Dissertação de Mestrado

Arquitetura e Modelos de Interações Cooperativas e Adaptativas entre Agentes Humanos e Artificiais no Domínio de Fração

Maria Aparecida Amorim Sibaldo
maasibaldo@gmail.com

Orientador:
Evandro de Barros Costa

Maceió, novembro de 2009

Maria Aparecida Amorim Sibaldo

Arquitetura e Modelos de Interações Cooperativas e Adaptativas entre Agentes Humanos e Artificiais no Domínio de Fração

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Curso de Mestrado em Modelagem Computacional de Conhecimento do Instituto de Computação da Universidade Federal de Alagoas.

Orientador:

Evandro de Barros Costa

Maceió, novembro de 2009

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecária Responsável: Maria Auxiliadora Gonçalves da Cunha

S563a Sibaldo, Maria Aparecida Amorim.
Arquitetura e modelos de interações cooperativas e adaptativas entre agentes humanos e artificiais no domínio de fração / Maria Aparecida Amorim Sibaldo. – 2009.
124 f. ; il.

Orientador: Evandro de Barros Costa.
Dissertação (mestrado em Modelagem Computacional de Conhecimento) – Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2009.

Bibliografia: f. 84-90.

1. Sistema tutores. 2. Modelagem aberta do aprendiz. 3. Sistema multi-agente. I.Título.

CDU: 004.78

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Modelagem Computacional de Conhecimento pelo Programa Multidisciplinar de Pós-Graduação em Modelagem Computacional de Conhecimento, da Universidade Federal de Alagoas, aprovada pela comissão examinadora que abaixo assina:

Prof. Dr. Evandro de Barros Costa
UFAL – Instituto de Computação
Orientador

Prof. Dr. Patrick Henrique Silva Brito
UFAL – Instituto de Computação
Co-orientador

Prof. Dr. Leandro Dias da Silva
UFAL – Instituto de Computação
Examinador

Prof. Dr. Crediné Silva de Menezes
UFES – Departamento de Informática
Examinadora

Maceió, novembro de 2009.

Resumo

Este trabalho apresenta um ambiente interativo de aprendizagem sobre Frações, dotado de mecanismos de suporte a interações cooperativas e adaptativas oferecidas por seus agentes tutores aos aprendizes humanos, focando principalmente em atividades de resolução de problemas. Para isso, propõe-se uma arquitetura baseada em agentes de *software* e serviços *Web* semânticos, daí, pôde-se verificar a viabilidade funcional da proposta e, posteriormente, apresentar uma revisão de tal arquitetura para suprir alguns requisitos anteriormente não visados, além de modelos que dão suporte às referidas interações. No que diz respeito às interações, o aprendiz receberá suporte pedagógico tanto de um agente tutor, quanto de algum de seus pares que fazem parte do ambiente. Particularmente, um agente tutor conta com um modelo aberto do aprendiz, a partir do qual passa a dispor de informações úteis para orientar suas ações. A idéia deste modelo ser aberto é a de permitir que o aprendiz possa ver qual a avaliação que o sistema tem a seu respeito, tendo ainda a oportunidade de discordar de tal avaliação, e assim contribuir para o refinamento do conteúdo de tal modelo.

Palavras-chave: Modelagem Aberta do Aprendiz; Sistemas Tutores Inteligentes; Sistemas Multi-agentes.

Abstract

This work presents an interactive environment for learning about fractions, with mechanisms to support cooperative and adaptive interactions offered by tutors agents to human learners, focusing mainly on activities to solve problems. For this purpose, an architecture based on software agents and semantic Web services was proposed, therefore, we verify the functional viability of the proposal and, posteriorly, to present a revision of that architecture to suply some requirements not previously covered, beyond models that support to those interactions. With respect to interactions, the learner will receive support from both a pedagogical agent tutor, as some of their peers who are part of the environment. Particularly, a tutor agent has an open learner model, from which it obtains information to guide their actions. The idea of this model be opened is to allow the learner seeing the evaluation that the system has about him, and also the opportunity to disagree with this assessment, and thus contribute to the refinement of the content of such a model.

Key-words: Open Learner Model; Intelligent Tutor Systems; Multiagent Systems.

Agradecimentos

Agradeço a DEUS, por tão grande amor para comigo e com minha família, por sempre me guiar, por me ter dado inúmeras segundas-chances, por não desistir de mim, por ser o meu Senhor e meu DEUS!

Agradeço aos meus pais, Margarida e José Sibaldo, pela doação para que seus filhos pudessem ter uma vida melhor que a deles, além de todo o amor, paciência, conselhos, ensinamentos, conversas, apoio, por terem me ensinado a não desistir e por tão gostosas comidinhas.

Agradeço também a minha irmã (Cristina de Fátima), irmãos (Marcos José, Marcone e Marcelo - o Amarelo), sobrinhas (Laís, Cleane, Melina Aparecida, Marcele Cristina, Milena Beatriz e Lívia Maria), sobrinhos (Mateus e Lucas) e cunhados (Clélia, Paula Helen, Douglas e Carol), por me darem um alicerce de alegria, carinho e incentivo que me faz continuar, sempre. Em especial, ao Marcelo, por ter dividido comigo as alegrias e dores de se concluir um mestrado, no caso dele, doutorado.

Agradeço ao meu orientador professor Doutor Evandro de Barros Costa por todas as oportunidades, preocupações, conselhos, desafios, tempo disponível e ajuda ao longo desta caminhada.

Agradeço ao professor Doutor Patrick Henrique Brito, pelas constantes ajudas para que este trabalho fosse realizado, além de sua paciência, conversas e tempo disponível.

Agradeço ao Pedro Bispo, pela enorme ajuda, tempo disponível, conselhos e chás de ânimo, tão necessários para a conclusão deste. Agradeço por ter sido um amigo durante todo esse tempo e peço desculpas pelas aulas de francês que ele teve que perder.

Agradeço ao Alexander Toledo e ao Anderson Fellipe Rocha pela amizade e ajuda na construção deste trabalho.

Agradeço ao professor Doutor Ig Ibert Bittencourt pela disponibilidade de recursos e dicas para a implementação deste.

Agradeço a todos do GroW, entre eles: Marlos Tárcio, Heitor, Lucas Braz, Rafael Ligeiro, Douglas Véras, Társis Marinho e Fernanda pelos esclarecimentos e ajudas acerca de algum tema.

Agradeço ao senhorito Oswaldo Cavalcante por ter me apresentado uma outra visão sobre o que é O mais importante nesta vida e por ter se deixado guiar pelo Espírito Santo para me aproximar mais do grande EU SOU. E agradeço ainda pelo seu cuidado, vindo como acréscimo.

Agradeço a muitas outras pessoas amigas pelas constantes perguntas sobre o andamento deste, pelas cobranças, pelas orações, pela torcida, entre elas: Ana Paula, Maria Cristiane, Cristiano Amorim, Patrícia Lessa, Camila Bezerra, Marcelo Damasceno, Alia Titara, Fabiane Queiroz, Hugo Estanislau, Douglas Cedrim, Alysson Cabral, Adelailson Peixoto, Vinícius, Leonardo Carvalho, Clarissa Codá, Renata Thomaz, Adeilson Sedrins, Manu Albuquerque, Rafael Lima, Denise Américo, Carlos Jean, Valdick Sales, Edmilson Vidal, Karla Danielle, Andressa Pontes, Déborah Cerqueira, Thaíse Tojal, Joel Henrique, Rosa Carolina, Evellyn Soares, Walter, Hélio, Leonardo, José Neto, Felipetcho, Sidney, Rosângela, Carol e ao pessoal do EJC de Cruz das Almas e ao pequeno grupo Por Amor.

Agradeço a FAPEAL por ter me possibilitado concluir este trabalho.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Os ambientes virtuais educacionais oferecem suporte ao aprendizado do estudante, estando este em um contexto de sala de aula presencial ou não, através de diversas ferramentas de *software*. Muitos desses sistemas focam o aprendizado individual do aprendiz, ou seja só o computador e ele, principalmente valendo-se do uso da Internet e da *Web*. Uma das modalidades dos referidos ambientes é a dos Sistemas Tutores Inteligentes (STI), que, segundo [29], são programas de *software* que dão suporte à atividade de aprendizagem, seja este processo da forma convencional (em salas de aula) ou em cursos à distância, seja através de sistemas *desktop* ou da *Web*. Os STIs guiam os aprendizes durante seu aprendizado de um determinado domínio, sendo um agente capaz de se assimilar a um professor humano [52].

O uso de STI na *Web* é cada vez mais comum, podendo o usuário fazer uso do sistema em diversos lugares e, mesmo assim, o STI manter as informações sobre o aprendiz e poder guiá-lo no seu processo de aprendizagem de onde foi parado. Um outro ponto que está cada vez mais comum em relação aos STI, é a construção de sua arquitetura ser baseada em sistemas multiagentes, que traz também como vantagem em relação às arquiteturas tradicionais uma flexibilidade maior no tratamento dos componentes que compõem o sistema [9] e por poderem resolver problemas de grande complexidade de forma cooperativa e dinâmica [28].

Há quatro módulos básicos em um STI, os quais são: Interface, que será o meio pelo qual o aprendiz irá interagir com o sistema; Módulo do Domínio, onde são guardadas informações sobre o que será ensinado ao aprendiz; Módulo Pedagógico, o qual apresenta as estratégias de ensino [26]; e Módulo do Aprendiz, que mantém informações sobre o aprendiz e seu estado cognitivo.

O Modelo do Aprendiz é construído através da união de diferentes aspectos do aprendiz e de seu processo de aprendizagem [62]. Como isso se dá através

da inferência do sistema a partir da interação do aprendiz com o ambiente, muitas vezes a modelagem inferida pelo sistema sobre o processo de aprendizagem do aprendiz não é fiel ao seu estado cognitivo [38, 39, 12], pois muitas vezes o aprendiz pode saber um ponto do estudo, mas ter em sua modelagem como não sabendo, podendo isso ter ocorrido por próprio erro de inferência ou do sistema, ou por dispersão do aprendiz.

O desenvolvimento de um STI não é uma tarefa simples por aliar tecnologia com educação, pois envolve, além da arquitetura do sistema, diversas áreas da Computação, e também interdisciplinaridade com áreas pedagógicas [21]. Além desse fator, há a escolha das tecnologias usadas e também do conhecimento destas, pois muitos STI, quando construídos, começam do zero, não fazendo reuso, nem integração [47] com outros sistemas já prontos e robustos.

Durante o processo de aprendizagem do aprendiz em um STI, ele terá a ajuda de um tutor virtual durante todo o tempo, porém, em algum momento o aprendiz pode se achar no seguinte impasse: tutor já ter usado de todos os seus recursos para tirar tal dúvida do aprendiz e este permanecer na dúvida. Esses impasses na aprendizagem sempre surgem, porém, em alguns assuntos da matemática, seja ela básica ou não, eles tendem a ser maiores e mais constantes, necessitanto, assim, de uma aprendizagem flexível e robusta durante seu aprendizado [50].

