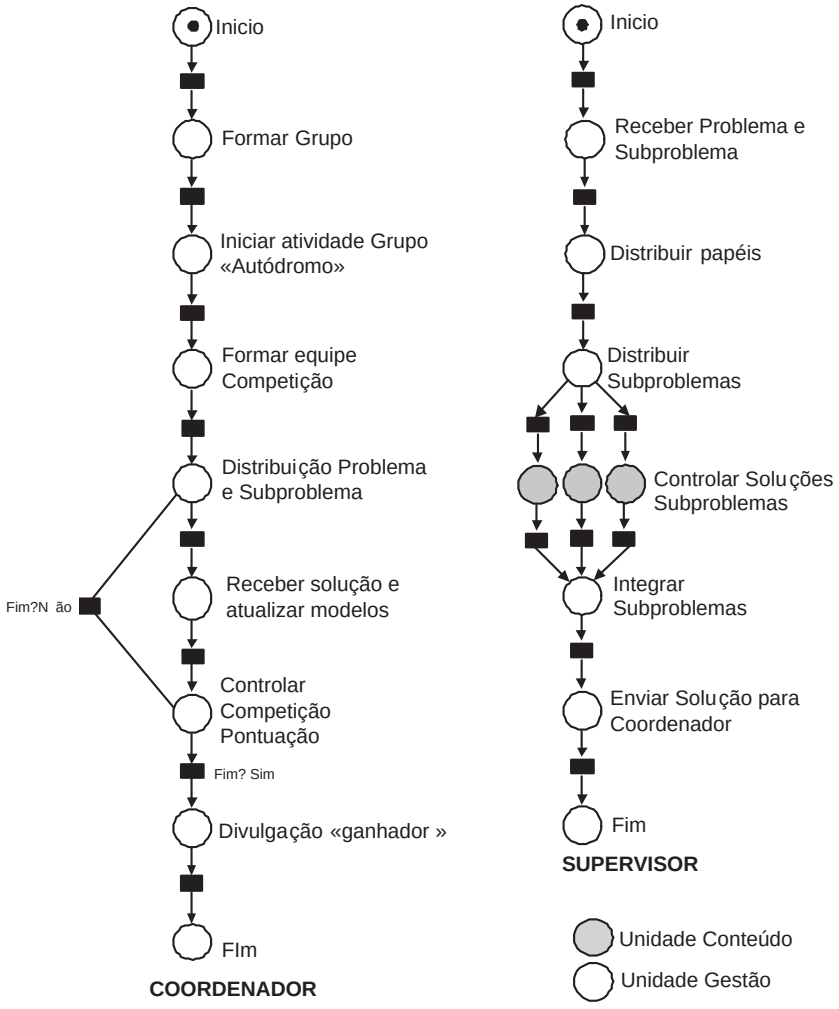

Figura 5.10: Protocolo de interação do cenário Competição.

- **Unidades de gestão:** As unidades de gestão necessárias para controlar o cenário estão ilustradas na Figura 5.10.

  Nível Intergrupos (Coordenador):

  – Formar o grupo: convite aos estudantes, recebe aceites e controla o número de membro por grupo.

  – Iniciar a atividade dos grupos: designação do cenário e criação do agente supervisor responsável pelo grupo. Nesta unidade efetua-se a criação da Pista do Autódromo para os grupos acompanharem os resultados das equipes.

  – Formar as equipes que irão competir neste cenário.

  – Distribuir os problemas para os grupos.

  - Controlar a competição: controle dos resultados, atualização do placar da competição e atualização dos modelos.
  - Divulgar o grupo Ganhador.

  Nível Intragrupo (Supervisor):

  - Receber problema e subproblema do Coordenador.
  - Designar os papéis.
  - Distribuir os subproblemas para os estudantes.
  - Controlar/Monitorar as soluções dos problemas.
  - Integrar os subproblemas.
  - Enviar a solução para o Coordenador.

- **Unidades de conteúdo:**

  Os conteúdos do cenário, a serem fornecidos pelo autor através da interface de autoria, consistem em uma explanação detalhada, através de um ou mais exemplos de soluções de problemas similares. O problema é o mesmo para todos os grupos participantes (competição).

5.2.2 Instância de Cenário

Para instanciar a atividade de grupo "Competição entre grupos", o professor propõe um domínio, por exemplo, de Geografia.

Grupo: o grupo é do tipo heterogêneo, formado por estudantes de diferente níveis de conhecimentos de Geografia.

Papéis: o papel de coordenador é dado ao professor do curso que gerencia todos os grupos. O papel supervisor é dado ao instrutor do grupo. O papel de líder do grupo é dado à um estudante que representa o grupo em determinada interações, por exemplo, enviar uma resposta do problema resolvido pelo grupo ao supervisor. A escolha do líder é aleatória. E o papel de solucionador do problema é dado a todos os estudantes.

Unidades de gestão: A atividade implementada usa uma formação síncrona de grupo na qual um convite é enviado a todos os estudantes. Os Estudantes devem confirmar a participação.

O sistema controla o tamanho máximo de cada grupo automaticamente. Os pré-requisitos necessários também são verificados para cada Estudante.

O grupo é divido em equipes para a competição. E então inicia-se a atividade dos grupos com uma explanação, a apresentação do Autódromo (ver Tabela 5.1), e a distribuição dos problemas (jogo de duas questões) para os grupos. Em seguida é atribuído um tempo de cinco minutos para que o grupo escolha uma das quatro alternativas de resposta. Esgotado o tempo, os estudantes são comunicados e são encerradas as discussões. O Supervisor escolhe em ordem alternada um líder de grupo em cada grupo que deve imediatamente responder a questão.

Tabela 5.1: Placar - Pista do Autódromo.

Pontos Conquistados													
Equipe	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**	**1200**	**1500**	Resultado
Alfa	X	X											
Beta	X	X	X										
Gama	X												
Delta	X												
Pista com óleo Volte uma casa				**Desastre na Pista Pare uma rodada**				**Reabastecimento Volte duas casas**			**Acidente Pare 2 vezes**		

Neste estudo de caso a distribuição de problemas é gerada automaticamente. O monitoramento de soluções dos problemas é baseado em exercícios incluídos nas unidades de conteúdo. A coordenação de problemas é realizada sob a responsabilidade do supervisor, através de uma ferramenta de chat e um mural. O Supervisor vai informando no mural ao lado do nome do grupo a alternativa escolhida de cada grupo.

Quando todos os grupos tiverem apresentado suas respostas, é anunciada a alternativa correta e assinalado no quadro o avanço ou não, das equipes. Esse registro é feito pelo Coordenador na "pista do autódromo". Anotando um X para as equipes que acertaram as questões, esta anotação corresponde ao avanço ou acerto das mesmas em relação às concorrentes.

Caso utilize uma pista como a sugerida na tabela 5.1, convenciona-se que todo acerto corresponde a um avanço (assinalado com "x") na pista e todo erro equivale a sanções previstas.

Por exemplo, as equipes Alfa e Beta acertaram a primeira questão e passam a contar com 100 pontos, acertaram a questão seguinte e agora possuem 200; na terceira questão apenas a equipe Beta acerta, indo assim para os 300 pontos, e como a equipe Alfa errou, não avança uma casa e, desta forma fica com 200 pontos e assim é cancelado o registro feito na pista.

O controle da competição é de responsabilidade do coordenador e o controle das interações no grupo é de responsabilidade do Supervisor. No final da atividade é apresentado um ganhador

da competição e os modelos são atualizados.

Unidades de conteúdo: Sendo um único problema para o grupo, os membros deverão chegar num consenso de qual é a solução adequada para o problema.

Neste estudo de caso o problema é um conjunto de questões objetivas, do tipo falso/verdadeiro, escolha múltipla ou outro e cada questão deve formar um conjunto.

5.3 Estudo de caso intergrupos: painel

Nesta Seção é apresentado um terceiro estudo de caso de atividade de grupos. Esta atividade de grupo, denominada Painel, é uma técnica ajustável a qualquer conteúdo de qualquer grau de ensino e útil para sala de aula ou qualquer tipo de atividade que envolva necessidade de fixação de conhecimento.

Neste estudo de caso existe o conceito de grupo e equipes. O grupo é a união de todos os estudantes que participam deste cenário e as equipes são pequenos grupos formados pela divisão deste grupo.

Nesta atividade de grupo poderão ser utilizados vários domínios. A formação de equipe é necessária para a distribuição dos subdomínios. Cada equipe recebe um subdomínio. Cada equipe poderá ter um cenário diferente para o aprendizado. Após um determinado tempo de interação entre os estudantes e o tutor, cada equipe deverá elaborar um relatório ou uma explanação sobre o conteúdo ensinado.

Após todas as equipes terem seus relatórios prontos começará o Painel, onde cada equipe irá expor para as outras equipes o assunto estudado e responder as questões. Para validar o aprendizado, poderá ser aplicado no final um questionário. O painel poderá ser aberto para todos os grupos ou para grupos restritos.

A diferença dos demais estudos de caso é a integração de vários cenários na mesma atividade, a integração de vários grupos/equipes e a diversidade de domínios. Neste cenário o estudante repassa o que aprendeu para outros estudantes.

5.3.1 Descrição Geral

De acordo com o nível de especificação de uma atividade em grupo, os seguintes conceitos são definidos:

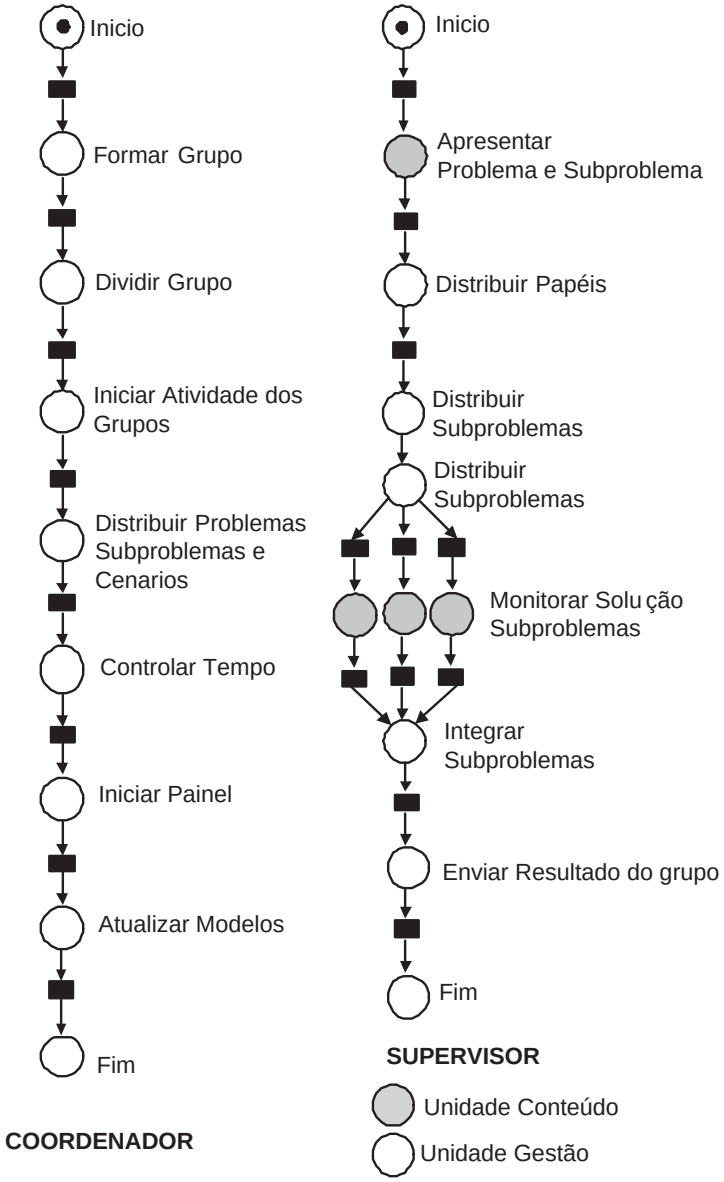

Figura 5.11: Protocolo de interação do cenário Painel.

- **Grupo:** a atividade precisa de no mínimo um grupo de estudantes para cada subdomínio do Problema.

- **Papéis:** a atividade inclui quatro papéis: coordenador, supervisor, líder do grupo e solucionador do subproblema. O papel do coordenador é coordenar as atividades intergrupos. O papel do supervisor é supervisionar a atividade intragrupo. E o papel do líder do grupo é representar a equipe nas interações intergrupos. E o papel de solucionador do subproblema é dado a todos os estudantes participantes.