Portanto, esta dissertação apresenta um ambiente educacional voltado ao ensino de fração baseado em sistemas multiagentes, os quais fazem uso de serviços *Web* semânticos, e que fornece uma aprendizagem adaptada para cada aprendiz, tal ambiente é chamado de FraW (*Fraction in the Web*).

O FraW é um ambiente que tem a presença de agentes de *software* e aprendizes que interagem entre si de forma que, tendo o aprendiz um problema/-questão a resolver, (1) ele pode solicitar a ajuda de um agente tutor para solucionar tal problema. Um agente tutor pode também solicitar a ajuda de outros tutores para ajudar ao aprendiz (2). Além disso, caso o aprendiz tenha algum impasse de aprendizagem, mesmo depois de sua interação com um agente tutor, (3) ele poderá receber como uma recomendação personalizada [45] um outro aprendiz que tenha um conhecimento complementar ao seu. Este cenário está apresentado na Figura 1.1.

Um exemplo da recomendação de par-complementar pode ser da seguinte forma:

- Um aprendiz que tenha uma certa dificuldade em um determinado assunto pode receber como recomendação um aprendiz de alto conhecimento no assunto, para que, com essa interação o primeiro aprendiz

possa melhorar seu conhecimento no assunto em questão. Para que as recomendações sejam feitas de forma eficaz, é preciso se ter um bom conhecimento dos usuários, neste caso, os alunos.

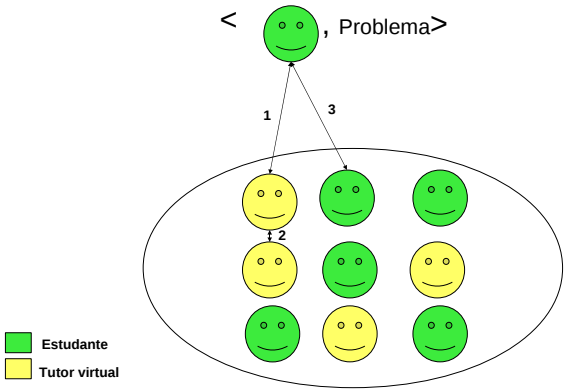

Figura 1.1: Exemplo de interações entre os agentes humanos e virtuais no FraW

As informações que serão usadas para se encontrar os pares-complementares são as informações cognitivas do aprendiz, ou seja, as informações que o sistema irá inferir a partir da interação do aprendiz com o ambiente durante seu processo de aprendizagem. Para isso, será preciso fazer uso da modelagem do aprendiz (ou outros perfis, como professor, monitor, pai etc), o qual foi construído aqui com base no padrão IMS [15], que contém informações que devem estar contidas no Modelo do Aprendiz, e sua taxonomia representada por ontologias [33]. Como citado, as informações contidas na modelagem do aprendiz pode não estar fiel com o estado cognitivo do aprendiz, portanto, o FraW permite que o aprendiz possa discordar do que está sendo apresentado a ele e, a partir de então, o sistema e o aprendiz começarem uma negociação para a atualização do estado cognitivo do aprendiz no sistema ou não.

Para que a interação entre o aprendiz que está com dificuldades de resolver o problema e o que irá ajudá-lo se dê, o sistema dispõe de uma ferramenta de *chat* (podendo ser outras ferramentas, tais como fórum ou portfólio) onde é analizada se a interação entre eles foi feita de acordo com a dúvida que se quer sanar. Os problemas a serem resolvidos no FraW são problemas matemáticos do domínio de fração; e os aprendizes são colocados em contato a partir da ferramenta de recomendação.

Baseado nisso, essa dissertação tenta unir características dos Sistemas Tutores Inteligentes com aprendizagem em grupos, onde estes interagem entre si de forma cooperativa ou colaborativa, que é um tipo de atividade social

que normalmente envolve alunos e professores que compartilham e recebem conhecimento [55, 4]. Este trabalho foca a união de pares complementares, apresentado na Figura 1.2, onde vemos a interação do aprendiz tanto com os agentes tutores que irão guiá-los como um professor ao passar o conteúdo de aprendizagem e avaliá-lo, quanto com outros aprendizes humanos, no qual ele pode receber ajuda durante seu processo de aprendizagem.

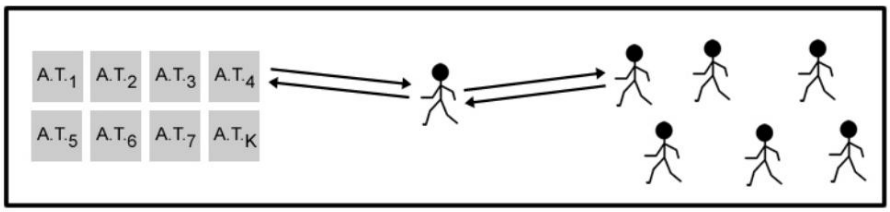

Figura 1.2: Aprendiz interagindo com tutores de *software* e outro aprendiz

Portanto, temos como desenvolvimento desta dissertação o FraW (Fraction in the *Web*), desenvolvido no Instituto de Computação da Universidade Federal de Alagoas (UFAL), que consiste em um ambiente de educação voltado ao ensino de Fração para as séries iniciais do ensino fundamental [23, 53], e que une em seu ensino a interação do aprendiz com agentes tutores de *software* e de outros aprendizes, caso precise, de um outro aprendiz.

Nas seções seguintes serão apresentados os Objetivos que se quer chegar (Seção 1.2) através desta dissertação, a Relevância deste trabalho (Seção 1.3) e, por fim, a Estrutura da Dissertação (Seção 1.4).

1.1 Problemática

A problemática deste trabalho está contemplada nos seguintes tópicos:

- Necessidade de se idealizar uma arquitetura de *software* que facilite o reuso e/ou integração com outros sistemas;
- Necessidade de reduzir o custo de desenvolvimento e evolução do sistema através de uma arquitetura de *software* flexível;
- Necessidade de melhorar a precisão do modelo do aprendiz, com respeito ao seu estado cognitivo, resultando em decisões mais acertadas por parte do tutor;
- Necessidade de estimular a interação entre pares em um STI, valorizando, assim, atividades cooperativas entre os aprendizes;

- Aumentar a escalabilidade, devido às perspectivas de uso em EaD (Educação a Distância).

1.2 Objetivos

Este trabalho propõe a construção de um ambiente interativo de aprendizagem baseado em agentes de *software* voltado ao domínio de Fração. Especificamente, objetiva-se:

- Uma arquitetura que visa o reuso e a integração deste sistema a outros, além de uma maior flexibilidade e facilidade de evolução;
- Uma solução para modelo aberto do aprendiz, permitindo que este aprendiz possa discordar do modelo gerado pelo sistema, isto levando a uma solução conciliatória via mecanismo de negociação;
- Propor a recomendação de par-complementar no STI, para que aprendizes que esteja em um impasse de aprendizagem, possa interagir com atores humanos através de uma ferramenta de *chat*.

1.3 Relevância

A concretização do ambiente proposto neste trabalho pretende contribuir com um ambiente educacional que permitirá apoiar tanto atividades presenciais, como complemento da sala de aula, quanto servirá para ser incorporada em uma plataforma de ambiente virtual para educação a distância. De uma maneira mais específica, espera-se ter contribuído com uma solução arquitetural apropriada para redução de custos no desenvolvimento de tais sistemas. Além disso, espera-se ter contribuído com uma solução prática para o modelo aberto do aprendiz, sendo este um recurso importante tanto para ter informação de maior qualidade para apoiar as decisões do sistema tutor, quanto para servir de um elemento de reflexão por parte do aprendiz. Por fim, procurou-se contribuir com a interação entre pares, com isso estimulando atividades cooperativas entre os estudantes.

1.4 Estrutura da Dissertação

A estrutura da dissertação aqui apresentada é a que se segue:

- Capítulo 2: Embasamento Teórico, o qual apresenta a descrição de alguns conceitos utilizando por este trabalho, para que facilite a leitura deste;
- Capítulo 3: Trabalhos Relacionados, apresenta o estado da arte através de alguns trabalhos que têm semelhanças a este no que corresponde a modelagem e negociação do modelo aberto do aprendiz, STI que fazem uso do recurso de pares-complementares e sistemas educacionais que fazem uso da *Web* semântica;
- Capítulo 4: Especificação dos Requisitos e Projeto da Interface Gráfica, onde é apresentado os requisitos do FraW e apresentada sua interface gráfica, além de alguns casos de uso do aprendiz no sistema;
- Capítulo 5: O Sistema FraW (*Fraction in the Web*) e os Modelos Associados, o qual descreve como o FraW foi construído, sua arquitetura, modelagem através de ontologias, agentes e serviços semâticos usados;
- Capítulo 6: Avaliação e Refinamento dos Resultados, onde as formas de interações existentes no sistema FraW são apresentadas e descritas, além de uma arquitetura idealizada a partir de ajustes feitos após as avaliações iniciais apresentadas no Capítulo 4;
- Capítulo 7: Conclusão e Trabalhos Futuros, onde os resultados, conclusões e trabalhos futuros são apresentados.

E, finalizando, temos a apresentação das Referências utilizadas.

x

Capítulo 2

Embasamento Teórico

Para que haja um melhor entendimento deste trabalho, serão apresentados a seguir alguns conceitos fundamentais envolvidos no mesmo. Com esse propósito, neste capítulo são apresentadas algumas características de Sistemas Tutores Inteligentes (STI) e sua arquitetura. Outros temas abordados aqui são: ontologias, modelo Mathema, agentes inteligentes de *software* e sistemas multiagentes, serviços semânticos e sobre o ambiente interativo Massayo/-ForBILE, onde será realizado o estudo de caso dos serviços criados.

2.1 Sistemas Tutores Inteligentes

Os Sistemas Tutores Inteligentes (STI) que, segundo [29], "são programas de *software* que dão suporte às atividades de aprendizagem", fazem uso de técnicas de Inteligência Artificial na Educação para proporcionar ensino individualizado com interação ativa [26]. Tais sistemas surgiram na década de 90 em contrapartida aos sistemas CAI (*Computer Aided Instruction*), que tinham sua representação de forma estática, nas quais suas decisões eram tomadas antes de sua implementação [26].

A Figura 2.1 apresenta a arquitetura clássica de um STI a qual tem três módulos básicos de um STI e sua interface gráfica. Tais módulos são os que se seguem:

- Módulo do Aprendiz, o qual age mantendo e modelando informações sobre **quem** é o aprendiz, respeitando as características e particularidades de cada um, sendo estas recebidas de forma explícita, informadas pelo aprendiz através de um formulário (como nome e *e-mail*), ou implícita, inferidas a partir das ações do aprendiz no sistema [11, 37]. Uma particularidade dos Modelo do Aprendiz é o Modelo Aberto do Aprendiz, que possibilita ao aprendiz observar seu andamento do aprendizado, podendo

também contestar as informações dispostas no seu perfil criado pelo ambiente. Características importantes no Modelo do Aprendiz Aberto são:

  - Possibilita ao aprendiz contribuir com informações sobre seu perfil: no caso de alguma contrariedade da inferência do sistema com a opinião do aprendiz, o ambiente pode propor um teste sobre o quesito que o aprendiz não confirma, depois desse, as informações serão atualizadas ou não, de acordo com o resultado do teste;
  - Provê reflexão e avaliação do aprendiz sobre seu aprendizado;
  - Ajuda ao aprendiz a planejar e monitorar sua aprendizagem [11].