- **Cenário:** o tipo do cenário é um Painel, onde o grupo aprende um determinado assunto, e depois ensina o conteúdo para outro grupo e vice-versa.

5.3.2 Instância de Cenário

Este exemplo de cenário é mais utilizado para a resolução de problemas da área de ciências exatas.

Grupo: os grupos são homogêneos e criados aleatoriamente.

Papéis: a atividade inclui três papéis: coordenador, supervisor e líder do grupo. E o papel de aprendiz e/ou especialista é dado aos estudantes participantes.

Cenário: neste estudo de caso o cenário poderá ser composto por outros cenários. Eventualmente, equipes diferentes podem utilizar cenários diferentes.

Unidades de gestão:

Nível Intergrupos (Coordenador): Para formar o grupo de estudantes é enviado um convite à todos os estudantes. Os estudantes devem confirmar a participação na atividade do grupo. Em seguida é realizada a divisão do grupo de estudantes em equipes. O sistema controla a entrada e saída de estudante no grupo automaticamente conforme a lista definida pelo professor e as informações do modelo do estudante. Para os grupos é determinado um tempo para resolver o problema, iniciar a atividade Painel e atualizar os modelos com os resultados.

Nível Intragrupo (Supervisor): apresentar o problema para o grupo, distribuir os papéis para os estudantes, distribuir os subproblemas, monitorar a solução, integrar os subproblemas e enviar os resultados para o Coordenador. Após definido o grupo de estudantes será repartido em sub-grupos para iniciar a atividade. Na atividade são distribuídos os papéis, os subproblemas e controlado o tempo para a resolução do subproblema. Concluído este tempo, cada grupo entregará a solução e iniciará o Painel. As unidades de gestão, ilustradas na Figura 5.11, são formação de grupo, divisão em subgrupos, iniciar a atividade com a distribuição de papéis, controlar o tempo, monitoração da solução, envio de resultados e iniciar o Painel.

O Painel é a troca de conhecimento entre os grupos. As unidades de gestão são ilustradas na Figura 5.12. A atividade do Painel inicia quando todos os grupos entregarem os relatórios das atividades intragrupos. O agente Supervisor é responsável pelos controles de recebimento dos relatórios, apresentações, interações e avaliação que poderá ser questões sobre o domínio apresentado.

Unidades de conteúdo: ilustradas na Figura 5.11, inicialmente é distribuído o problema e cenário. Após ocorre a apresentação do problema e subproblemas, distribuição e integração dos mesmos. O diferencial deste estudo de caso é a possibilidade da aplicação de cenário diferente para a resolução do problema numa atividade intragrupo. Durante as atividades intragrupos e

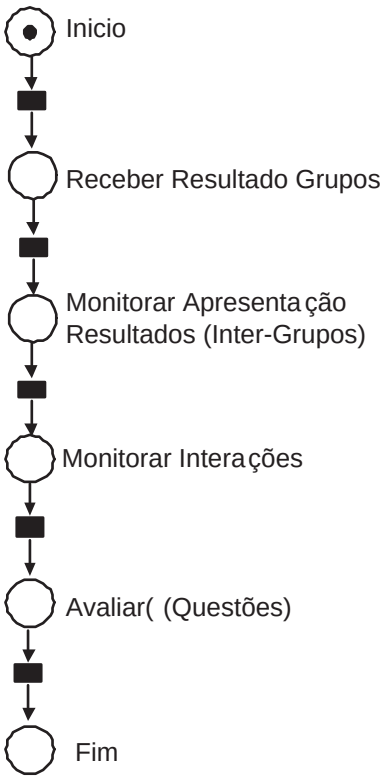

Figura 5.12: RPO da unidade de gestão Painel.

intergrupos são atualizados os modelos de estudantes e de grupos.

Neste estudo de caso o grupo é homogêneo, porém outro estudo de caso poderia ser criado com a mesma idéia do Painel mas com grupos heterogêneos. Poderia ser lançado um desafio para todos os estudantes, estimulando os estudantes a trabalhar em conjunto porque cada estudante é conhecedor de um domínio específico.

5.4 Conclusão

Este capítulo descreveu os estudos de caso utilizados para validar o modelo proposto na presente tese, onde foram criados cenários diferenciados com problemas resolvidos individualmente e/ou em grupos. E também discorreu sobre a implementação de um STI para suportar o aprendizado Colaborativo com o modelo proposto na presente tese, integrado na ferramenta de autoria FAST que é baseada no modelo MATHEMA.

O primeiro cenário foi uma atividade de grupo que desenvolve a estratégia "dividir para conquistar" para a resolução de problemas. Ele supõe um problema que pode ser repartido em um certo número de subproblemas. Cada subproblema pode ser solucionado independentemente e suas soluções devem ser combinadas para solucionar o problema original. No segundo

cenário, o enfoque é a competição, o problema é o mesmo para todos os estudantes e ocorre a interação intergrupos, onde o Coordenador é responsável pelo controle geral e o supervisor é um mero agente intermediário no grupo. E no terceiro, ocorre a integração de dois cenários numa mesma atividade e o estudante poderá ter o papel do aprendiz e/ou especialista.

Capítulo 6

Conclusão e trabalhos futuros

6.1 Conclusão

Neste trabalho de tese foi proposto um novo modelo para gerenciamento de grupos em Sistemas Tutores Inteligentes, que utiliza uma biblioteca com vários cenários de atividades de grupos. Para estabelecer uma atividade do grupo, o professor escolhe um cenário da biblioteca, fornece os parâmetros e o conteúdo da atividade. Esta informação é compilada numa rede de Petri que monitora a atividade do grupo.

O modelo proposto pela Autora da presente tese explora modelo do domínio e as informações do modelo do estudante, além do modelo de grupo que foi criado de tal forma que o cenário é dividido em unidades. Cada unidade do cenário pode ser reaproveitada pelos Autores para construir novos cenários com novas estratégias pedagógicas.

Para a execução da atividade em grupos é utilizada uma arquitetura multiagentes. A arquitetura torna operacional o aprendizado em grupo, proporcionando a colaboração entre os estudantes de um mesmo grupo e também entre estudantes de grupos distintos . É definida uma sociedade heterogênea composta por agentes-aprendizes e agentes gerenciadores de grupos (coordenador de grupos e supervisores de grupos). Os agentes-aprendizes são responsáveis por assistir o estudante e representá-los no sistema. O agente coordenador e os agentes supervisores de grupos são responsáveis por gerenciar os grupos e acompanhar a interação entre os estudantes.

Este trabalho estende o modelo MATHEMA, que é um modelo para o desenvolvimento de STI baseado numa arquitetura multiagentes e integra a ferramenta de autoria FAST para o aprendizado individualizado. Este aproveitamento do modelo e da ferramenta foi importante

para proporcionar também o aprendizado individualizado aos estudantes no trabalho de grupo.

Uma das vantagens do modelo proposto é que ele contempla tanto a aprendizagem intra-grupos, através da comunicação síncrona entre os agentes-aprendizes do mesmo grupo, quanto a aprendizagem inter-grupos, entre agentes-aprendizes de grupos distintos, podendo ser realizada de forma síncrona ou assíncrona.

A originalidade deste trabalho deve-se também ao fato da integração de diferentes técnicas, a saber, ontologias, redes de Petri, biblioteca de cenários e agentes.

6.2 Contribuições

A principal contribuição desta tese é a proposta de um modelo formal para suporte ao aprendizado em grupo em Sistemas Tutores Inteligentes, onde os estudantes escolhem a forma de aprendizado com cenários diversificados e os professores configuram o domínio de ensino do STI.

Algumas vantagens encontradas no modelo são:

- **Reutilização de Unidades** O cenário é constituído de unidades, por exemplo gestão e conteúdos. Cada unidade do cenário pode ser reaproveitada pelos Professores/Autores para construir novos cenários com novas estratégias pedagógicas.

- **Abordagens intra e intergrupos** o trabalho contempla tanto a aprendizagem intra-grupos, através da comunicação síncrona entre os agentes-aprendizes do mesmo grupo, quanto a aprendizagem intergrupos, entre agentes-aprendizes de grupos distintos, podendo ser realizada de forma síncrona ou assíncrona.

- **Arquitetura hierarquizada para controle de grupos:** a arquitetura multiagentes proposta para controlar os grupos foi definida de tal forma que existe um agente supervisor por grupo e um agente coordenador de grupos que possui uma visão geral de todos os grupos de estudantes.

- **Modelo do Grupo:** as decisões de gerenciamento dos grupos são baseadas em um conhecimento “profundo”, pois leva em conta a estruturação do modelo do domínio e as informações do modelo do estudante, definidas no modelo MATHEMA Costa (1997).

6.3 Trabalhos Futuros

Sugere-se para futuros trabalhos:

- Construir a ferramenta de autoria FAST-G para operacionalizar a interação do Professor/Autor com o STI. Por exemplo, na inclusão de conteúdos e na definições de parâmetros para os grupos.
- Implementar diferentes cenários.
- Reutilizar as unidades de gestão em cenários diversificados.
- Permitir, a partir da biblioteca de cenários, a seleção personalizada e automática do conteúdo para gerar novos recursos e adaptação conforme preferências dos estudantes e professores.
- Validar o modelo com interações síncrona, estudantes numa sala de aula, e assíncrona, com estudantes em locais e tempos diferentes.
- No nível especificações, detalhar melhor a representação do modelo de estudante e domínio.

Apêndice A

Técnicas e Ferramentas

Este apêndice descreve as ferramentas que foram utilizadas para a modelagem e o desenvolvimento do sistema de gerenciamento de grupos proposto nesta tese.

A.1 JADE - *Java Agent DEvelopment Framework*

JADE é um ambiente para o desenvolvimento de aplicações baseadas em agentes em linguagem de programação Java TILAB (2007). JADE é baseado no padrão FIPA (*Foundation for Intelligent Physical Agents*) para interoperabilidade entre sistemas multiagentes. Pode ser considerado como um *middleware* de agentes que implementa um *framework* de desenvolvimento e uma plataforma de agentes.

Em JADE cada agente é implementado como uma thread Java e então é inserido em um repositório de agentes chamado de *container*. O *container* é responsável por todo o suporte a execução do agente. A Figura A.1 ilustra, genericamente, a plataforma de agentes JADE distribuída em três máquinas (*Host*) distintas. A execução do JADE é realizada pela máquina virtual java, presente em cada máquina.

JADE permite que em cada máquina (*Host*) seja possível executar mais de um agente de maneira concorrente. A comunicação entre os agentes é provida pelo mecanismo de invocação remota de métodos (RMI do Inglês *Remote Method Invocation*), mecanismo de comunicação nativo da linguagem de programação Java. A vantagem em se utilizar uma máquina virtual para a execução dos agentes é a facilidade de trabalhar com distintas configurações de hardware.

Um agente JADE é simplesmente uma instância da classe Agent, no qual os programadores ou desenvolvedores deverão escrever seus próprios agentes como subclasses de Agent,

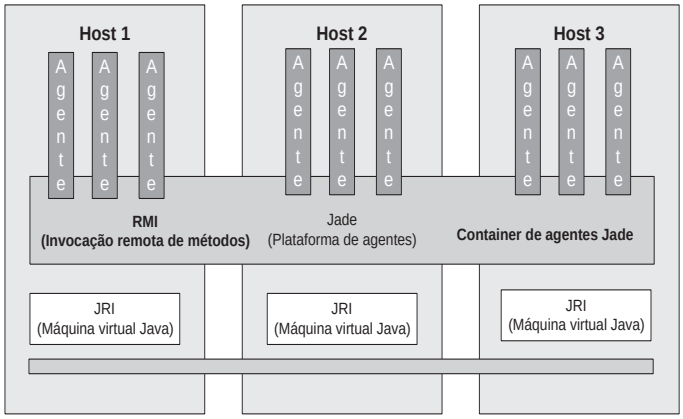

Figura A.1: Visão geral da Plataforma de Agentes JADE.

adicionando comportamentos específicos de acordo com a necessidade e objetivo da aplicação, através de um conjunto básico de métodos, e utilizando as capacidades herdadas que a classe Agent dispõe tais como mecanismos básicos de interação com a plataforma de agentes (registro, configuração, gerenciamento remoto, etc).

JADE também provê suporte ao desenvolvimento de agentes móveis. Neste caso, os agentes podem migrar e clonar-se entre os containeres que fazem parte da aplicação multiagente. Também disponibiliza uma biblioteca que contém a definição de ontologias para a mobilidade em JADE, vocabulário com uma lista de símbolos usados e todas as classes Java que implementam essas ontologias.

A.2 Servlet

Um Servlet é uma classe Java que é instanciada e executada em associação com um servidor WEB, atendendo requisições realizadas por meio do protocolo HTTP (*Hypertext Transfer Protocol*) Deitel e Deitel (2006). Um Servlet é uma API para a construção de componentes do lado servidor com o objetivo de fornecer uma padrão para comunicação entre clientes e servidores. Por exemplo, um Servlet pode receber dados através de um formulário HTML (*Hypertext Markup Language*), processar esses dados e gerar alguma resposta dinamicamente para o cliente que fez a requisição.

Servlets não possuem interface gráfica e suas instâncias são executadas dentro de um ambiente Java denominado de *container*. Um container gerencia as instâncias dos Servlets e provê os serviços de rede necessários para as requisições e respostas. O container atua em associação com servidores Web recebendo as requisições reencaminhadas por eles.

Basicamente um container é responsável por: inicializar os Servlets, redirecionar os pedidos dos clientes para os respectivos Servlets, e finalizar os Servlets. Como existe somente uma instância de cada Servlet, o container pode criar várias threads de modo a permitir que uma única instância de um Servlet atenda mais de uma requisição simultaneamente.

Características dos Java Servlets Deitel e Deitel (2006):

- **Eficiência**: o código de inicialização do servlet é executado apenas a primeira vez que ele é carregado pelo servidor HTTP.
- **Persistência**: servlets podem manter estados entre requisições de clientes, uma vez que ele é carregado, permanece residente na memória enquanto serve requisições de clientes.
- **Portabilidade**: como são desenvolvidos em Java, são portáveis, podendo ser executados em diferentes sistemas operacionais sem a necessidade de recodificação.
- **Robustez**: podem tratar exceções de forma eficiente, como qualquer outra aplicação em Java.
- **Extensibilidade**: podem ser beneficiar das características da programação orientada a objetos, como, herança e polimorfismo, para a criação de objetos mais apropriados para uma aplicação em particular.
- **Segurança**: executam do lado do servidor, herdando assim as características de segurança do servidor HTTP.

Um dos servidores mais popular e também gratuito para Servlets é o Tomcat. O Tomcat é tanto a implementação da API Servlet como a implementação de um container, que pode trabalhar em associação com um servidor Web, como o Apache, por exemplo, ou pode também trabalhar isoladamente, desempenhado o papel de um servidor Web. O Tomcat foi o servidor utilizada na implementação do modelo para o aprendizado em grupos.

A.3 Ontologias

O termo ontologia pode ser considerado como uma visão abstrata e simplificada do mundo que se deseja representar para algum propósito. Ontologia é uma antiga disciplina que vem desde o estudo feito por Aristóteles sobre as categorias e a metafísica, é a ciência que estuda o ser e suas propriedades.

As ontologias geralmente descrevem:

- Indivíduos (Instâncias): os objetos básicos;
- Classes (Conceitos): conjuntos, coleções ou tipos de objetos;
- Atributos: propriedades, características ou parâmetros que os objetos podem ter e compartilhar;
- Relacionamentos: as formas como os objetos podem se relacionar com outros objetos.

Segundo Gomes-Pérez (1999), a definição de uma ontologia busca solucionar problemas relacionados à falta de conhecimento compartilhado facilitando a comunicação entre as partes envolvidas. Ontologias objetivam capturar o conhecimento consensual de um modo genérico, podem ser recusáveis e compartilhadas entre aplicações (software) e por grupos de pessoas. Ontologias são normalmente construídas por um grupo de pessoas em diferentes locais.

Para a comunidade de Inteligência Artificial, ontologias são teorias que especificam um vocabulário relativo a um certo domínio. Este vocabulário define entidades, classes, propriedades, predicados e funções e as relações entre estes componentes (Guarino, 1998).

Em IA o conhecimento de um domínio é representado num formalismo declarativo e o conjunto de objetos que podem ser representados é chamado de universo de discurso. O conjunto de objetos, e suas relações são representados num vocabulário em um programa que pode representar o conhecimento.

Segundo Guarino (1997) as ontologias podem ser classificadas de acordo com sua dependência em relação a uma tarefa específica ou a um ponto de vista:

- Ontologias de Alto Nível: descrevem conceitos bem gerais;
- Ontologias de Domínio: descrevem um vocabulário relacionado a um domínio genérico. Por exemplo, uma disciplina a ser ensinada pelo tutor;
- Ontologias de Tarefas: descrevem uma tarefa ou uma atividade, como avaliação das respostas dadas pelos estudantes a um problema relacionado ao conteúdo ensinado;
- Ontologias de Aplicação: descrevem conceitos que dependem tanto de um domínio específico como de uma tarefa específica, e geralmente são uma especialização de ambos.

O uso de ontologias pode contribuir na solução dos seguintes problemas: representação, reuso, compartilhamento, aquisição e integração de conhecimento; processamento de linguagem natural; tradução automática; comunicação de informação entre sistemas, agentes, empresas ou pessoas, recuperação de informação e especificação de software.

O uso das ontologias em Sistemas Multiagentes é bastante desejável, pois as ontologias servem como ferramenta para organização, reuso e disseminação de conhecimento já especificado, facilitando a construção de novos agentes. Além disso, uma ontologia comum define um vocabulário com o qual os agentes trocarão mensagens (informações), este vocabulário nada mais é do que uma descrição dos objetos que pertencem ao domínio em questão. Ainda que os agentes compartilhem um mesmo vocabulário isso não implica que eles possuam o mesmo conhecimento. Também não se espera que um agente que se comprometa com uma ontologia seja capaz de responder a todas as perguntas que possam ser formuladas com o vocabulário compartilhado.

Gruber (1993) propõe um conjunto básico de critérios para o projeto de ontologias:

- Claridade: uma ontologia deve comunicar efetivamente o significado pretendido dos termos definidos. As definições devem ser objetivas; devem ser formais, ou seja, independentes de contexto social ou computacional; completas (quando possível); e documentadas em linguagem natural.

- Coerência: uma ontologia deve ser coerente, inferências feitas devem ser consistentes com as definições; ou seja, se uma sentença que pode ser inferida dos axiomas contradizer uma definição ou exemplo informal, então a ontologia ´e incoerente.

- Extensibilidade: uma ontologia deve ser capaz de definir novos termos para usos especiais baseados no vocabulário existente, sem que seja requerido uma revisão das definições existentes.

- Minima influência de código: a influência de código resulta quando as escolhas de representação são feitas puramente para conveniência da notação ou implementação. Estas influências devem ser minimizadas.

- Mínimo comprometimento ontológico: uma ontologia deve fazer o mínimo de alegações possíveis acerca do mundo sendo modelado, permitindo aos parceiros especializar e instanciar a ontologia livremente.

Mizoguchi (2003) e Gomes-Pérez (1999) fazem uma análise das principais ferramentas, linguagens e metodologias existentes para o desenvolvimento de ontologias. Existem diversas metodologias para o desenvolvimento de ontologias, como por exemplo: METHONTOLOGY (Corcho *et al.*, 2003), On-To-Knowledge (Staab *et al.*, 2001), Activity-First Method Mizoguchi (2003).

Ontolingua, RDF e OWL figuram entre as linguagens de representação de ontologias mais difundidas. Assim como, WebODE (Corcho *et al.*, 2003) e Protégé são algumas das ferramentas mais usadas para a construção de ontologias.

Protégé

Protégé (http://protege.stanford.edu/) é um editor de ontologias que possui uma interface gráfica de fácil utilização para o desenvolvimento de sistemas baseados em conhecimento. Protégé é feito em Java, tem código aberto e permite a criação, visualização e manipulação de ontologias em vários formatos de representação (RDF, OWL e XML Schema). Uma vantagem do Protégé é a existência de diversas extensões como por exemplo o componente que integra o Jess chamado JessTab. Este componente permite que o Jess manipule ontologias através do Protégé.

A.4 Redes de Petri

As Redes de Petri (RdP) surgiram da tese de doutorado de Carl Adam Petri, defendida em 1962, na Universidade de Darmstadt, Alemanha, cujo título era Comunicação com Autômatos. Uma RdP pode ser definida como uma ferramenta gráfica e matemática que se adapta bem a um grande número de aplicações em que as noções de eventos e de evoluções simultâneas são importantes (Cardoso e Valette, 1997).

As primeiras aplicações de RdP surgiram no projeto norte-americano intitulado Teoria dos Sistemas de Informação, da A.D.R. (*Applied Data Research, Inc.*), em 1968. Muito da teoria inicial, da notação e da representação de RdP foi desenvolvido neste projeto e foi publicado em seu relatório final. Este trabalho ressaltou como RdP poderiam ser aplicadas na análise e na modelagem de sistemas com componentes concorrentes.

Conforme Cardoso e Valette (1997) as vantagens da utilização da RdP podem ser resumidas pelas considerações seguintes:

- pode-se descrever uma ordem parcial entre vários eventos, o que possibilita levar-se em

conta a flexibilidade;

- os estados, bem como os eventos, são representados explicitamente;
- uma única família de ferramentas é utilizada através da especificação, da modelagem, da análise, da avaliação do desempenho e da implementação;
- uma única família de ferramentas é utilizada nos diversos níveis da estrutura hierárquica do controle, o que facilita a integração destes níveis;
- uma descrição precisa e formal das sincronizações torna-se possível, o que é essencial para alcançar-se a necessária segurança de funcionamento.

Uma RdP é composta pelos seguintes elementos:

- **Lugares**: representam uma condição, uma atividade ou um recurso.
- **Fichas, marcas ou *tokens***: representam o estado de um sistema.
- **Transições**: representam um evento.
- **Arcos**: indicam os lugares de entrada ou saída para as transições.

A Figura A.2 ilustra um exemplo de uma RdP e os seus respectivos elementos.

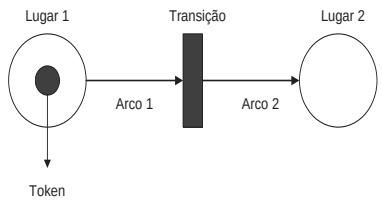

Figura A.2: Elementos de uma Rede de Petri.

Os arcos podem possuir pesos, ou seja, valores naturais positivos, e são sempre direcionados, ligando um lugar a uma transição ou uma transição a um lugar. Os nós lugares podem conter elementos chamados de fichas. Ao conjunto de fichas associadas aos lugares num dado momento dá-se o nome de marcação.

A função de uma transição é representar um evento, ou seja, a transição é responsável por modificar a marcação de uma Rede de Petri, ou seja, modificar o estado de uma Rede de Petri. Esta ocorrência só é válida quando existem tokens nos lugares de entrada de uma transição.

Portanto, para modificar o estado ou a marcação de uma RdP, basta disparar as transições existentes.

Uma transição é dita sensibilizada, quando cada lugar que é conectado à transição por um arco lugar-transição (lugares de entrada da transição) tem um número de fichas maior ou igual ao peso do arco. Transições que estão sensibilizadas podem disparar, causando a retirada das fichas dos lugares de entrada correspondentes ao peso do arco, e ao mesmo tempo inserindo fichas nos lugares de saída (os lugares ligados a transição via arcos transição-lugar), correspondendo também aos pesos dos arcos.

Uma variação das RdP são as RPO (Redes de Petri Objeto). As RPO são uma extensão das Redes de Petri, em que são usadas técnicas de modelagem orientadas a objeto, o que visa facilitar a modelagem de sistemas complexos (Lakos, 1995).