- Módulo do Domínio, módulo que contém o conhecimento sobre **o que** será lecionado ao aprendiz: seus conceitos, questões, dicas, o conteúdo em si;

- Módulo Tutor, também conhecido como Módulo Pedagógico, onde estão as informações sobre **como** como passar o conhecimento ao aluno, ou seja, responsável pelas estratégias e táticas de ensino.

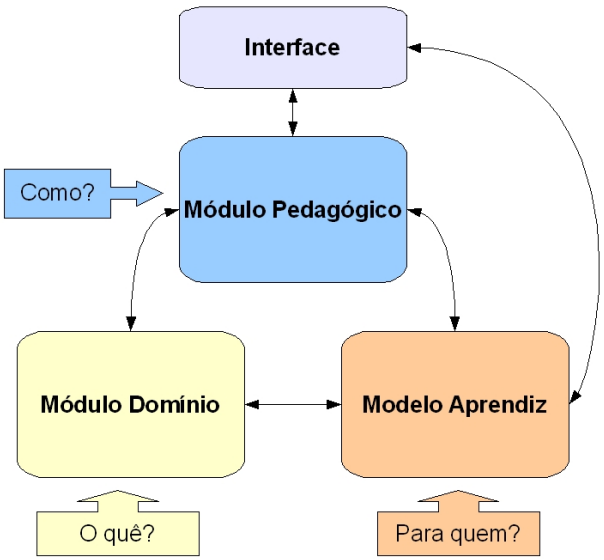

Figura 2.1: Arquitetura Clássica de um Sistema Tutor Inteligente (adaptado de [48])

Os STIs criam modelos computacionais cognitivos de acordo com o entendimento dos aprendizes e responde as suas ações de acordo com suas interações e erros típicos, identificados nos modelos. Para cada aprendiz, o

STI, usando técnicas de Inteligência Artificial e imitando a ação de um tutor humano, age de forma personalizada, respeitando seu processo individual de aprendizagem: respondendo a questões dos alunos e apresentando estratégias de resolução [55].

2.2 Ontologias

Em Ciência da Computação, uma ontologia é um conjunto de entidades com suas relações, restrições, axiomas e vocabulário, o qual pode definir um domínio ou, mais formalmente, uma conceitualização acerca deste domínio [33], além de representar um vocabulário comum entre usuários e ambiente [56]. As ontologias são utilizadas em Engenharia de *Software*, Inteligência Artificial e *Web* Semântica como uma forma de representação do conhecimento, especificando formal e explicitamente uma conceituação compartilhada [32, 27]

O vocabulário das ontologias são descritas através de classes de objetos, atributos, relações e indivíduos (instâncias) [27]. As ontologias utilizadas aqui foram especificadas/conceitualizadas/formalizadas/implementadas e sua manutenção é feita através da ferramenta Protégé [49], a qual possibilita o desenvolvimento de ontologias nas três sublinguagem OWL (*Ontology Web Language*):

- OWL *Lite*: representa uma linguagem muito simples onde sua expressividade é baixa, porém é recomendada para definição de hierarquias e restrições simples;
- OWL-DL: é a linguagem que foi utilizada nas ontologias utilizadas aqui. Ela é mais expressiva que a OWL *Lite* e é decidível, assim como essa última, implementa a Lógica de Descrição [2, 47];
- OWL FULL: as linguagens OWL *Lite* e DL estão contidas nessa linguagem e, apesar dela ser muito expressiva, não há garantia de computabilidade. É possível manipular e modificar metaclasses.

É bastante desejável o uso de ontologias em Sistemas Multiagentes pois servem como ferramenta para a construção de novos agentes, devido a sua organização, reuso e disseminação de conhecimento. Além disso, a troca de mensagem entre os agentes será facilitada devido a ontologia definir um mesmo vocabulário em comum [27].

2.3 Modelo Mathema

O Mathema [25] é um modelo de ambiente interativo de aprendizagem baseado em múltiplos agentes, o qual visa dar um bom provimento de suporte da aprendizagem do aprendiz a partir do conhecimento sobre o domínio, de pedagogia, do aprendiz, e da capacidade de interação.

Acerca da modelagem do conhecimento sobre um domínio, o modelo Mathema se apresenta como na Figura 2.2, onde se tem um domínio D constituído por seus dij, que são apresentados como pontos do plano cartesiano como a profundidade e o contexto de um certo domínio, como pode ser visto do primeiro plano da figura. Nos outros planos são apresentados domínio laterais ao domínio D.

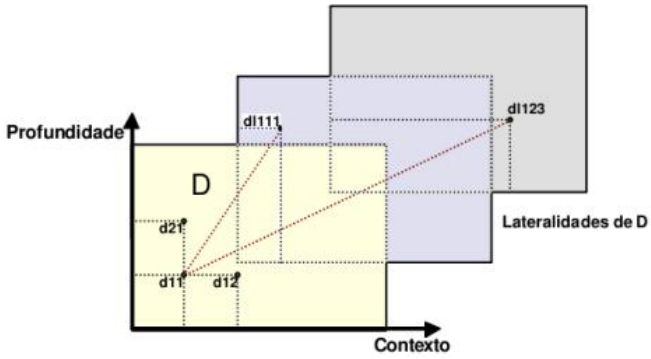

Figura 2.2: Visão multidimensional do conhecimento do conhecimento do domínio [25])

Um outro ponto apresentado no modelo Mathema é a sua arquitetura baseada em agentes tutores, no qual um aprendiz poderia interagir não apenas com um agente, mas com uma sociedade de agentes tutores, como apresentado na Figura 2.3. Na figura citada, é apresentada o Mediador Externo, o qual pode ser um professor, tutor etc; a Sociedade de Especialistas Humanos (SEH), fonte integrada de conhecimento externa ao sistema computacional; Aprendiz Humano, o qual irá interagir com o sistema para aprender um determinado domínio; Sociedade de Agentes Tutores Artificiais (SATA), o qual é composto por vários agentes tutores que podem cooperarem entre si para promover a aprendizagem de um determinado aprendiz; Agente de Manutenção, representando o elo de ligação entre o SEH e o SATA; Agente de Interface, que representa o elo entre o Aprendiz e o SATA.

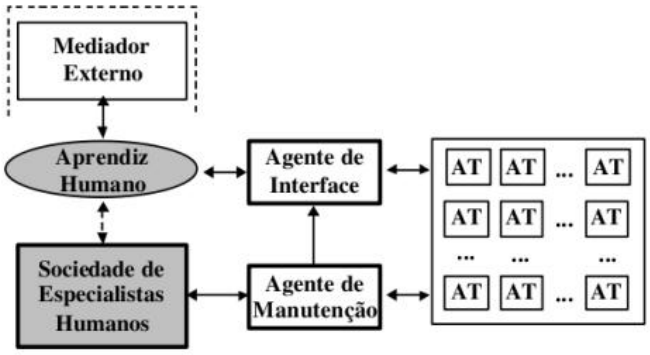

Figura 2.3: Arquitetura do Mathema [25])

2.4 Agentes de *Software* e Sistemas Multiagentes

Um Agente de *Software* é uma entidade artificial ou real que está situada em um ambiente virtual podendo agir sobre ele e se comunicar com outros agentes. Um agente percebe seu ambiente através de sensores, age sobre o ambiente através de atuadores, possui objetivos próprios explícitos ou implícitos e escolhe suas ações a partir das percepções que ele tem de seu ambiente.

O agente racional é um agente que faz a melhor coisa possível seguindo o conceito de racionalidade, pois, dando uma sequência perceptiva o agente escolhe, segundo seus conhecimentos, as ações que melhor satisfazem seu objetivo. Um agente racional pode ser descrito através de seu PAGE, onde o P quer dizer *Perceptions* (percepções), quais as percepções do agente em relação ao ambiente; o A, *Actions* (ações), as ações que o agente pode executar; o G, *Goals*, quais os objetivos do agente; e o E, *Environment* (ambiente) que o agente age [1].

Um Sistema Multiagente (SMA) é a união de dois ou mais agentes em um mesmo ambiente podendo interagir entre si, onde tem como metáfora o comportamento social. Os agentes, protocolos de interação e modelos organizacionais de um SMA são projetados independentemente do problema específico que será solucionado. Assim, a reutilização de tal projeto se torna viável para outro problema [51].

Uma estrutura típica de um SMA é apresentada na Figura 2.4. Um SMA contém um determinado número de agentes que se comunicam entre si (as setas da figura apresentada representam as interações). Os agentes são capazes de agir em um ambiente, e diferentes agentes têm diferentes esferas de influência (representada na figura pela Organização). O agente tem controle, ou tem influência, sobre sua esfera de influência e um ou mais agentes podem ter controle sobre a mesma esfera [64]. Na figura apresentada, podemos ver as visões dos agentes no ambiente, ou seja suas esferas de controle. Os

triângulos representam os recursos que os agentes fazem uso.

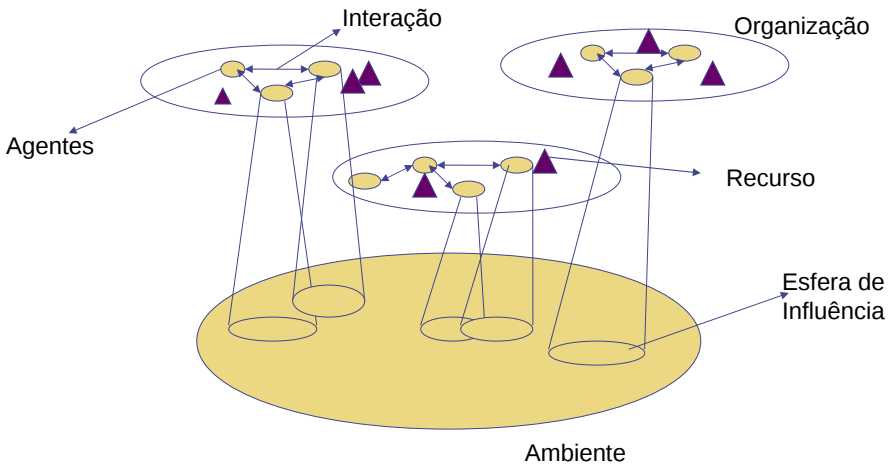

Figura 2.4: Estrutura de um Sistema Multiagente (adaptado de [64])

A escolha por Sistemas Multiagentes foi devido a sua grande capacidade de interação e cooperação entre os agentes, além da autonomia de cada um deles, possibilitando a execução concorrente de tarefas.

2.5 Serviços Semânticos

A tecnologia de *Web Services* facilita o uso de serviços remotos e a integração de sistemas, fazendo com que a comunicação entre aplicações remotas escritas em diferentes linguagens de programação, ou até mesmo desenvolvidas em diferentes plataformas, como Windows e Linux, possa ser feita de forma compatível, reutilizável e padronizada.