Apêndice B

Regras dos Agentes

Apresenta-se o código JESS do estudo de caso implementado (Seção 5.1.3)

B.1 Regras JESS: Agente Coordenador

```
;;; ======== Class Templates

(deftemplate Student
(slot busy
(default FALSE))
(slot answer)
(slot done (default 0))
(slot name)
(slot ID)
(multislot where) )

;;; ======== Place Template and Place Instances

(deftemplate place (slot name)
(slot type (default problem))
(multislot content))

(deffacts ped_places
(place (name CreateSupervisorAgent))
(place (name InviteToGroup))
(place (name GetAllAgents))
```

```
(place (name
CoordinatorManagerError)) )

;;; ======== Transitions Templates and Transitions Instances

(deftemplate trans-1to1
(slot name)
(slot place-in1)
(slot place-out1)
(slot condition)
(slot action))

(deffunction cond_t2err (?token)
(and (neq (fact-slot-value ?token answer) 0)
(neq (fact-slot-value ?token answer) nil)))

(deffunction act_t2err (?token)
(modify ?token (answer nil)))

(deffunction cond_t1 (?token)
(eq (fact-slot-value ?token answer) 0))

(deffunction act_t1 (?token)
(modify ?token (answer nil)))

(deffunction cond_t3err (?token)
(and (neq (fact-slot-value ?token answer) 0)
(neq (fact-slot-value ?token answer) nil)))

(deffunction act_t3err (?token)
(modify ?token (answer nil)))

(deffunction cond_t1err (?token)
(and (neq (fact-slot-value ?token answer) 0)
(neq (fact-slot-value ?token answer) nil)))

(deffunction act_t1err (?token)
(modify ?token (answer nil)))
```

```
(deffunction cond_t2 (?token)
(eq (fact-slot-value ?token answer) 0))

(deffunction act_t2 (?token)
(modify ?token (answer nil)))

(deffacts ped_transitions
(trans-1to1 (name t2err)
(place-in1 InviteToGroup)
(place-out1 CoordinatorManagerError)
(condition cond_t2err)
(action
act_t2err))

(trans-1to1 (name t1)
(place-in1 GetAllAgents)
(place-out1 InviteToGroup)
(condition cond_t1)
(action act_t1))

(trans-1to1 (name t3err)
(place-in1 CreateSupervisorAgent)
(place-out1 CoordinatorManagerError)
(condition cond_t3err)
(action act_t3err))

(trans-1to1 (name t1err)
(place-in1 GetAllAgents)
(place-out1 CoordinatorManagerError)
(condition cond_t1err)
(action act_t1err))

(trans-1to1 (name t2)
(place-in1 InviteToGroup)
(place-out1 CreateSupervisorAgent)
(condition cond_t2) (action act_t2))

)
```

```
;;; ======== Transitions Rules

(defrule rule-trans-1to1
;;token Student ?token <- (Student (where $?token-where))

;;Lugares de entrada
?in1 <- (place (name ?place-in1)
(content $?conts1&~nil))

;;Lugares de saída
?out1 <- (place (name ?place-out1))

;;a transição
(trans-1to1
(place-in1 ?place-in1)
(place-out1 ?place-out1)
(condition ?cnd)
(action ?act))

;;testa se o token está em todos os lugares acima
(test (member$ ?token $?conts1))

;;testa a condição especificada
(test (apply ?cnd ?token)) => (bind $?temp (create$ ?token))

;;cria lista com todos os lugares de entrada
(bind $?ins (create$ ?place-in1))

;;cria lista com todos os lugares de saída
(bind $?outs (create$ ?place-out1))

;;modifica o conteudo de ?in(x) para ficar com a lista de contents
(modify ?in1 (content (complement$ $?temp $?conts1)))

;;retira do where do token, o(s) lugar(es) de entrada
(modify ?token (where (complement$ $?ins
(fact-slot-value ?token where))))

;;coloca no where do token, o(s) lugar(es) de saída
```

```
(modify ?token (where (insert$
(fact-slot-value ?token where) 1 $?outs)))

;;pega o conteudo dos lugar(es) de saida
;;se o conteudo não existir, cria, senão,
insere o token no final da lista

(bind $?old_cont (fact-slot-value ?out1 content))
(if (or (eq $?old_cont nil)
(eq $?old_cont (create$ nil))) then
(modify ?out1
(content
$?temp)) else
(modify ?out1
(content (insert$ $?temp 2 $?old_cont))))

;;coloca o busy do token como FALSE,
para poder disparar (modify ?token (busy FALSE))

;;Aplica ação
(apply ?act ?token))

;;variavel global
;; objeto de comunicação com o mundo externo ao jess
(defglobal ?*jessComm* = null)

;;variavel global
;; bean de comunicação assincrona
(defglobal ?*jessCommBean* = null)

;; regra: se houver resposta do agente, coloca no token. ;;

(defrule agentAnswered ?f <- (answerFromAgent ?ans)

?token <- (Student (name

?name)(where $?where) (done ?done)) => (modify ?token (answer ?ans))

(retract ?f)
```

```
(modify ?token (where $?where))

)

;; regra - se um student estiver em um lugar, dispara um pedido para
;;agente externo

(defrule place ?in <- (place (name ?place-in)
(content $?conts & ~nil) )
?token <- (Student (where $?where)
(busy FALSE) (done ?done))

;;testa pra ver se o lugar pertence ao where do Student
(test (neq (member$ ?place-in $?where) FALSE))

=>

;;indica que não eh para disparar com esse
token de novo (modify ?token (busy TRUE))

(call ?*jessComm* tell ?place-in) (printout t ?place-in) )

;; adiciona novo token ;;

(defrule newToken ?f <- (newTokenFromAgent ?name)

?in <- (place (name GetAllAgents) ) =>

;;insere o token Student na base de dados

(bind ?t (assert (Student (name ?name)
(where (create$ GetAllAgents))  )))

(retract ?f)

;;insere no where do lugar (modify ?in (content (create$ ?t))) )

;;seta estratégia de disparo para fifo (set-strategy breadth)
```

B.2 Regras JESS: Agente Supervisor

```
;;; ======== Class Templates

(deftemplate Student
(slot groupTasks (default 1))
(slot busy (default FALSE))
(slot answer)
(slot done (default 0))
(slot name) (slot ID)
(multislot listStudents)
(multislot where) )

;;; ======== Place Template and Place Instances

(deftemplate place
(slot name)
(slot type (default problem))
(multislot content))

(deffacts ped_places
(place (name SupervisorManagerError))
(place (name FinishGroup))
(place (name ReceiveFromGroup))
(place (name DistributeToGroup)) )

;;; ======== Transitions Templates and Transitions Instances

(deftemplate trans-1to1
(slot name)
(slot place-in1)
(slot place-out1)
(slot condition)
(slot action))

(deffunction cond_t3 (?token)
```

```
(eq (fact-slot-value ?token answer) 0))

(deffunction act_t3 (?token)
(modify ?token (answer nil)))

(deffunction cond_t3err (?token)
(and (neq (fact-slot-value ?token answer) 0)
(neq (fact-slot-value ?token answer) nil)))

(deffunction act_t3err (?token)
(modify ?token (answer nil)))

(deffunction cond_t4err (?token)
(and (neq (fact-slot-value ?token answer) 0)
(neq (fact-slot-value ?token answer) nil)))

(deffunction act_t4err (?token)
(modify ?token (answer nil)))

(deffunction cond_t5 (?token)
(and (eq (fact-slot-value ?token answer) 0)
(>= (fact-slot-value ?token done) (fact-slot-value ?token
groupTasks))))

(deffunction act_t5 (?token) (modify ?token (answer nil)))

(deffunction cond_t4 (?token)
(and (eq (fact-slot-value ?token answer) 0)
(< (fact-slot-value ?token done) (fact-slot-value ?token
groupTasks))))

(deffunction act_t4 (?token) (modify ?token (answer nil)))

(deffacts ped_transitions
(trans-1to1 (name t3)
(place-in1 DistributeToGroup)
(place-out1 ReceiveFromGroup)
(condition cond_t3) (action act_t3))
```

```
(trans-1to1 (name t3err)
(place-in1 DistributeToGroup)
(place-out1 SupervisorManagerError)
(condition cond_t3err)
(action act_t3err))

(trans-1to1 (name t4err)
(place-in1 ReceiveFromGroup)
(place-out1 SupervisorManagerError)
(condition cond_t4err)
(action act_t4err))

(trans-1to1 (name t5)
(place-in1 ReceiveFromGroup)
(place-out1 FinishGroup)
(condition cond_t5)
(action act_t5))

(trans-1to1 (name t4)
(place-in1 ReceiveFromGroup)
(place-out1 DistributeToGroup)
(condition cond_t4)
(action act_t4))

)

;;; ======== Transitions Rules

(defrule rule-trans-1to1

;;token Student
?token <- (Student (where $?token-where))

;;Lugares de entrada
?in1 <- (place (name ?place-in1)
(content $?conts1&~nil))

;;Lugares de saída
?out1 <- (place (name ?place-out1))
```

```
;;a transição
(trans-1to1 (place-in1 ?place-in1)
(place-out1 ?place-out1)
(condition ?cnd) (action ?act))

;;testa se o token está em todos os lugares acima
(test (member$ ?token $?conts1))

;;testa a condição especificada
(test (apply ?cnd ?token)) => (bind $?temp (create$ ?token))

;;cria lista com todos os lugares de entrada
(bind $?ins (create$ ?place-in1))

;;cria lista com todos os lugares de saída
(bind $?outs (create$ ?place-out1))

;;modifica o conteudo de ?in(x) para ficar com a lista de contents,
menos o token q foi retirado agora
(modify ?in1 (content (complement$ $?temp
$?conts1)))

;;retira do where do token, o(s) lugar(es) de entrada
(modify ?token (where (complement$ $?ins (fact-slot-value ?token where))))

;;coloca no where do token, o(s) lugar(es) de saída
(modify ?token (where (insert$ (fact-slot-value ?token where) 1 $?outs)))

;;pega o conteudo dos lugar(es) de saida
;;se o conteudo não existir, cria, senão, insere o token no final da lista
(bind $?old_cont
(fact-slot-value ?out1 content))
(if (or (eq $?old_cont nil)
(eq $?old_cont (create$ nil))) then
   (modify ?out1 (content $?temp))
else
   (modify ?out1 (content (insert$ $?temp 2 $?old_cont))))
```

```
;;coloca o busy do token como FALSE, para poder disparar
(modify ?token (busy FALSE))

;;Aplica ação (apply ?act ?token))

;;variavel global ;;objeto de comunicação com o mundo externo ao jess
(defglobal ?*jessComm* = null)

;;variavel global - bean de comunicação assincrona
(defglobal ?*jessCommBean* = null)

;; regra: se houver resposta do agente, coloca no token. ;;
(defrule agentAnswered ?f <- (answerFromAgent ?ans)
?token <- (Student (name ?name)
(where $?where) (done ?done)) =>
(modify ?token (answer ?ans))
(retract ?f)

;;se o token estiver no lugar de receber respostas de tarefas,
;;incrementa o campo done
(if (neq (member$ ReceiveFromGroup $?where) FALSE) then
   (modify ?token (done (++ ?done)))
)

(modify ?token (where $?where))

)

 ;; regra - se um student estiver em um lugar, dispara um pedido para
;;agente externo
;; (defrule place ?in <- (place (name ?place-in)
(content $?conts & ~nil) ) ?token <-
(Student (where $?where) (busy FALSE)
(done ?done))

;;testa pra ver se o lugar pertence ao where do Student
(test (neq (member$ ?place-in $?where) FALSE))

=>
```

```
;;indica que não eh para disparar com esse token de novo (modify ?token (busy TRUE))

;;se for lugar de distribuição ou de recebimento de tarefas, tem anexar ao nome
;;o nro de tarefas prontas

(if (or (eq ?place-in DistributeToGroup) (eq ?place-in ReceiveFromGroup)) then
   (call ?*jessComm* tell (str-cat ?place-in ?done))
   (printout t (str-cat ?place-in ?done))
else
   (call ?*jessComm* tell ?place-in)
   (printout t ?place-in)
) )

;; adiciona novo token ;;

(defrule newToken
?f <- (newTokenFromAgent ?name ?gTasks $?lStudents)
?in <- (place (name DistributeToGroup) ) =>

;;insere o token Student na base de dados
(bind ?t (assert (Student (name ?name)
(where (create$ DistributeToGroup))
(groupTasks
?gTasks)(listStudents $?lStudents) )))

(retract ?f)

;;insere no where do lugar (modify ?in (content (create$ ?t))) )

;;seta estrategia de disparo para fifo
(set-strategy breadth)
```

B.3 Coordenador-OPN

```
PETRINET

/* Classe Student, será o token da rede */
```

```
Class := tokenclass: Student:
  (ID)            //"chave primaria" do Student
  (name)
  (busy FALSE)      //indica que está fazendo algo, e esperando pela resposta
  (multifield where)
  (done 0)          //tarefas já realizadas
  (answer)          //ultima resposta
  ;


/* Lugares */

Places :=
GetAllAgents: (Student);
InviteToGroup: (Student);
CreateSupervisorAgent: (Student);
CoordinatorManagerError: (Student);

/* Transições */ Structure :=

/* transições "normais" */
t1: (GetAllAgents (Student) )
     -> (InviteToGroup (Student) );
t2: (InviteToGroup (Student) )
     -> (CreateSupervisorAgent (Student) );

/* transições que podem ser "erro" */

t1err: (GetAllAgents (Student) )
     -> (CoordinatorManagerError (Student) );
t2err: (InviteToGroup (Student) )
     -> (CoordinatorManagerError (Student) );
t3err: (CreateSupervisorAgent (Student) )
     -> (CoordinatorManagerError (Student) );

/* condições para a rede funcionar */

Conditions :=
```

```
t1: eq (Student.answer, 0);
t2: eq (Student.answer, 0);

t1err: and
(neq (Student.answer, 0),
neq (Student.answer, nil));

t2err: and
(neq (Student.answer, 0),
neq (Student.answer, nil));

t3err: and
(neq (Student.answer, 0),
neq (Student.answer, nil));



/*
   Ações a serem tomadas - assim como as condições alteram a dinâmica da RdP
   fazendo com que transições sejam desabilitadas/habilitadas, as ações tem
   importância para controlar ações que vão além da semântica da RdP
*/
Actions :=

t1: Student.answer := nil;
t2: Student.answer := nil;

t1err: Student.answer := nil;
t2err: Student.answer := nil;
t3err: Student.answer := nil;

FunctionFile := "coordinator-opn-functions.txt"

ENDNET
```

B.4 Supervisor-OPN

```
;;variavel global - objeto de comunicação com o mundo externo ao jess
(defglobal ?*jessComm* = null)

;;variavel global - bean de comunicação assincrona
(defglobal ?*jessCommBean* = null)

;; regra: se houver resposta do agente, coloca no token. ;;
(defrule agentAnswered
?f <-(answerFromAgent ?ans)
?token <- (Student (name ?name)
(where $?where) (done ?done)) =>
(modify ?token (answer ?ans))
(retract ?f)

;;se o token estiver no lugar de receber respostas de tarefas,
;;incrementa o campo done
(if (neq (member$ ReceiveFromGroup $?where) FALSE) then
   (modify ?token (done (++ ?done)))
)

(modify ?token (where $?where)) )

;; regra - se um student estiver em um lugar, dispara um pedido para
;;agente externo ;;
(defrule place ?in <-
(place (name ?place-in)
(content
$?conts & ~nil) )
?token <- (Student (where $?where)
(busy FALSE) (done ?done))

;;testa pra ver se o lugar pertence ao where do Student
(test (neq (member$ ?place-in $?where) FALSE))

=>

;;indica que não eh para disparar com esse token de novo
(modify ?token (busy TRUE))
```

```
;;se for lugar de distribuição ou de recebimento de tarefas, tem anexar ao nome
;;o nro de tarefas prontas
(if (or (eq ?place-in
DistributeToGroup)
(eq ?place-in ReceiveFromGroup)) then
   (call ?*jessComm* tell (str-cat ?place-in ?done))
   (printout t (str-cat ?place-in ?done))
else
   (call ?*jessComm* tell ?place-in)
   (printout t ?place-in)
) )

;; adiciona novo token ;;
(defrule newToken ?f <- (newTokenFromAgent ?name ?gTasks $?lStudents)
?in <- (place (name DistributeToGroup) ) =>

;;insere o token Student na base de dados
(bind ?t (assert (Student (name ?name)
(where (create$ DistributeToGroup))
(groupTasks ?gTasks)
(listStudents $?lStudents) )))

(retract ?f)

;;insere no where do lugar (modify ?in (content (create$ ?t)))

)
```

Apêndice C

Interações do Estudante com o STI

O estudante interage com o STI em seis situações: a primeira quando o estudante entra no sistema; a segunda quando ocorre a formação do grupo; a terceira quando o estudante interage com outros estudantes dos grupos; a quarta quando o estudante interage com os elementos do tutorial; a quinta quando o estudante executa e responde a tarefa de grupo; e a sexta é quando todos os estudantes do grupo terminam suas tarefas.

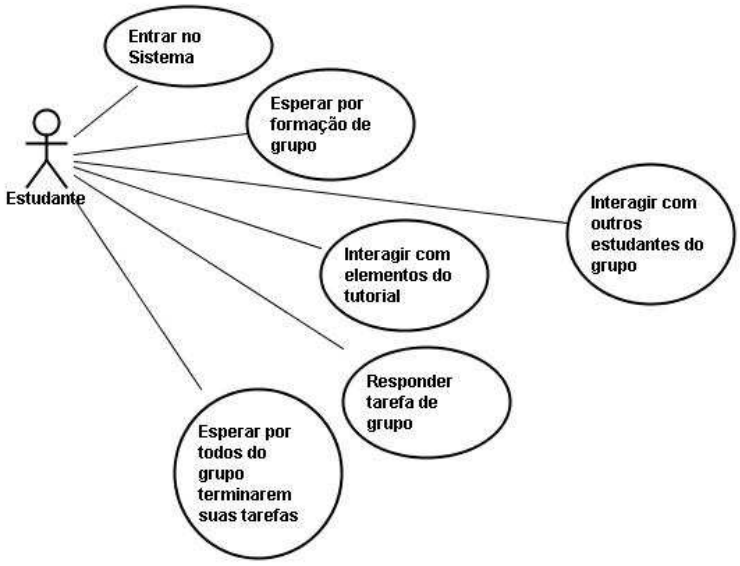

Figura C.1: Interações do estudante com o STI

A seguir, apresentamos os gráficos das interações do Estudante com o STI do cenário Dividir para Conquistar.

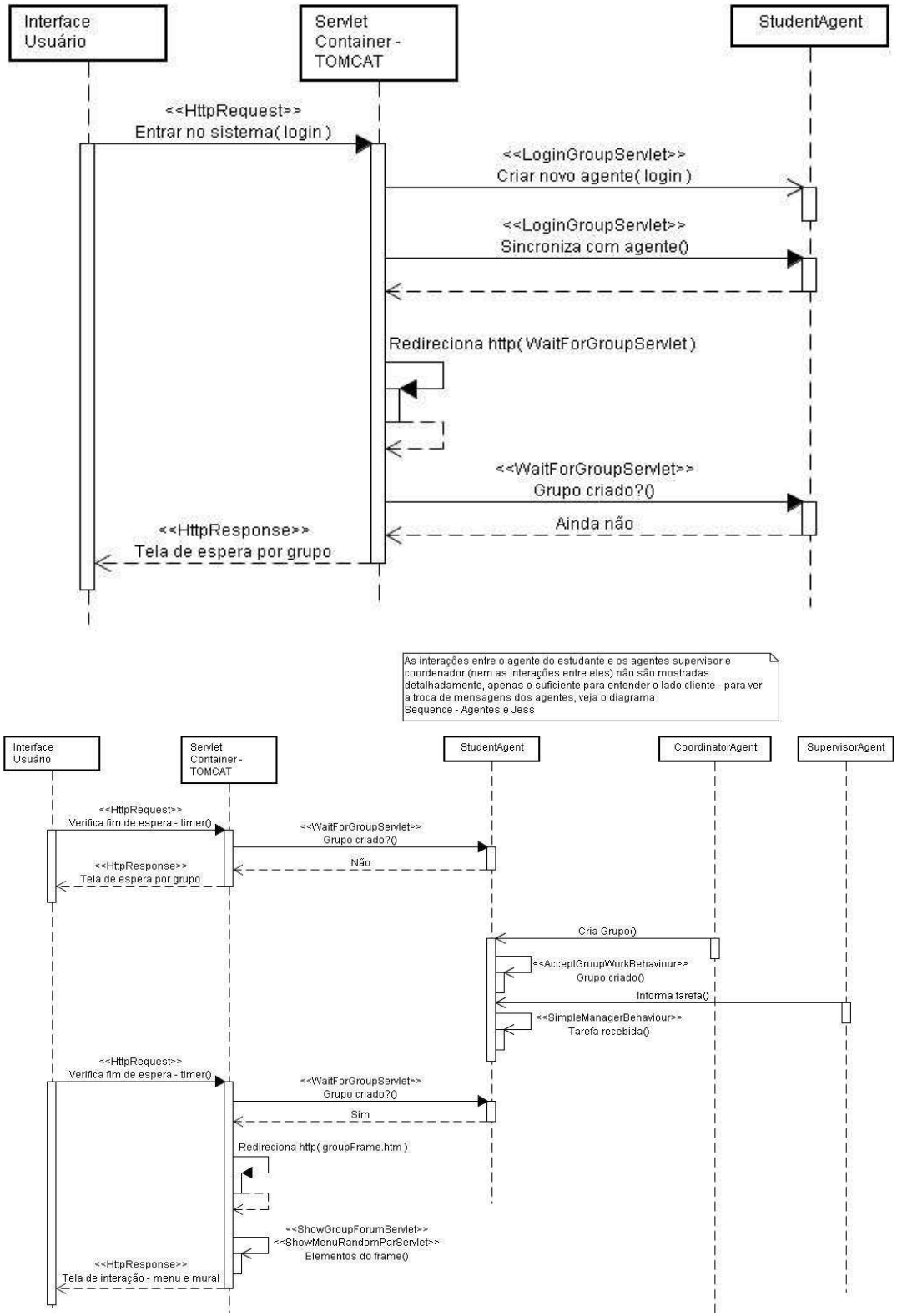

Figura C.2: Diagramas: Entrada e formação de grupos.

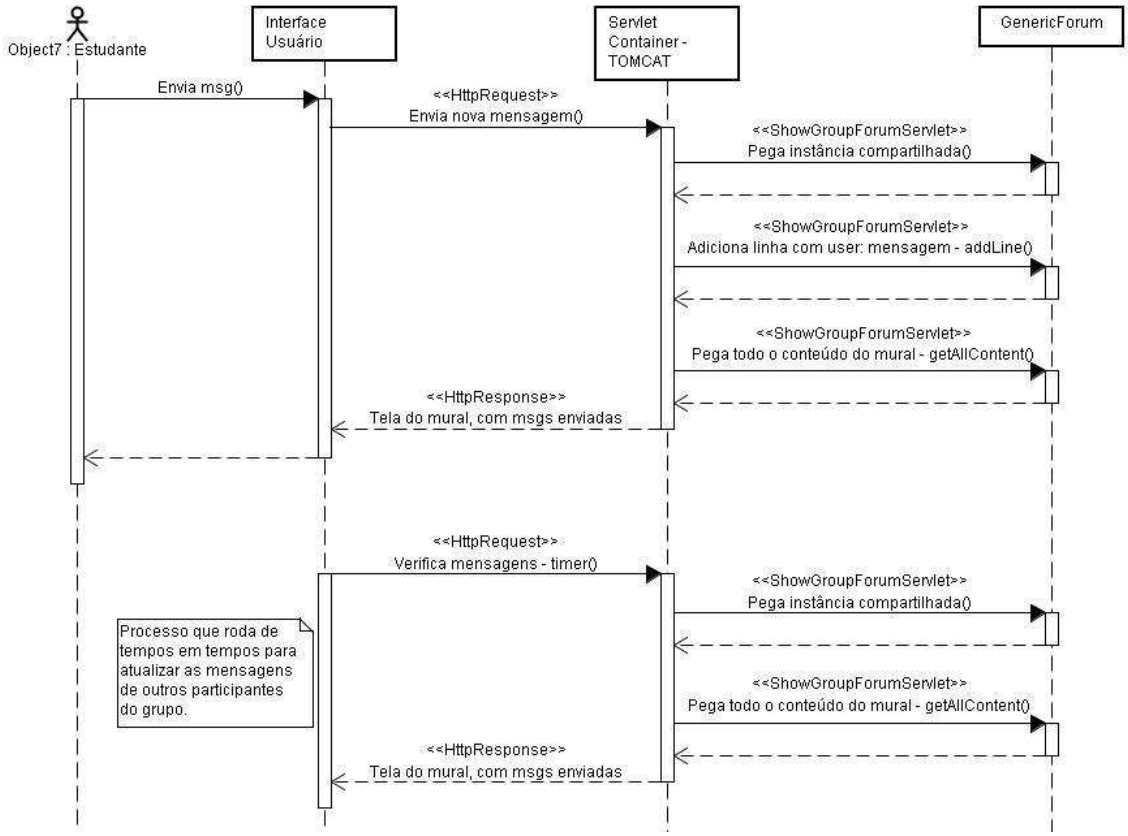

Figura C.3: Diagrama de Sequência: Interagir com outros estudantes do grupo.