As estruturas dos *Web Services* têm como base padrões comuns na *Web*, como o formato de arquivo XML (*eXtensible Markup Language*) e o protocolo SOAP (*Simple Object Access Protocol*), utilizado na troca de mensagens entre aplicações e *Web Services* [17]. Cada *Web Service* é descrito através de um arquivo XML, utilizando-se a especificação WSDL (*Web Services Description Language*). Em cada um desses arquivos de descrição estão contidas informações sobre as funcionalidades que o *Web Service* proverá a seus clientes.

Como ilustrado na Figura 2.5, o provedor do serviço deve publicar tais serviços em um registro de serviços, conhecido como UDDI (*Universal Description Discovery and Integration*). Os serviços a serem publicados são definidos através de um arquivo WSDL. Para que o cliente de um *Web Service* possa encontrá-lo e obter sua definição, ele precisa, antes, fazer uma consulta ao registro de serviços (UDDI), de forma a receber as informações de que necessita para poder usar o serviço. Assim, após descobrir o serviço, o sistema cliente invoca o *Web Service* em um servidor remoto (i.e., provedor), encapsulando os dados da requisição em uma mensagem SOAP. O servidor, ao receber tal requisição, irá processá-la e executar o serviço desejado pelo cliente, enviando ao mesmo uma resposta encapsulada em uma mensagem SOAP.

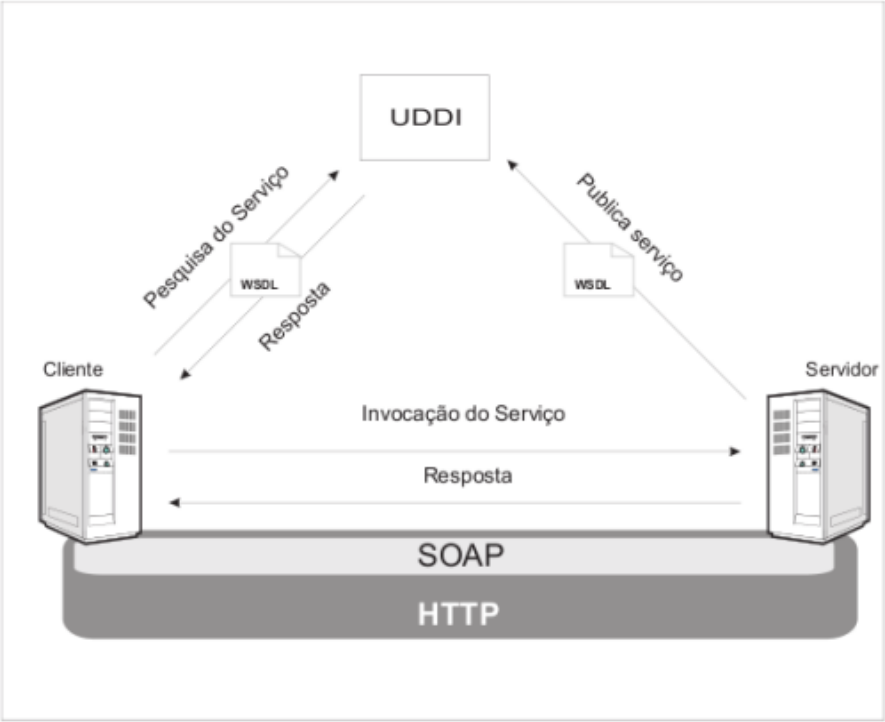

Figura 2.5: Funcionamento de um *Web Service*

Até então foi falado sobre os *Web Services*, porém a *Web* Semântica necessitou de serviços que o computador entenda e que os agentes de *software* possam descobrir, compor e executar automaticamente tais serviços [42]. Como os arquivos WSDL tem notação puramente sintática, nos Serviços Semânticos se busca fazer a descrição semântica dos serviços através de ontologias.

Os Serviços Semânticos tentam unir características dos Serviços *Web* com a *Web* Semântica, tendo, assim, o uso de serviços de forma flexível e com baixo acoplamento, ao mesmo tempo que se tem a interoperabilidade semântica. Portanto, os Serviços Semânticos são vistos como um aprimoramento dos Serviços *Web* com notações semânticas, conseguindo, assim, um alto grau de

automação na descoberta, composição, monitoramento e invocação dos Serviços *Web* [46].

2.6 Massayo

O Massayo é um *framework* que tem como intuito facilitar o desenvolvimento de sistemas educacionais multiagentes e semânticos, de forma a requerer o mínimo de modificação ou adição de código [8].

Como pode ser visto na Figura 2.6, existe uma comunidade de agentes no framework, e os agentes disponibilizados pelo *framework* são os seguintes:

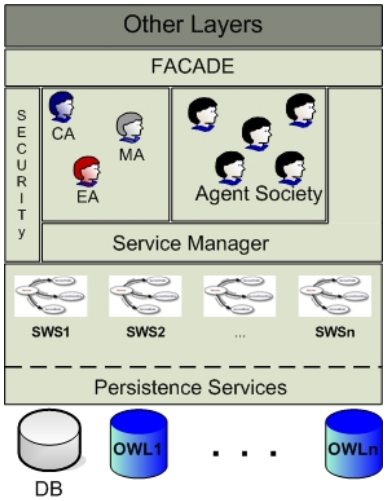

Figura 2.6: Arquitetura baseada em agentes do *framework*

- **Agente Controlador**: Na Figura 2.6, o agente controlador é representado por CA (*Controller Agent*). Tal agente é responsável por adicionar e remover agentes e serviços no ambiente em tempo de execução.
- **Agente de Evolução**: O agente de evolução (EA - *Evolution Agent*) é responsável por analisar as mudanças que ocorrem no sistema em tempo de execução. Essa verificação de mudanças é feita através de consultas as ontologias que armazenam informações sobre o ambiente, como por exemplo, quais são os agentes tutores que estão ativos no sistema e quais serviços tais agentes estão a oferecer.
- **Agente Mediador**: Tal agente (MA - *Mediator Agent*) é responsável por mediar as interações entre os agentes (tanto os agentes tutores quanto os agentes humanos).

- **Agente Tutor**: A sociedade de agentes da Figura 2.6 é composta de vários agentes tutores. Todo agente tutor do framework oferece serviços de gerenciamento do tutoramento do aprendiz, porém, como os problemas variam de acordo com o domínio da aplicação, é necessário implementar o mecanismo de resolução de problemas de cada tutor da sociedade de agentes (*Agent Society*).

Cada agente utiliza a camada de Gerenciador de Serviços como é mostrado na Figura 2.6, e a partir desta camada, é feita a invocação dos *web services* semânticos. Para invocar um serviço semântico, o agente passa para o gerenciador de serviços quais são as entradas, as saídas e a classificação do serviço que será invocado. A partir desses parâmetros, o Gerenciador de Serviços executa o algoritmo de invocação e infere qual o serviço, dentre os vários *web services* presentes no ambiente, que mais se encaixa de acordo com esses parâmetros e o invoca [13].

A Figura 2.7 demonstra a parte de infra-estrutura do *framework*. Tanto a informação relacionada ao modelo de ensino (modelo clássico de um STI: modelo do domínio, pedagógico e do aprendiz [61]), quanto a informação relacionada aos agentes do sistema que estão armazenadas em ontologias, representadas pela linguagem OWL (*Ontology Web Language*) [1].

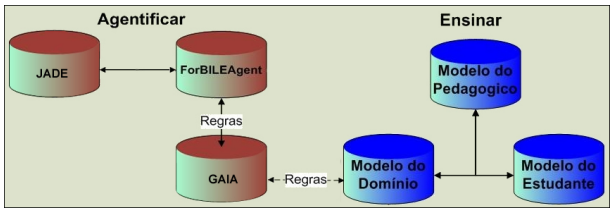

Figura 2.7: Ontologias do Massayo-F

Como pode ser visto, são três as ontologias ligadas à sociedade de agentes: GAIA, JADE e ForBILEAgent. A ontologia de GAIA guarda informações a cerca da modelagem dos agentes do sistema utilizando a metodologia GAIA [65]. A partir das informações presentes na ontologia de domínio, faz-se um mapeamento dos currículos presentes no domínio que estão sendo ensinados na aplicação (currículo nada mais é do que uma subárea do domínio dado, como no exemplo da aplicação proposta no presente trabalho, existe o domínio de fração, e dentro do domínio de fração existe o currículo de soma de fração [25]) para os agentes tutores da aplicação, dado que cada agente tutor é responsável por um currículo de acordo com as especificações do *framework*.

[1] http://www.w3.org/TR/owl-features/

Em seguida, depois que é feito esse mapeamento, os agentes modelados em GAIA e instanciados na ontologia de GAIA são mapeados para a ontologia ForBILEAgent (ForBILE é o antigo nome do *framework*, que é um acrônimo para *Framework for Building Interactive Learning Environments*) que extende a ontologia de JADE, que contém conceitos da plataforma JADE (*Java Agent DEvelopment Framework*) [5] representados em classes. Isso facilita a implementação de agentes, dado que a partir do modelo em GAIA e dos currículos do domínio, os agentes do sistema são gerados semi-automaticamente, sendo necessário apenas configurações mais detalhadas a nível de implementação que não são fornecidas durante a modelagem GAIA. Outra vantagem é a possibilidade da evolução dinâmica dos agentes, pois todas as informações sobre os seus comportamentos e seus serviços encontram-se na ontologia, não sendo totalmente preciso parar o ambiente durante a sua execução para realizar modificações na sociedade de agentes.

2.6.1 Camada da Aplicação

Essa camada pode ser visualizada na Figura 2.8, e ela é apresentada em [8]. Foi decidido então utilizar essa abstração onde se tem um conjunto de ferramentas interativas sendo observadas e manipuladas por uma sociedade de agentes interativos (SAI). Tais ferramentas interativas permitem o aprendizado colaborativo auxiliado por computador (CSCL). Como cada agente varia a partir da ferramenta interativa que ele observa e manipula, não foi criado um agente padrão para essa sociedade, assim como foi criado no *framework*, visto anteriormente, logo ele necessita ser especificado e implementado de acordo com as suas peculiaridades.

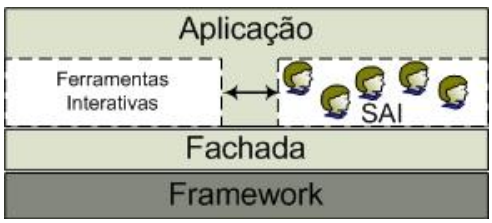

Figura 2.8: Camada da Aplicação do Massayo

Capítulo 3

Trabalhos Relacionados

Ao longo deste capítulo serão apresentadas alguns ambientes e ferramentas que têm similaridades com o ambiente educacional proposto por este trabalho. Tendo em vista esse objetivo, serão feitas breves descrições e comparações entre o FraW, ou alguns de seus serviços, e outros trabalhos.

3.1 I-Help

I-Help é um sistema educacional desenvolvido na Universidade de Saskatchewan que faz uso de várias ferramentas para apoiar a aprendizagem, tais como: *chat*, fórum, material *on-line*, entre outros. Além dessas ferramentas citadas, o I-Help faz uso de recomendações para estudantes que estão implementando algum programa [63], tal sistema de recomendação é um evolução do PHelpS [31, 30, 60].