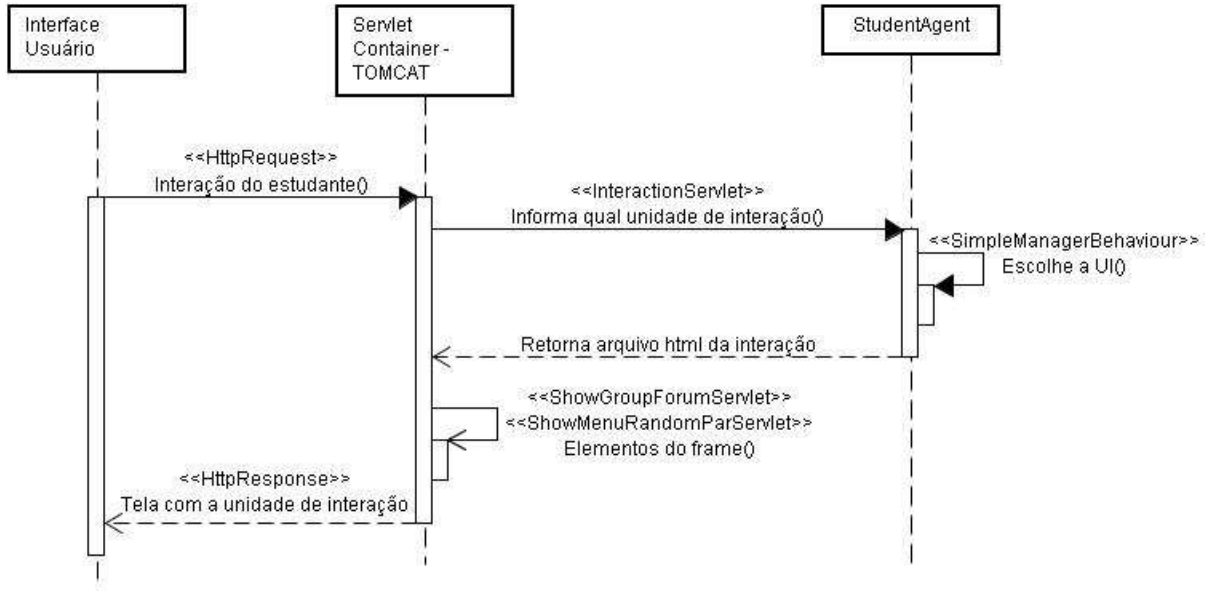

Figura C.4: Diagrama de Sequência: Interagir com elementos do tutorial.

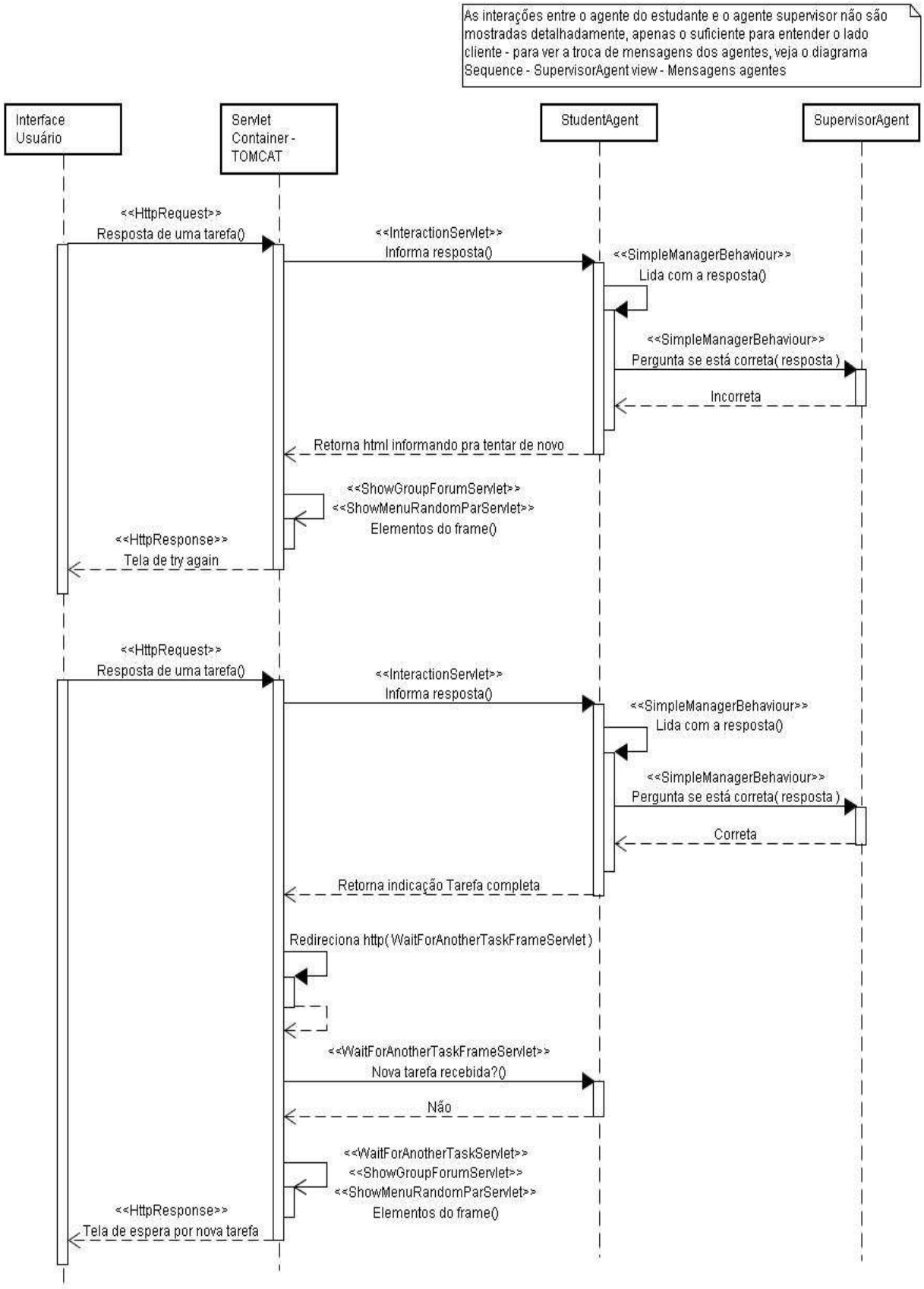

Figura C.5: Diagrama de Sequência: Responder tarefa de grupo.

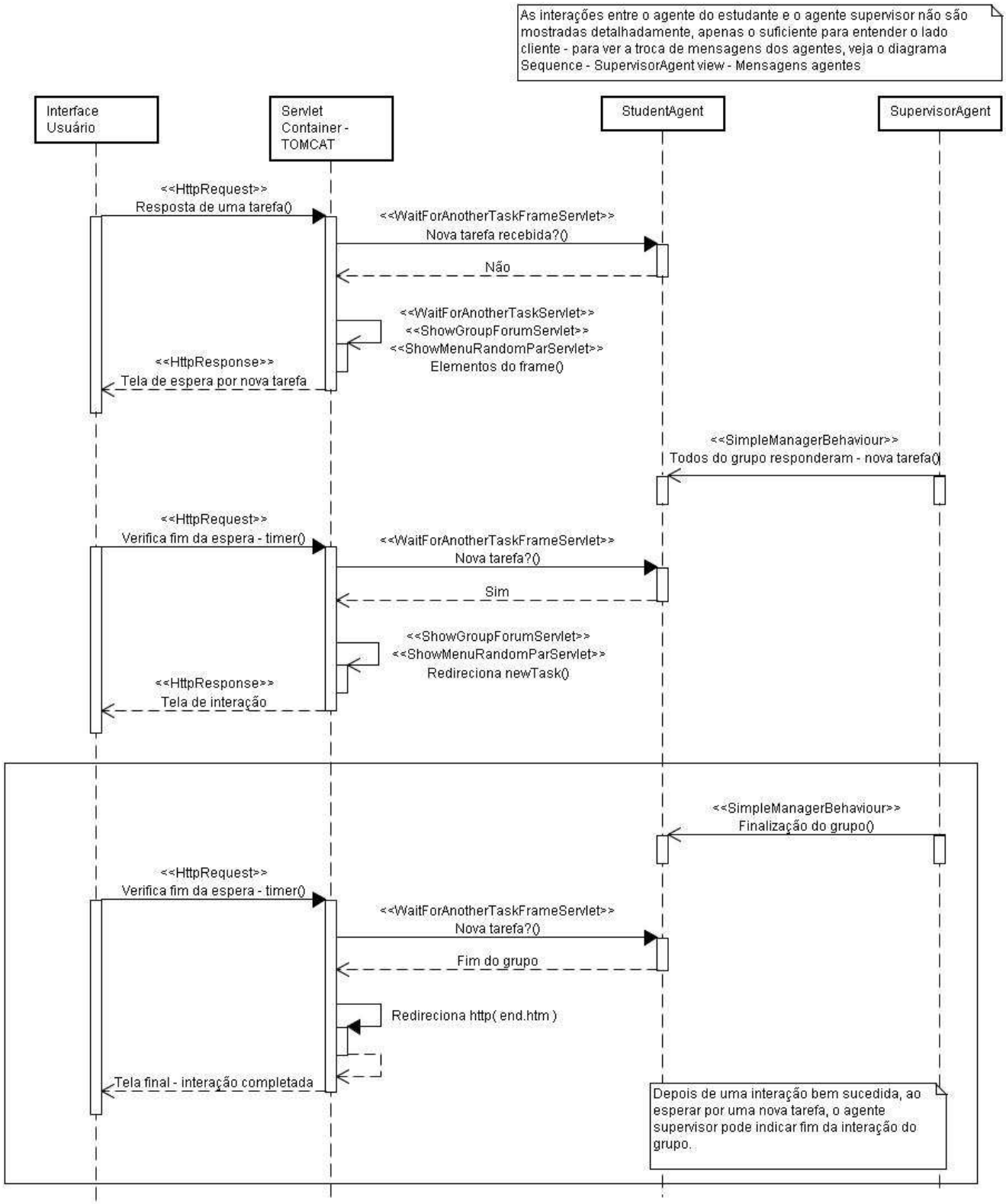

Figura C.6: Diagrama de Sequência: Esperar por todos do grupo terminarem suas tarefa.

Apêndice D

Compilador para transformar grafo de pré-requisitos em regras

Como visto no funcionamento da FAST (Seção 2.5) à partir do grafo de pré-requisitos é gerado base de conhecimento no JESS.

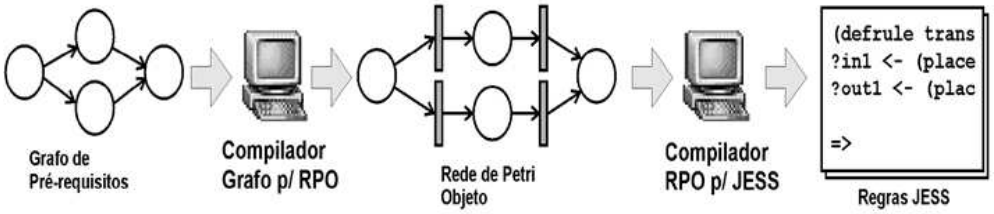

Figura D.1: Compilador. Extraída de (Frigo, 2007).

Como ilustra a Figura D.1 o primeiro passo é transformar o grafo de pré-requisitos numa RPO e posteriormente da RPO para as regras no JESS.

D.0.1 Compilador Grafo - Rede de Petri Objetos

O compilador recebe como entrada um grafo na forma textual. Na representação descrita no fragmento de Código C.1 os nós representam os problemas extraídos da Figura 2.6 (Frigo, 2007).

Codigo D.1 `Grafo`

```
PREREQGRPH
   UP2Pb1 := [UP2Pb0],
   UP2Pb2 := [UP2Pb0],
   UP2Pb3 := [UP2Pb1,UP2Pb2],
   UP2Pb4 := [UP2Pb3]
ENDGRAPH
```

O fragmento de código C.1 mostra que:

- um nó n possui dois ou mais arcos de entrada necessários, tem-se n := [(n1,n2, ...)] como UP2Pb3;
- um nó n possui dois ou mais arcos de entrada alternativos, tem-se n := [(n1), (n2), ...)].

A partir do grafo de pré-requisitos o compilador gera uma Rede de Petri na forma textual.

As fichas são formadas por objetos de dados que apontam para o modelo do estudante e para o modelo de domínio. No fragmento de Código C.2 são apresentadas as informações dinâmicas do modelo do estudante e que fazem parte do conhecimento local do agente. As informações estáticas pertencem ao conhecimento social dos agentes e são obtidas a partir de formulários e questionários a serem preenchidos pelo próprio estudante. O campo *answer* contém a informação da interação do estudante com a interface que pode ser do tipo: *halt* indicando que ele quer se desconectar do sistema ou ainda exercício, exemplo ou explanação que indica que ele gostaria de realizar alguma destas unidades de interação.