Quando um determinado aluno está programando e encontra algum impasse, o sistema I-Help tenta ajudá-lo procurando por pessoas que já tenham passado por aquele problema. Essa ajuda ocorre da seguinte forma: o agente pessoal do aluno solicita ao *matchmaker* um outro usuário I-Help que esteja *on-line* e que já tenha passado por aquele impasse de programação, ou seja, que possa ajudá-lo. O *matchmaker* cria uma lista ordenada de usuários potenciais e a envia para o agente pessoal do estudante que solicitou a ajuda. Então, o agente pessoal do solicitante começa a negociar com os agentes pessoais dos outros estudantes, procurando encontrar um que concorde em ajudar pelo preço de unidades de crédito I-Help que seja satisfatório [10, 59], como pode ser observado na Figura 3.1.

Depois dos agentes pessoais chegarem a um acordo o agente pessoal do possível ajudante irá avisar ao seu respectivo estudante sobre a solicitação de ajuda; caso o estudante não aceite ajudar, o agente pessoal do estudante

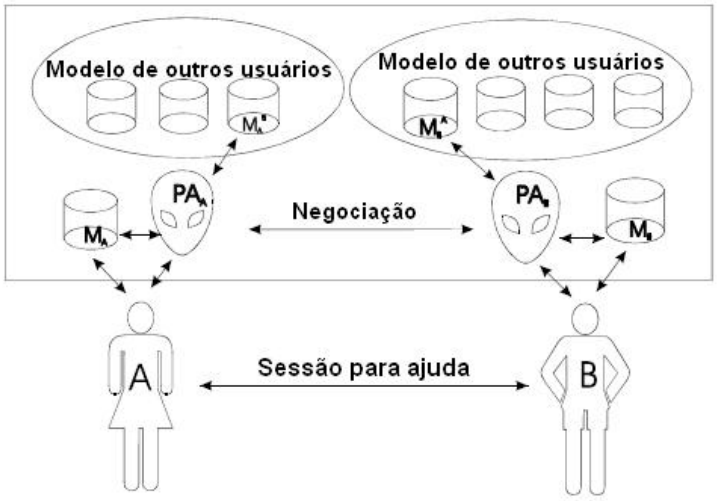

Figura 3.1: I-Help: cada agente pessoal mantém o modelo do seu próprio aprendiz e de seu par (adaptado de [**?**])

que solicitou a ajuda irá negociar com o próximo agente da lista; caso o estudante aceite ajudar, será aberta uma sessão de *chat* entre ambos: o que necessita de ajuda, e o que quer ajudar. Após o fim da conversa entre ambos, ao fechar a sessão de *chat*, será aberta uma janela para que cada estudante dê um *feedback* sobre o outro, para que o perfil de conhecimento de cada um seja atualizado e mantido, com essas e outras características, pelo agente *matchmaker* [59].

A arquitetura do I-Help é baseada em sistemas multiagentes e consiste de agentes pessoais e da aplicação, os quais usam ontologias e linguagem de comunicação em comum. Os agentes pessoais guardam informações do estudante como nível de conhecimento sobre alguns tópicos do domínio, além de outras informações, como: habilidade em ajudar, idade etc. Os agentes da aplicação mantêm o modelo dos tópicos endereçados pelo material instrucional [59].

O I-Help e o FraW têm grandes similaridades em relação a interação entre os pares-complementar, pois ambos procuram por um ajudante *on-line* para um aprendiz que necessite de ajuda (apesar do I-Help fazer isso de forma mais complexa), além de abrir uma janela de *chat* para que eles possam se comunicar, além de serem sistemas multiagentes que fazem uso de ontologias. Porém, o I-Help não tem em sua modelagem do aprendiz uma aceitação do próprio aprendiz, visto que o aprendiz nem sabe o que o ambiente conhece sobre ele, devido a não existência do odelo Aberto do Aprendiz. Sobre o domínio de cada ambiente, eles são diferentes: no FraW é estudado fração, enquanto no I-Help, linguagem de programação.

3.2 Aplusix

O Aplusix é um *software* que ajuda alunos de Ensino Fundamental e Médio durante a aprendizagem de álgebra, tratando da resolução de problemas através de exercícios.

Ele tem uma interface voltada à área matemática, como podemos ver na Figura 3.2, e tem três características básicas: (a) está sempre indicando se os cálculos estão ou não corretos; (b) caso pedido, fornece a solução de uma questão; (c) indica se o exercício que o estudante indicou como finalizado realmente o está [14].

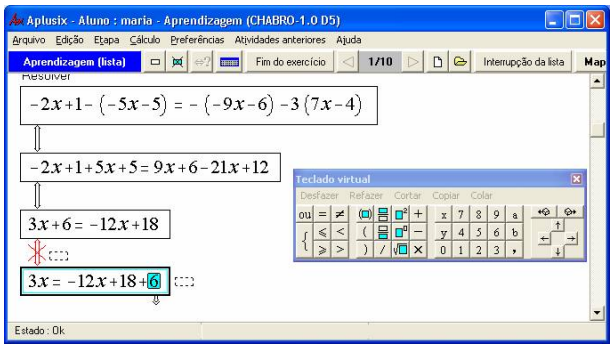

Figura 3.2: Tela de resolução de uma questão do Aplusix [14]

Aplusix traz uma base de exercício organizada sobre os diversos assuntos de álgebra a também possibilita ao professor inserir novas questões na base de questões do ambiente.

Há também a ferramenta de gravação, onde o *software* faz a gravação das ações do aprendiz. E o professor, como pode ser visto na Figura 3.3 pode visualizar as estatísticas de suas classes, avaliando sob o ponto de vista de exercícios feitos bem resolvidos (barra da cor verde) ou não (barra da cor amarela), dentre todos os exercícios resolvidos (barra da cor vermelha).

O Aplusix é um programa *desktop*, com versões exclusivamente para o sistema operacional Windows, e é um *software* comercializável, apesar de ter uma versão *trial*, possibilitando seu uso por 30 dias, disponível em seu *website*[1].

A primeira diferença que se nota do Aplusix em relação ao FraW é que o primeiro é *desktop*, enquanto que o segundo é *Web*. Além disso, o Aplusix trata de um domínio mais amplo (álgebra) que o FraW (operações com fração), e este último faz uso da modelagem aberta do aprendiz, sendo esta apresentada ao próprio aprendiz, onde ele pode negociar sua modelagem com o ambiente, enquanto que no Aplusix só os professores vêm o desenvolvimento do aluno

[1] http://aplusix.imag.fr

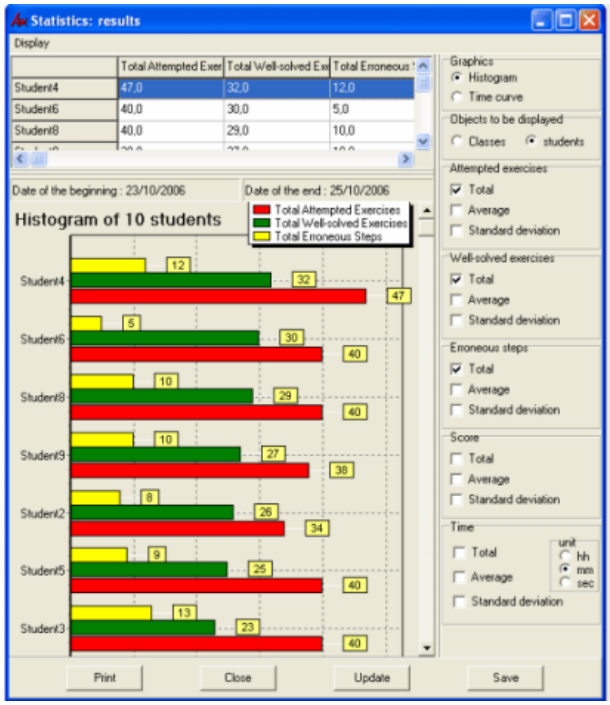

Figura 3.3: Tela de avaliação estatística dos aprendizes pelo professor [14])

através de suas modelagens. Em relação à interface, ambos dão um bom suporte para a resolução de questões com fração.

3.3 *ActiveMath*

ActiveMath é um ambiente *Web* de aprendizagem adaptativa, nos quais seu conteúdo matemático está representado semanticamente em arquivos XML (*eXtensible Markup Language*). O ActiveMath gera dinamicamente uma lista de conteúdo para um determinado indivíduo, de acordo com o conteúdo necessitado pelo aprendiz e também por suas preferências [44].

O ActiveMath, como pode ser visto em sua arquitetura, presente na Figura 3.4, integra os seguintes componentes: o gerenciador de sessão, a base de conhecimento matemático, o plano de apresentação, o modelo do aprendiz, o módulo pedagógico, o Omega (serviços computacionais de dedução para planejar provas) e o CAS (*Computer Algebra System*). Tais componentes se comunicam através da Internet [43].

Os cursos contidos no ActiveMath vão desde assuntos matemáticos preliminares, como frações, a assuntos para séries mais adiantadas, como: cálculo diferencial, lógica, estatística, entre outros. O ambiente também tenta estimar o conhecimento corrente do aprendiz através de exercícios que o aprendiz resolve e conteúdo que ele lê, além disso, a modelagem do aprendiz sobre um

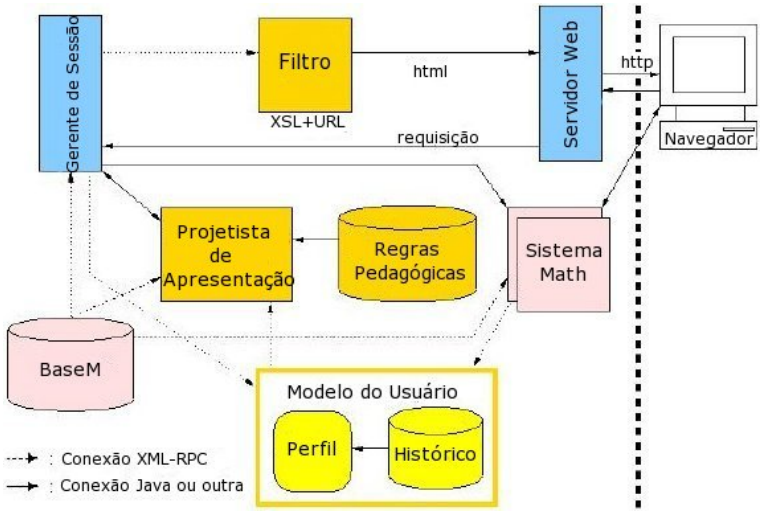

Figura 3.4: Arquitetura do ActiveMath (adaptado [44])

conhecimento lateral ao assunto em questão também é criado a medida que o aprendiz resolve questões que envolve tais conhecimentos. O sistema também permite que o aprendiz informe o que ele irá estudar no momento.

Tal sistema guia o aprendiz através de um modelo de sistema regular e individual, disponibiliza uma ferramenta de mapas conceituais para os alunos e também a ferramenta de busca semântica, onde o aprendiz faz a pesquisa por conteúdo a ser melhor estudado. Um exemplo do ActiveMath pode ser visto na Figura 3.5, onde está sendo apresentado um conteúdo ao estudante.

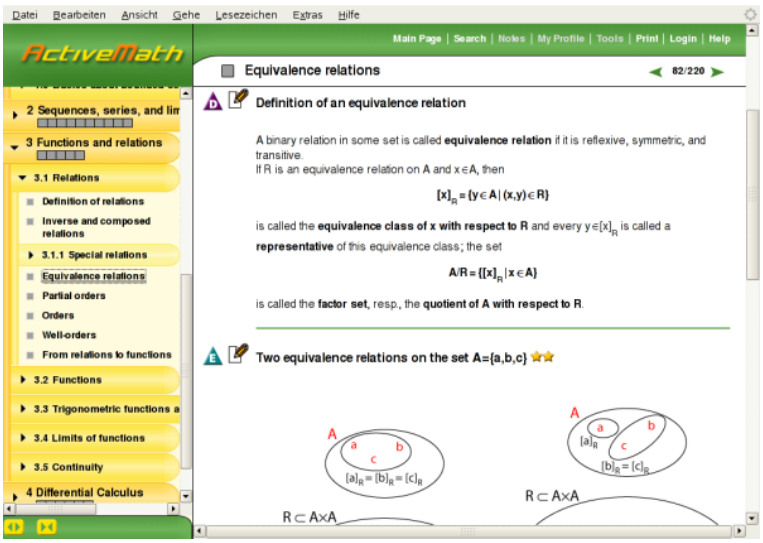

Figura 3.5: Tela de conteúdo do ActiveMath

A modelagem do aprendiz é representada por cores, onde o (i) cinza quer dizer que não há informação disponível, (ii) vermelho, onde precisa melhorar o conhecimento, (iii) laranja, informa que o aprendiz deveria saber mais, (iv) amarelo, está bem, mas deveria aprender mais, (v) verde claro, diz que o

aprendiz deve estar entendendo bem o assunto, (vi) verde escuro, informa que o aprendiz sabe muito bem o assunto.

O ActiveMath trabalha bem a parte de modelagem do aprendiz, abordando também sua modelagem lateral, porém o aprendiz não pode negociar a mudança de sua modelagem. O FraW faz uso da aprendizagem colaborativa, através de um par-complementar, enquanto no ActiveMath o aprendiz tem seu processo de aprendizagem sempre de forma isolada. Em contrapartida, o domínio do ActiveMath é maior que o do FraW.

3.4 Flexi-OLM

Flexi-OLM é um sistema tutor inteligente voltado ao ensino da linguagem de programação C++. As questões que ele apresenta são questões que se deve preencher lacunas (*fillblanks*), podendo o aprendiz também informar que não sabe responder tal questão [38].

Um ponto forte no Flexi-OLM é a forma de apresentação do modelo do aprendiz aberto. O aprendiz pode escolher a forma de representação de seu modelo, entre eles: hierarquicamente, através de mapas conceituais, em texto corrido (leitura), apresentando seus pré-requisitos, na forma de sumário e ou de índice. Além disso, o ambiente permite que o aprendiz veja seu modelo inferido de acordo com as questões que ele fez e, caso o aprendiz não concorde com o que está sendo apresentado, ele pode discordar, pedindo para que o sistema argumente o porque daquela modelagem.

O módulo de negociação entre o sistema e o aprendiz em relação a sua modelagem se dá da seguinte forma: tem-se a modelagem do aprendiz que o sistema inferiu; o aprendiz pode discordar do sistema informando um novo *status* de conhecimento sobre aquele determinado assunto; quando o aprendiz informa um nível maior do que o que o ambiente inferiu, o ambiente apresenta a ele as questões que ele errou, argumentando, assim, o porque de ele estar com aquele *status* de aprendizagem. Caso o aprendiz continue afirmando que deveria estar em um *status* de conhecimento maior, é pedido para que ele resolva as questões que ele errou. Assim, se o aprendiz resolver tais questões corretamente, sua modelagem é modificada para um *status* de conhecimento maior [36].

A negociação da modelagem aberta do aprendiz está presente tanto no Flexi-OLM, o qual até permite várias formas de representação, quanto no FraW. A diferença entre os dois ambientes educacionais é que o FraW é um STI, que faz uso, também, da aprendizagem colaborativa, enquanto que no

Flexi-OLM isso não é encontrado. Além disso, ambos focam diferentes domínios, um a linguagem de programação C++ (Flexi-OLM), e o outro operações com fração (FraW).

3.5 WebStatisTutor

Em [21, 20] é apresentado o WebStatisTutor, o qual tem uma arquitetura para STI que é apoiada pelos fundamentodas da Web Semântica. Neste trabalho houve a preocupação em ser usado realmente a Web Semântica, e não, como em muitos trabalhos que foram apresentados, apenas o uso de algumas tecnologias que a Web Semântica faz, como ontologias. Em muitos trabalhos há o uso de ontologias apenas como uma forma de manter seus dados, ou tal tecnologia nem é, de fato, usado na Web.

A arquitetura proposta por este trabalho está apresentada na Figura 3.6, a qual é contruída com ontologias (que não serão apresentadas aqui) e há interação entre seus módulos. A descrição de seus módulos é a seguinte:

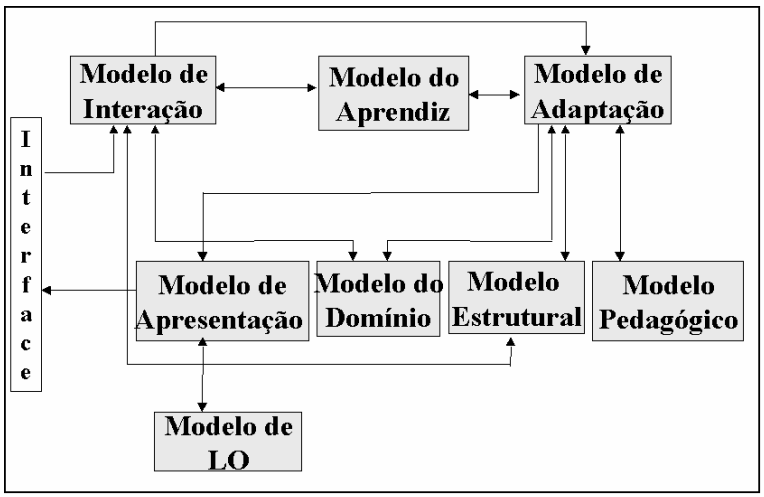

Figura 3.6: Arquitetura para STI apoiada por fundamentos da Web Semântica [21]

- Interface: interface *Web* através da qual o aprendiz irá interagir;
- Modelo de Interação: identifica padrões de comportamento e intenções do aprendiz;
- Modelo do Aprendiz: contém as informações sobre o aprendiz;
- Modelo de Adaptação: trabalha com um conjunto de regras, axiomas e heurísticas para construir uma decisão, a qual é tomada de acordo com

informações contidas nos Modelos do Aprendiz, de Interação, Pedagógico, Estrutural e de Domínio;

- Modelo de Apresentação: responsável por compor a informação que será apresentada ao usuário;
- Modelo do Domínio: representa o tema a ser abordado na aplicação;
- Modelo Estrutural: especifica de que forma os conceitos contidos no Modelo do Domínio serão agrupados dentro de unidades semânticas/de aprendizagem;
- Modelo Pedagógico: especifica as estratégias de aprendizagem (cada uma resultante de uma regra de negócio) da aplicação;
- Modelo de LO (*learning object* - objeto de aprendizagem): referente à modelagem dos objetos de aprendizagem.

O grande foco do WebStatisTutor é dado ao conteúdo que será apresentado ao aprendiz, ou seja, aos objetos de aprendizagem. Porém, ambos os trabalhos, WebStatisTutor e o FraW, trabalham com seus módulos definidos por ontologias, tendo o FraW também o mapeamento de seus agentes em ontologias, além de sua modelagem aberta do aprendiz e recomendação de um outro aprendiz.

3.6 Conclusão

Dentre as soluções apresentadas para a modelagem aberta do aprendiz (MAA), negociação da MAA, aprendizagem colaborativa em pares, aprendizagem de temas matemáticos e de sistemas que fazem uso da Web semântica, foram selecionadas e descritas aqui apenas aquelas que tinham uma significativa relação com o contexto deste trabalho. Com isso, pode-se ter uma visão geral de como vem sendo tratada a problemática em questão. Na Tabela 3.1, tem-se um resumo das características dos ambientes educacionais descritos neste capítulo, juntamente com as da dissertação em questão (FraW).

O conhecimento acerca das soluções apresentadas neste capítulo trouxe uma melhor consciência sobre a aplicabilidade de técnicas e conceitos voltados para efetivar o processo de aprendizagem. Com o conhecimento das similaridades e diferenças desta dissertação com os trabalhos apresentados, tem-se uma maior clareza acerca do diferencial deste trabalho, além do reconhecimento de sua contribuição voltada ao ensino-aprendizagem [14].

	I-Help	Aplusix	ActiveMath	Flexi-OLM	FraW
Diferentes perfis de usuário	-	-	-	-	x
Envolv. Pares Humanos	x	-	-	-	x
Envolv. Pares Computacionais	x	-	-	-	x
Facilidade de Evolução	-	-	-	-	x
Matemática	-	x	x	-	x
Modelo Aberto do Aprendiz (MAA)	-	x	x	x	x
Negociação MAA	-	-	-	x	x
Recomendação de Pares	x	-	-	-	x

Tabela 3.1: Comparação entre as funcionalidades dos ambientes: I-Help, Aplusix, ActiveMath, Flexi-OLM eo FraW

Capítulo 4

Especificação dos Requisitos e Projeto da Interface Gráfica

O FraW é um sistema tutor inteligente que permite a aprendizagem em pares, voltado ao ensino de fração para os aprendizes das séries iniciais de matemática. Há vários tipos de interação a qual o aprendiz pode atuar, como pode ser visto na Figura 4.1, nas Seções abaixo deste capítulo cada interação será melhor apresentada.

4.1 Caso de Uso 1: Aprendiz se registra no sistema FraW

O aprendiz que irá começar seus estudos no FraW e ainda não tem cadastro no sistema, deve se inscrever e inserir informações como: login, senha e *e-mail*, e depois efetuará o login, como apresentado na Figura 4.2. Após o cadastro, o aprendiz poderá começar suas aulas sobre fração, sendo apresentado primeiro o conteúdo a ser estudado e, posteriormente, uma lista de questões por fazer, isso para cada Currículo que o aprendiz for estudar, ou seja, para cada estudo de expressão com frações.