Codigo D.2 `Rede de Petri - Estudante`

```
Class := tokenclass: Student:
  (ID)
  (name)
  (doing Curriculum)
  (doing Pb)
  (doing IU)
  (multifield where)
  (multifield done)
  (multifield report)
  (answer)
  (bestgrade);
```

Codigo D.3 `Rede de Petri - Lugar`

```
Places := GhostPlace: (Student);
  UP2Pb1: (Student);
  UP2Pb4: (Student);
  UP2Pb0: (Student);
  UP2Pb2: (Student);
  UP2Pb3: (Student);
  bufUP2Pb1UP2Pb2: (Student);
  bufUP2Pb1bufUP2Pb1UP2Pb2: (Student);
  bufUP2Pb2bufUP2Pb1UP2Pb2: (Student);
```

Os lugares (fragmento de Código C.3) correspondem aos problemas que constituem as Unidades Pedagógicas. O lugar *GhostPlace* é usado para enviar as fichas dos estudantes que já terminaram a execução na rede, mas que continuam num lugar da rede. Por exemplo, depois de passar por um OR, uma ficha pode ter ido parar no final, mas outra pode não ter sido disparada, e ter continuado num lugar antes do OR. Os lugares que iniciam o nome com buf representam *buffers*.

Codigo D.4 `Rede de Petri - Estrutura`

```
  Structure :=
  tUP2Pb3UP2Pb4: (UP2Pb3 (Student) ) - > (UP2Pb4 (Student) );
```

Codigo D.5 `Rede de Petri - Condição`

```
Condition :=
  tUP2Pb3UP2Pb4: neq (member(UP2Pb3, Student.done), FALSE);
```

Codigo D.6 `Rede de Petri - Ação`

```
Actions :=
  tUP2Pb3UP2Pb4: Student.doing Pb := Next Problem(Student);
```

A estrutura da rede apresenta as transições que indicam, como no exemplo do fragmento de Código C.6, os pré-requisitos para os problemas pertencentes a unidade pedagógica. As transições da rede são controladas por condições lógicas (fragmento de Código C.5) que fazem referência ao modelo do estudante e o disparo destas transições produzem ações (fragmento de Código C.6) que atualizam o modelo do estudante. Como agora o lugar possui a classe Student, a variável *x* associada aos arcos pode instanciar à ficha que possui também a classe *Student*. Então é necessário adicionar a cada transição condições que fazem intervir a variável formal x (associada aos arcos de entrada) e os atributos da classe Student associada a ficha.

D.0.2 Tradutor da RPO em JESS

Redes de Petri Objeto são ferramentas matemáticas utilizadas freqüentemente para modelagem de sistemas. Entretanto, para a construção do Sistema Tutor Inteligente, deve-se utilizar uma ferramenta que execute o modelo em RPO. As ferramentas existentes são em sua grande maioria focadas na modelagem, análise e simulação de Redes de Petri. Para execução de uma RPO, as ferramentas são mais escassas (Yamane, 2006).

Para executar as RPOs, optou-se por utilizar um shell de Sistemas Especialistas baseado em regras, regras estas que implementam uma RPO. Além disso, o uso de um sistema de regras traz a possibilidade de se adicionar novas capacidades/habilidades, por meio de regras específicas (não relacionadas diretamente com a RPO). A opção de tal shell recaiu sobre o JESS, devido sobretudo a sua forte integração com Java.

Primeiramente, definiu-se a estrutura da RPO na base de fatos e regras do JESS: cada lugar da rede corresponde a um fato do template (tipo de fato em JESS) lugar; cada transição da rede corresponde a um fato do template de um tipo de transição (por tipo de transição, entenda-se quantos lugares de entrada e quantos lugares de saída uma transição possui); os arcos são implementados como propriedades dos fatos de transição; as condições de disparo, bem como as ações são funções ligadas a propriedades dos fatos de transição.

Em seguida, utilizando-se a ferramenta JavaCC, foi construída uma classe de *parser* (análise sintática) para a gramática que descreve uma RPO. Em conjunto com o *parser*, um conjunto de classes representando as estruturas da RPO foram desenvolvidas. Estas classes contém métodos que geram códigos em JESS que implementam a RPO.

Em termos de implementação, as regras do JESS que correspondem a estrutura da RPO é obtida da seguinte maneira:

- Cada lugar *UPiPb j* da rede corresponde a um fato *UPiPb j* em JESS (template) do tipo *PLACE*;
- Cada transição *tUPiPbjUPyPbz* da rede corresponde a um fato *tUPiPbjUPyPbz* tipo *TRANSITION*;

  Arcos são implementados como propriedades dos fatos tipo *TRANSITION*;

As condições de disparo e as ações são funções relacionadas com as *TRANSITION*.

- Cada ficha *S* da rede corresponde a um fato *S* do tipo *STUDENT*;

A definição dos fatos *PLACE* correspondente a um lugar da RPO é no fragmento de Código C.7. Supondo que a rede modela um domínio relacionado à matemática possíveis fatos são apresentados.

Segundo o número de lugares de entrada e de saída de uma transição, um *template* diferente é gerado. Para uma transição simples (um lugar de entrada, um lugar de saída), o *template* gerado é descrito no fragmento de Código C.8. Os fatos gerados a partir deste template são exemplificados pela transição tUP2-Pb0UP2-Pb1.

Codigo D.7 `Place - Lugar`

```
(deftemplate place
   (slot name)
   (slot type (default problem))
   (multislot content))
   (N1
   (name UP2Pb0)
   (type pedagogicalUnit)
   (content (Soma, Adição, Exponenciação)))
   (N2
   (name UP2Pb1)
   (type problem)
   (content (Exponenciação)))
```

Codigo D.8 `Trans - Transição`

```
((deftemplate trans-1to1
   (slot name)
   (slot place-in1)
   (slot place-out1)
   (slot condition)
   (slot action))
   (trans-1to1
   (name tUP2Pb0UP2Pb1)
   (place-in1 UP2Pb0)
   (place-out1 UP2Pb1)
   (condition cond tUP2Pb0UP2Pb
   (action act tUP2Pb0UP2Pb1))
```

Referências Bibliográficas

Ainsworth, S. (2000). Redeem: Its authoring tools and human teaching. *Proceedings of the 5th International Conference ITS 2000 Workshop on Modeling Human Teaching Tactics and Strategies, Montreal*, Vol. 1, No. 1, pp. 182–191.

Ainsworth, S. (2007). Using a single authoring environment across the lifespan of learning. *Educational Technology and Society,*, Vol. 10, No. 3, pp. 22–31.

Ainsworth, S. e Fleming, P. (2005). Evaluating a mixed-initiative authoring environment: Is redeem for real? *Proceedings of the 12th International Conference on Artificial Intelligence in Education*, Vol. 1, No. 1, pp. 9–16.

Alcântara, P. R., Siqueira, L. M. M., e Valaszi, S. (2004). Vivenciando a aprendizagem colaborativa em sala de aula: experiências no ensino superior. *Revista Diálogo Educacional*, Vol. 4, No. 12, .

Aleven, V., McLaren, B. M., Sewall, J., e Koedinger, K. R. (2006). The cognitive tutor authoring tools (ctat): Preliminary evaluation of efficiency gains. *Proceedings 8th International Conference Intelligent Tutoring System*, Vol. LNCS 4053, No. ISSN 0302-9743, pp. 61–70.

Andrade, A. F. d., Jaques, P. A., Jung, J. L., e Bordini, Rafael H.and Vicari, R. M. (2001). A computacional model of distance learning based on vygotsky socio-cultural approach. *X International Conference on Artificial Intelligence on Education*, Vol. 1, No. 1, pp. 33–40. San Antonio-Texas 19-23 May 2001.

Antunes, C. (1987). *Manual de Técnicas de Dinâmica de Grupo de Sensibilização de Ludopedagogia*, Vol. 1 of *1*. Editora Vozes, RJ, Brasil, 16 edição.

Aroyo, L., Inaba, A., Soldatova, L., Mizoguchi, R., Lester, J., Vicari, R. M., e Paraguaçu, F. (2004). Ease: Evolutional authoring support environment. *Intelligent tutoring systems, Lecture notes in computer science*, Vol. 1, No. 1, pp. 140–149.

Arriada, M. e Ramos, E. M. F. (2000). Como promover condições favoráveis à aprendizagem cooperativa suportada por computador? *Anais do V Congresso Ibero Americano de Informática Educativa (RIBIE 2000)*, Vol. 3, No. 1, pp. 146–159. Chile.

Avouris, N., Margaritis, M., e Komis, V. (2004). Modelling interactions during small-goups synchronous problem-solving activities: The synergo approach. *ITS 2004 Whorkshop on Computational Models of Collaborative Learning*, Vol. 1, No. 1, pp. 13–18. Alagoas.

Azevedo, H. J. S. d. e Scalabrin, E. E. (2005). *Designing Distributed Learning Enviroment with Intelligent Software Agents*, Vol. 1, chapter A Humam Collaborative Online Learning Enviroment Using Intelligent Agents, pp. 1–32. Informatiom Science Publishing.

Barros, B. e M. Verdejo, F. (2000). Analysing student interaction processes in order to improve collaboration. the degree approach. *International Journal of Artificial Intelligence in Education*, Vol. 11, No. 1, pp. 221–241.

Bittencourt, G. (1998). *Inteligência Artificial: Ferramentas e Teoria*, Vol. 2. Editora UFSC.

Bond, A. e Gasser, L. (1988). *An Analysis of Problems and Research in DAI. Readings in Distributed Artificial Intelligence*, Vol. 1. Morgan Kaufmann.

Cardoso, J., Bittencourt, G., Frigo, L., e Pozzebon, E. (2004a). Mathtutor: A multi-agent intelligent tutoring systems. *IFIP 18Th World Computer Congress, International Conference on Artificial Intelligence Applications and Innovations, Toulouse, França*, Vol. TC12, No. 1, pp. 231–241.

Cardoso, J., Bittencourt, G., Frigo, L., e Pozzebon, E. (2004b). Petri nets for authoring mechanisms. *XV Simpósio Brasileiro de Informática na Educação (SBIE'2004)*, Vol. 1, No. 1, pp. 378–387.

Cardoso, J. e Valette, R. (1997). *Redes de Petri*, Vol. 1. Editora da UFSC, Florianópolis, SC.

Chen, W. e Mizoguchi, R. (2004). Leaner model ontology and leaner model agent. *Cognitive Support for Learning*, Vol. IOS Presspp. 189–200.

Conati, C., McCoy, K. F., e Paliouras, G., editors (2007). *User Modeling 2007, 11th International Conference, UM 2007, Corfu, Greece, June 25-29, 2007, Proceedings*, Vol. 4511 of *Lecture Notes in Computer Science*. Springer.

Constantino-Gonzalez, M. d. l. A., Suthers, D. D., e Santos, J. G. E. d. l. (2003). Coaching web-based collaborative learning based on problem solution differences and participation. *International Journal of Artificial Intelligence in Education*, Vol. Volume 13, Numbers 2-4, No. 1, pp. 263 – 299.

Corcho, M., Fernandez-Lopez, A., e Lopez-Cima, A. (2003). Building legal ontologies with methontology web ode. *Law and the Semantic Web*, Vol. 3369, No. 1, .

Costa, E. d. B. (1997). *Um Modelo de Ambiente Interativo de Aprendizagem Baseado numa Arquitetura Multi-Agentes*. Tese de doutorado, Pós-graduação em Engenharia Elétrica da Universidade Federal da Paraíba, Campina Grande, Paraíba, Brasil.

de Évora, U. (2008). Aprendizagem colaborativa assistida por computador: aproximação ao conceito. *Portugal: Universidade de Évora*, Vol. 1. Disponível em: http://minerva.uevora.pt/cscl. Acessado em 10 de agosto de 2008.

Deitel, H. e Deitel, P. (2006). *Java como Programar*, Vol. 1. Pearson, 6a edição.

Demazeau, Yves; Müller, J.-P. (1990). Descentralized artificial intelligence. *North-Holland: Elselvier Science Publishers, 1990 Trabalho apresentado no European Workshop on Modelling Autonomous Agents in a Multi-Agent World, 1, 1989, Cambridge*, Vol. 1pp. 3–13.