4.2 Caso de Uso 2: Aprendiz propõe um problema para que o sistema FraW resolva e explique

Em todo Sistema Tutor Inteligente o aprendiz tem a oportunidade tanto de resolver as questões propostas pelo tutor, quanto de apresentar alguma questão para que o STI responda e explique ao aprendiz cada passo da resolução daquela questão. É esta última situação que aqui é apresentada, onde o aprendiz

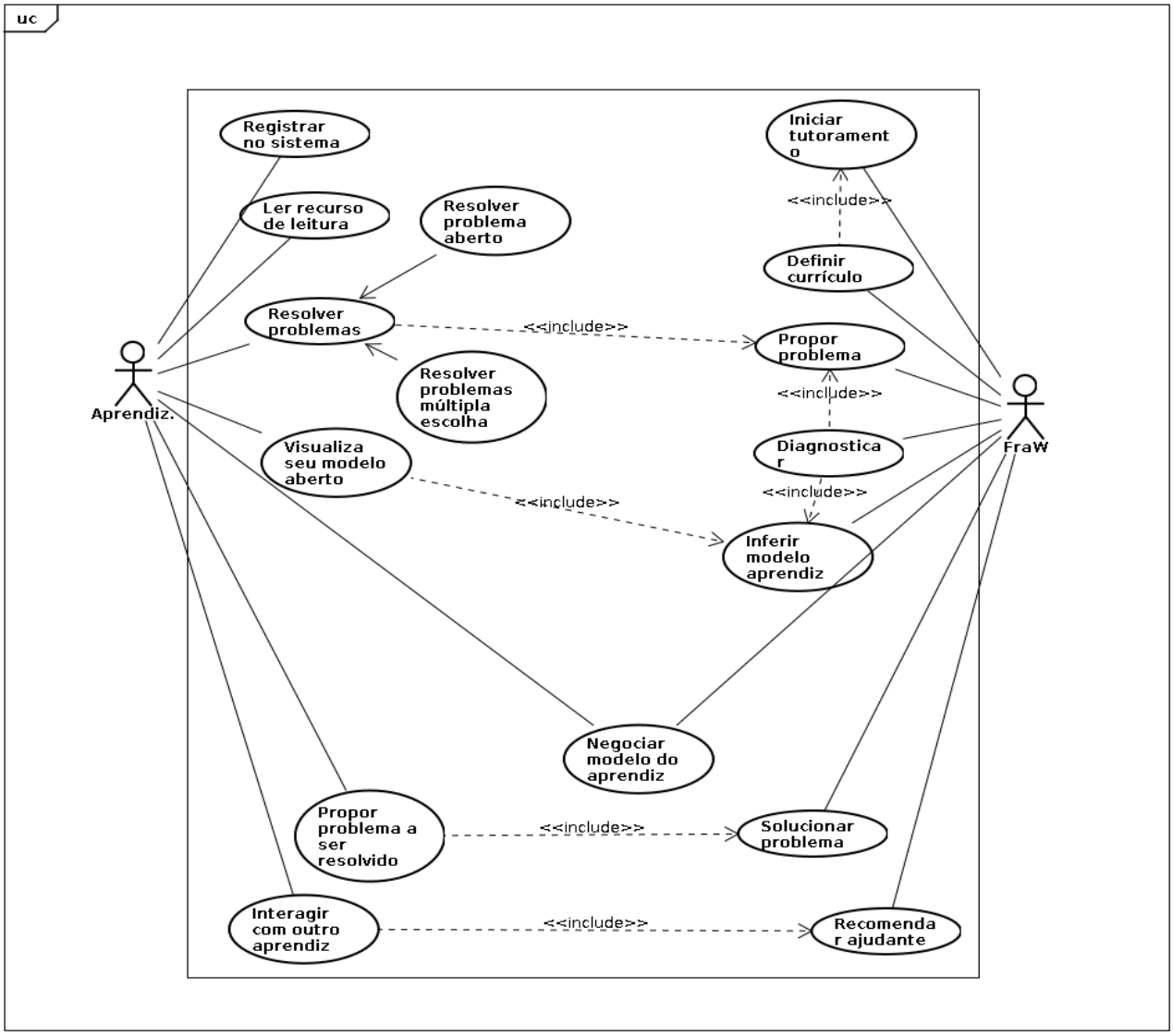

Figura 4.1: Diagrama de casos de uso do FraW

insere uma questão para que o STI, através de seus Tutores, possa resolver e explicar.

Como pode ser visto na Figura 4.3, a interface que é apresentada ao aprendiz para que ele possa postar suas questões é de fácil uso para Frações. No exemplo, a questão que o aprendiz inseriu foi a seguinte: $\frac{1}{2}+(\frac{3}{4}-\frac{2}{3})$. Nas Figuras 4.4, 4.5, 4.6 e 4.7 é apresentado a continuação da resolução e explicação da questão, contendo também a resolução de MMC (Mínimo Múltiplo Comum).

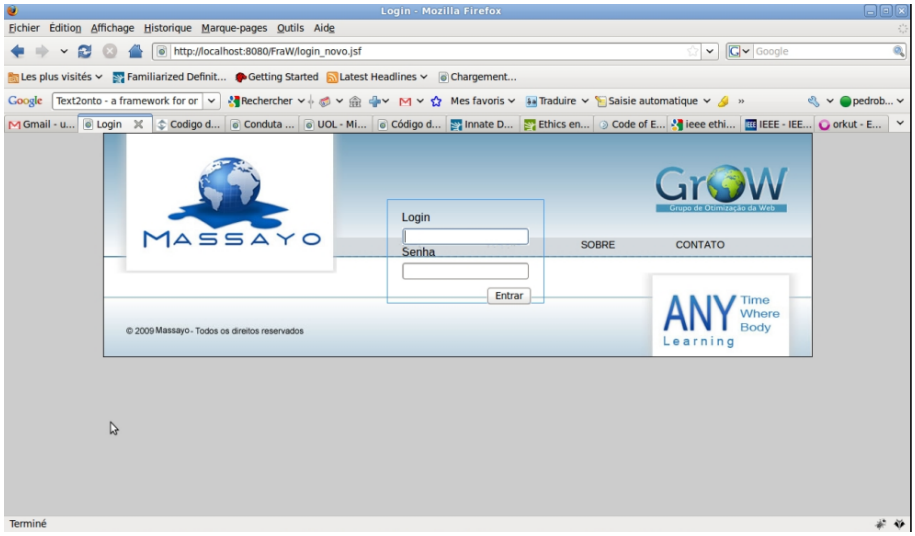

Figura 4.2: Tela de login do Massayo, o qual dará acesso ao FraW

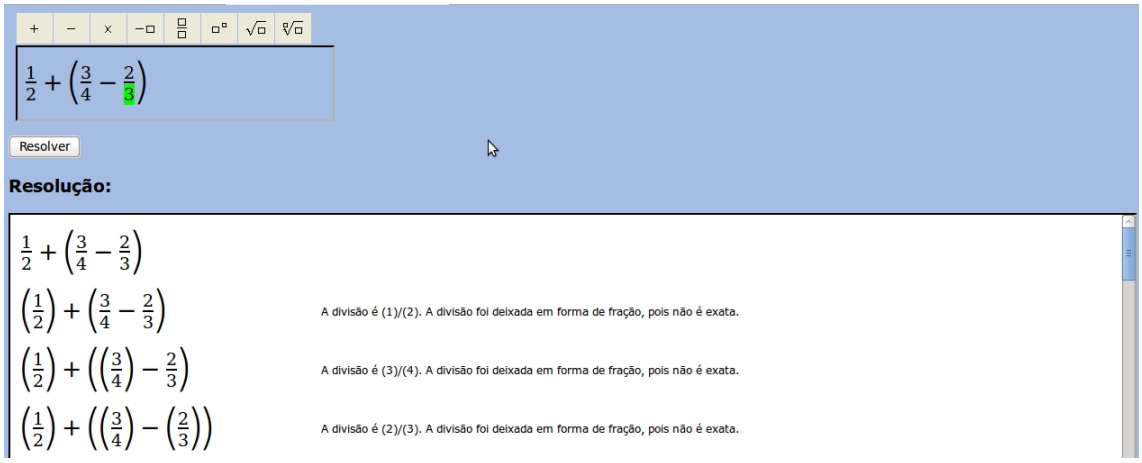

Figura 4.3: Resolução e explicação pelo STI da questão $\frac{1}{2}+(\frac{3}{4}-\frac{2}{3})$ postada pelo estudante

4.3 Caso de Uso 3: O sistema FraW propõe uma questão para que o aprendiz resolva

Neste cenário, temos o seguinte: o FraW apresenta ao aprendiz uma questão para que ele resolva passo-a-passo. A vantagem de se ter a resolução passo-a-passo do aluno é saber exatamente qual a deficiência do aluno em relação à questão, caso ele tenha alguma, já que teremos o conhecimento do passo que ele está errando. Como já foi dito, caso o aprendiz erre a questão, ele receberá uma dica sobre como resolver tal questão. Se o erro ainda persistir, o aprendiz receberá a recomendação de um outro aprendiz para que este último possa ajudá-lo a superar tal problema.

Na Figura 4.8 temos a questão $1x3+2+\frac{5}{7}$ que foi proposta pelo FraW para

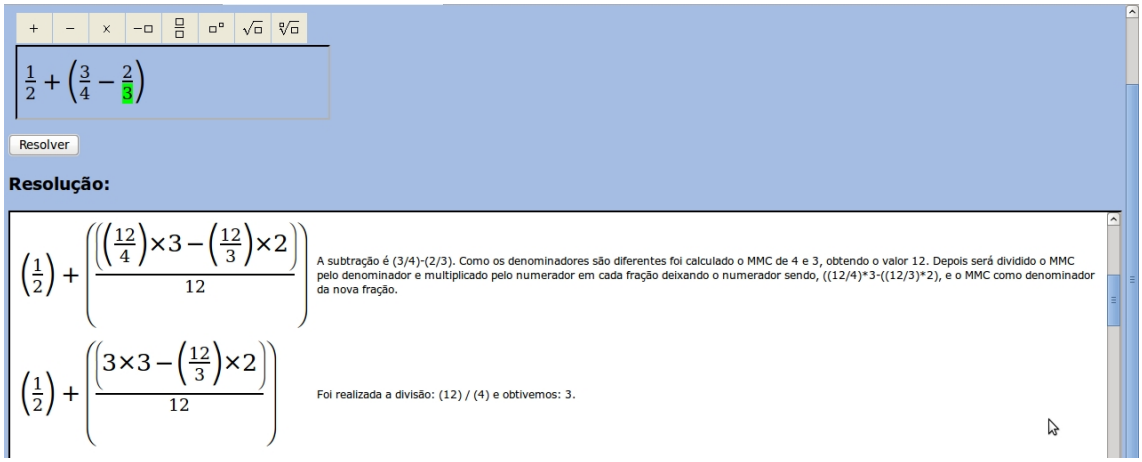

Figura 4.4: Continuação da resolução e explicação pelo STI da questão postada pelo estudante, realizando um MMC

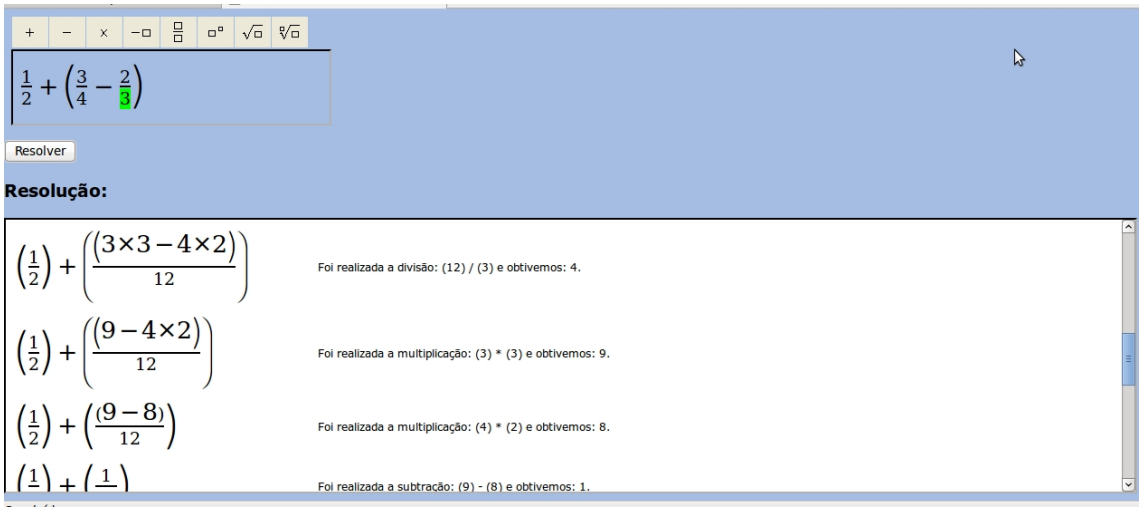

Figura 4.5: Continuação da resolução e explicação pelo STI de uma questão postada pelo estudante

que o aprendiz a resolvesse. Abaixo, na tela, como se pode ver, tem-se os termos e o operador que o aprendiz deve escolher. Caso o aprendiz escolha um termo ou operação errada para responder naquele momento, o sistema informa sobre o erro.