Desmoulins, C. e Labeke, N. V. (1996). Towards student modelling in geometry with inductive logic programming. *Proceedings of the European Conference on Artificial Intelligence in Education.*, Vol. 1pp. 94–100.

Dillenbourg, P. (1999). *Collaborative-learning: Cognitive and Computational Approaches*, chapter What do to mean by Collaborative Learning?, pp. 1–19. Oxford: Elsevier.

Ellis, C., Gibbs, S., e Rein, G. (1991). Groupware: Some issues and experiences. *Communications of the ACM*, Vol. 34, No. 1, pp. 38–58.

Ferber, J., G. O.-M. F. (2003). From agents to organizations: an organizational view of multi-agent systems. *Lecture Notes in Computer Science, Springer Berlin / Heidelberg*, Vol. Volume 2935, No. 1, pp. 214–230. Melbourne, Australia.

Ferber, J. (1999). *Multi-Agent Systems*, Vol. 1. Addison-Wesley Professional.

Frigo, L. (2007). *Um modelo para Autoria de Sistemas Tutores Adaptativos*. tese de doutorado, Curso de Pós Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.

Gallagher, S., Rosenthal, H., e Stepien, W. (1992). "the effects of problem-based learning on problem solving. *Gifted Child Quarterly*, Vol. 36, No. 4, pp. 195–200.

Giraffa, L. (1999). *Uma arquitetura de tutor utilizando estados mentais*. Tese de Doutorado, Programa de Pós-graduação em Ciência da Computação da Universidade Federal Rio Grande do Sul.

Giraffa, L. M. M. (1995). Fundamentos de teorias de ensino-aprendizagem e sua aplicação em sistemas tutores inteligentes. *CPGCC/UFRGS Porto Alegre*, Vol. 487.

Gomes-Pérez, A. (1999). Ontological engineering: A state of the art. *British Computer Society*, Vol. 2, No. 1, pp. 33–44.

Gruber, T. (1993). A translation approach to portable ontologuy specifications. *Knowledge Acquisition*, Vol. 5, No. 1, pp. 199–220.

Guarino, N. (1997). Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. Em Pazienza, M., editor, *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology*, Vol. 1, pp. 139–170, Frascati, Italy. International Summer School, SCIE-97.

Guarino, N. (1998). Formal ontology and information systems. Proc. *Formal Ontology in Information Systems*, Vol. 1, pp. 3–15, Amsterdam, Netherlands.

Harasim, L., Teles, L., Turoff, M., e Hiltz, S. (2005). *Redes de Aprendizagem, um guia para ensino e aprendizagem online*, Vol. 1. Editora Senac São Paulo.

Hoppe, H. U. (1995). Using multiple student modeling to parameterize group learning. *World Conference on Artificial Intelligence in Education (AI-ED 95), AACE, Charlottesville VA.*, Vol. 1, No. 1, pp. 234–241.

Inaba, A., Ohkubo, R., Ikeda, M., e Mizoguchi, R. (2003). Models and vocabulary to represent learner-to-learner interaction process in collaborative learning. *Proceedings of the International Conference on Computers in Education*, Vol. 1, No. 1, pp. 1088–1096.

Inaba, A., Supnithi, T., Ikeda, M., Mizoguchi, R., e Toyoda, J. (2000). How can we form effective collaborative learning groups? Vol. 1, No. 1, pp. 282–291.

Isotani, S. e Mizoguchi, R. (2007). Deployment of ontologies for an effective design of collaborative learning scenarios. *CRIWG - International Workshop on Groupware,Lecture Notes in Computer Science. Berlin Heidelberg, Springer-Verlag*, Vol. 4715, No. 1, pp. 223–238.

Isotani, S. e Mizoguchi, R. (2008a). Adventures in the boundary between domain-independent ontologies and domain content for cscl. *International Conference on Knowledge-Based and Intelligent Information Engineering Systems (KES) Zagreb. Lecture Notes in Artificial Intelligence. Berlin Heidelberg : Springer-Verlag*, Vol. 5179, No. 1, pp. 523–532.

Isotani, S. e Mizoguchi, R. (2008b). An ontology-based framework and its application to effective collaboration. *CLEI Electronic Journal*, Vol. 1, No. 11, pp. 1–9.

Kaliannan, J. (1999). A software plataform to enable multi domain collaborative applications. Dissertação de Mestrado, Master of Science in Computer Science in University of Lowa.

Kerly, A., Ellis, R., e Bull, S. (2008). Calmsystem: A conversational agent for learner modelling. *Knowledge Based Systems, The 27th SGAI International Conference on Artificial Intelligence*, Vol. 21, No. 3, pp. 238–246.

Kuyven, N. L. (2002). Proposta de metodologia de desenvolvimento de um sistema tutor inteligente baseado nas teorias de aprendizagem. Dissertação de mestrado do curso de pos-graduação de ciência da computação da universidade federal de santa catarina, UFSC, Florianópolis-SC.

Labidi, S., Costa, Q. C., e Jaques, P. (2006). Inferring socio-affective factors and cooperation capacity in computer assisted collaborative teaching/learning environments. *The 6th IEEE International Conference on Advanced Learning Technologies, Advanced Technologies for Life-Long Learning, Danvers, MA, USA*, Vol. 60, No. 1, pp. 608–612.

Lakos, C. (1995). The object orientation of object petri nets. Proc. *In Proceedings of the first international workshop on Object-Oriented Programming and Models of Concurrency*, Vol. 1 of *16th International Conference on Application and Theory of Petri Nets*, pp. 1–14.

Langley, P. (1999). User modeling in adapative interfaces. Em Banff, Canada, S., editor, *Proceedings of the Seventh International Conference on User Modeling*, Vol. 1, pp. 357–370.

Lin, F. O. (2005). *Designing Distributed Learning Environments with Intelligent Software Agents*. Information Science Publishing.

Mabbott, A. e Bull, S. (2006). Student preferences for editing, persuading, and negotiating the open learner model. *Intelligent Tutoring Systems, Lecture Notes in Computer Science*, Vol. 4053pp. 481–490.

McCalla, G. (2000). The fragmentation od culture, learning, teaching and technology -implication for the artificial intelligence in education research agenda in 2010. *International Journal of Artificial Intelligence in Education*, Vol. 11pp. 177–196.

Mitrovic, A., Suraweera, P., Martin, B., e Zakharov, K. (2006). Authoring constraint based tutors in aspire. *The 8th International Conference on Intelligent Tutoring Systems*, Vol. 1, No. 1, pp. 41–50.

Mizoguchi, R. (2003). Tutorial on ontological engineering ontology development. *Tools and Languages New Generation Comput*, Vol. 22, No. 1, .

Mizoguchi, R. e Bourdeau, J. (2000). Using ontological engineering to overcome common aied problems. *AI in Education*, Vol. 1pp. 107–121.

Mohamed, A. (2005). Intelligent tutoring system for distributed learning. *in Designing Distributed Learning Environments with Intelligent Software Agents, editor Fuhua Oscar Lin*, Vol. 1, No. 1, pp. 162–181.

Moulin, B. e Chaib-Draa, B. (1996). *An Overview of Distributed Artificial Intelligence*, Vol. 3-47. Fundations of Distributed Artificial Intelligence.

Muelenbrock, M. e Hoppe, U. (1999). Computer supported interaction analisys of group problem solving. *CSCL*, Vol. 1pp. 398–405. California.

Muller, J. (1998). Architectures and applications of intelligent agents: A survey. *Knowledge Engineering Review*, Vol. 13, No. 4, pp. 353–380.

Murray, T. (1999). Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education*, Vol. 10, No. 1, pp. 98–129.

Murray, T., Blessing, S., e Ainsworth, S. (2003). An overview of intelligent tutoring system authoring tools: Updated analysis of the state of the art. *Authoring Tools for Advanced Technology Learning Environments*, Vol. 1, No. Chapter 17, pp. 491–538.

Nwana, H. S. (1995). Software agents: An overview. *Knowledge Engineering Review*, Vol. 11, No. 2, pp. 205–244.

Oliveira, F. M. d. (1996). Inteligência artificial distribuída. *ESCOLA REGIONAL DE INFORMÁTICA Anais da Sociedade Brasileira de Computação*, Vol. 4pp. 54–73.

Park, O. (1988). Functional characteristics of intelligent computer-assisted instruction: Intelligent features. *Educational Technology*, Vol. 1, No. 7-14, .

Pozzebon, E. (2003). Tutor inteligente adaptavel conforme as preferências do aprendiz. Dissertação de Mestrado, Pos-graduação em Ciência da Computação da Universidade Federal de Santa Catarina.

Pozzebon, E., Cardoso, J., Bittencourt, G., e Hanachi, C. (2006). A multi-agent architecture for group learning. *IADIS International Conference e-Society 2006, Dublin, Ireland*, Vol. 13-26pp. 1032–1043.

Pozzebon, E., Cardoso, J., Bittencourt, G., e Hanachi, C. (2007). A group learning management method for intelligent tutoring systems. *International Journal of Computing and Informatics*, Vol. 31pp. 191–199. ISSN: 0350-5596.

Razzaq, L., Feng, M., e Nuzzo-Jones, G. (2005). The assistment project: Blending assessment and assisting. *Proceedings of the 12th International Conference on Artificial Intelligence In Education*, Vol. 1pp. 555–562. Amsterdam: ISO Press.

Rodrigues, L. e Carvalho, M. (2004). Emotional and motivational its architecture. *Proceedings of the IEEE International Conference on Advanced Learning Technologies, ICALT 2004*, Vol. 2.

Rodrigues, M., Costa, A., e Bordini, R. (2003). A system of exchange values to support social interactions in artificial societies. *AAMAS 2003 - Second Joint Conference on Autonomous Agents and MultiAgent Systems, Melbourne*, Vol. 1pp. 291–301.

Rosatelli, M., Self, J., e Thiry, M. (2000). Lecs: A collaborative case study system. *In Lectures Notes in Computer Science. Inteligent Tutoring Systems. Procedings of 5th International Conference, ITS 2000*, Vol. 1pp. 242–251. Montreal/Canada.

Saywell, M. (2002). Negotiating agents: An overview. *Multimedia Systems Coursework, Dept. of Electronics and Computer Science. UK*, Vol. 1. Disponivel em http://mms.ecs.soton.ac.uk/mms2002/papers/21.pdf, acessado 10 de agosto de 2008.

Scalabrin, E., Vandenberghe, L., de Azevedo, H., e Barthes, A. (1996). A generic model of cognitive agent to develop open systems. *Proceedings of the 13th Brazilian Symposium on Artificial Intelligence*, Vol. 1pp. 61–70. Springer-Verlag.

Self, J. (1992). Computational mathetics the missing link in intelligent tutoring systems research. *International Journal Artificial Intelligence In Education*, Vol. 1pp. 146.

Self, J. (2000). The defining characteristics of intelligent tutoring systems research: Its care, precisely. *International Journal of Artificial Intelligence in Education*, Vol. 10pp. 350–364.

Shneiderman, B. (1992). *Designing The User Interfaces: Strategies For Effective Human Computer Interaction*, Vol. 1. Addisson-Wesley.

Smith, R. G. (1980). The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, Vol. C-29, No. 12, pp. 1104–1113.

Staab, S., Studer, R., Schnurr, H., e Sure, Y. (2001). Knowledge processes and ontologies. *IEEE Intelligent Systems*, Vol. 16, No. 1, pp. 26–34.

Thibodeau, M., Bélander, S., e Frasson, C. (2000). White rabbit.matchmaking of user profiles based on discussion analysis using intelligent agents. *Procedings of 5th International Conference, ITS 2000*, Vol. 1pp. 113–122. Montreal/Canada.

TILAB (2007). Java agent development framework. Disponível em <http://jade.tilab.com/>. Acesso em 20 de Maio de 2007.

Vicente, A. e Pain, H. (1998). Motivation diagnosis in intelligent tutoring systems. *Proceedings 4th International Conference ITS98 San Antonio*, Vol. 1, No. 1, pp. 86–95.

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems*, Vol. 1. Morgan Kaufmann Publishers.

Wooldrige, M. (2002). *Introduction to MultiAgent Systems*. John Wiley and Sons, Chichester, England.

Wooldrige, M. J. e Jennings, N. (1995). Agent theories, architctures and languages: A survey. *Intelligent Agents Springer-Verlag, Berlin*, Vol. 1.

Yamane, E. (2006). Modelagem e implementação de uma ferramenta de autoria para construção de tutores inteligentes. Dissertação de Mestrado, Curso de Pós-Gradução em Engenharia Elétrica da Universidade Federal de Santa Catarina.