Na Figura 4.9 pode ser observado que o aprendiz escolheu o valor "1"como termo 1, o operador "*"e o, como o termo 2, o valor "3". E, na Figura 4.10, o estudante vai inserir a resposta da operação que ele escolheu, no caso, o valor "3".

Na Figura 4.11, pode-se ver a escolha de mais um termo, operador e termo e, na Figura 4.12 a resposta "5", para o "3+2".

Na Figura 4.13 o aprendiz escolheu como termo 1 o valor "5"e como termo 2 o valor "$\frac{5}{7}$", onde a ordem seria indiferente, e, como operador, "+". Na Fi-

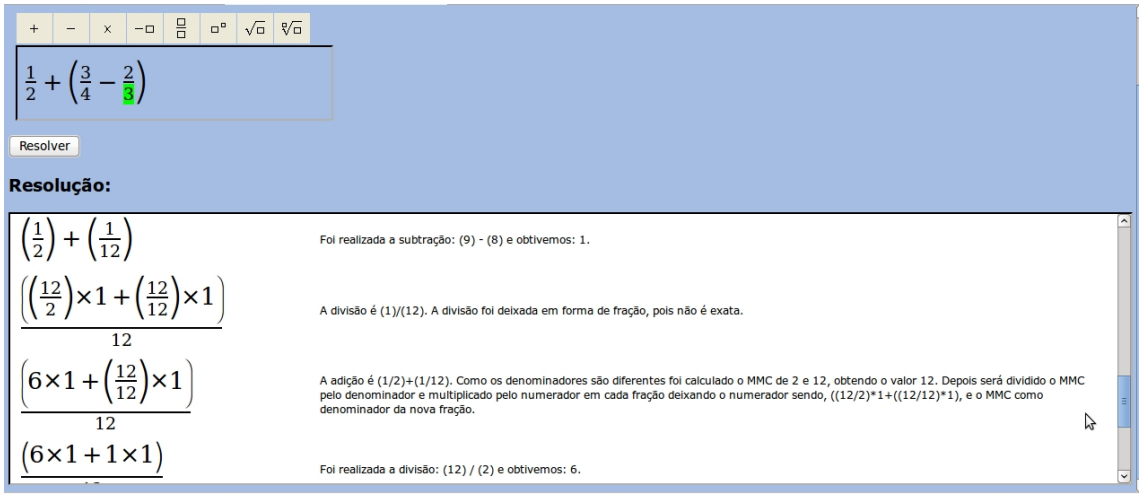

Figura 4.6: Continuação da resolução e explicação pelo STI de uma questão postada pelo estudante, realizando um outro MMC

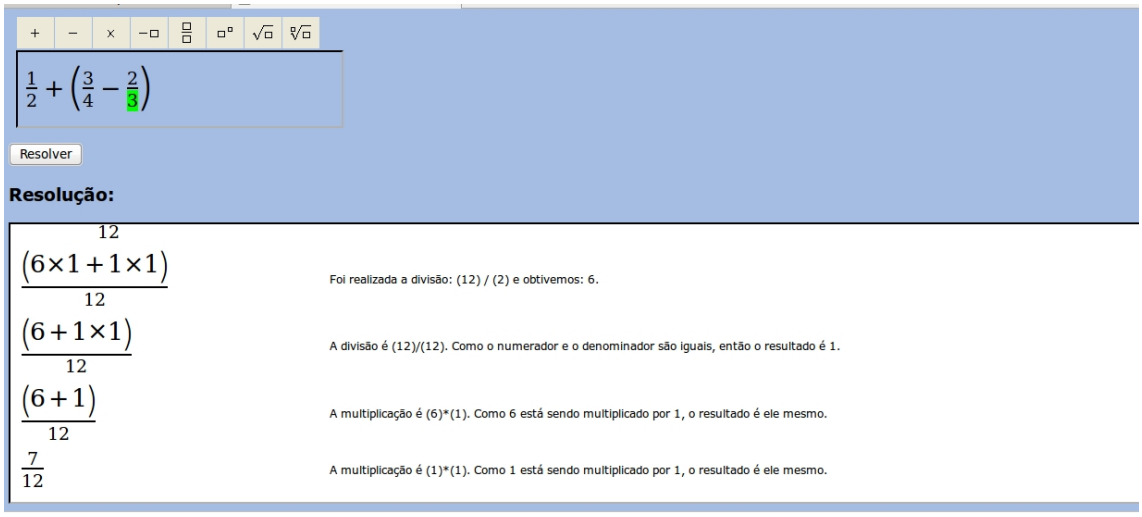

Figura 4.7: Conclusão da resolução e explicação pelo STI da questão postada pelo estudante

gura 4.14 o aprendiz insere a resposta para a operação escolhida: "$\frac{40}{7}$". E, finalmente, na Figura 4.15 é encontrado a resposta final para aquela questão.

4.4 Caso de Uso 4: Modelagem do Aprendiz apresentado e sua negociação

O aprendiz solicita ao FraW a modelagem de seu aprendizado, então, o agente Mediador faz a coletage das informações do aprendiz no banco de ontologias. Quando o FraW retorna ao aprendiz sua modelagem, Figura 4.16, apresentando os currículos que ele já estudou e qual o *status* de conhecimento dele

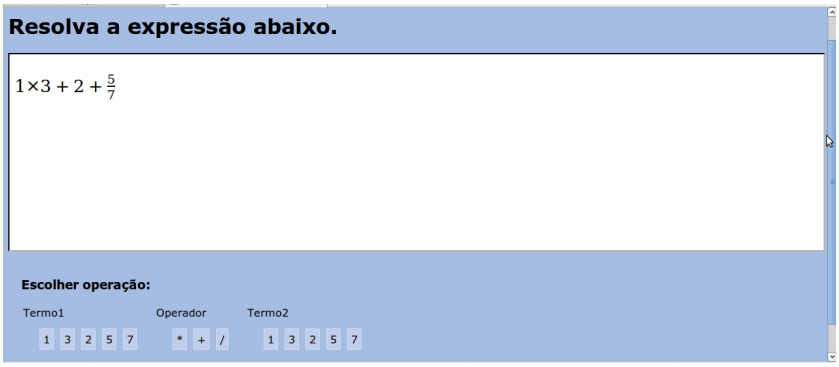

Figura 4.8: FraW propõe uma questão para que o aprendiz resolva eh aquela velha história de q sempre se tem o ônus e o bônus

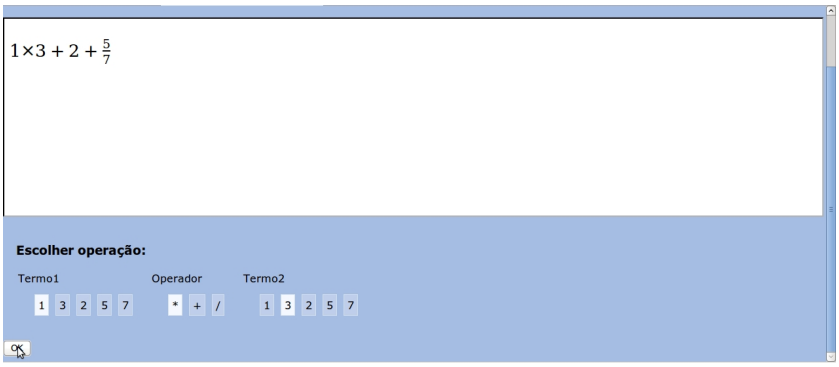

Figura 4.9: Escolha pelo aprendiz sobre a operação a ser resolvida

em cada um, o aprendiz pode concordar ou não com o que é apresentado pelo sistema. Caso ele não concorde, o sistema irá apresentar para ele quais questões ele errou e se mesmo assim o aprendiz ainda não aceitar a modelagem feita sobre ele, o FraW apresenta ao aprendiz algumas novas questões sobre o Currículo em debate. Caso o aprendiz acerte tais questões, sua modelagem será mudada para um *status* de aprendizagem maior, caso não, o *status* de aprendizagem permanece o mesmo.

4.5 Caso de Uso 5: Aprendiz necessita de ajuda

Um aprendiz está interagindo com um Tutor de um determinado Currículo e durante a resolução de um determinado problema, ele erra um passo. O Tutor, observando o erro, apresenta ao aprendiz uma dica sobre a resolução da questão para que ele não caia no erro novamente. Mas caso o aprendiz novamente cometa um erro nesta mesma questão, o Módulo Pedagógica do Tutor irá fazer a busca por um outro aprendiz que possa ajudá-lo com suas dúvidas e equívocos. A busca pelo aprendiz que irá ajudar o aprendiz que cometeu os erros será a partir da análise das modelagens feitas pelo sistema

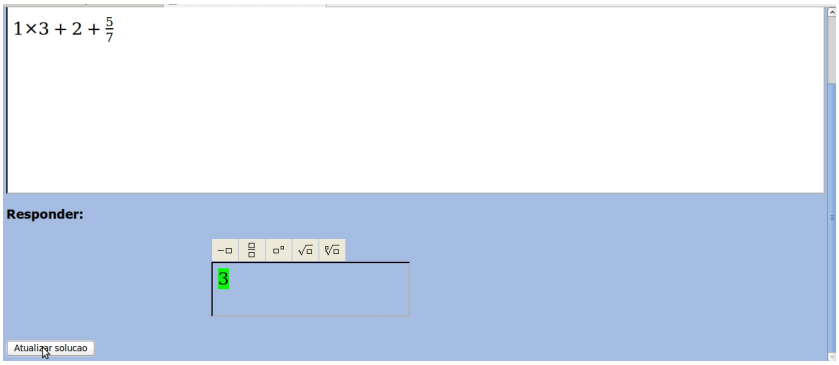

Figura 4.10: Resposta do aprendiz à operação escolhida por ele

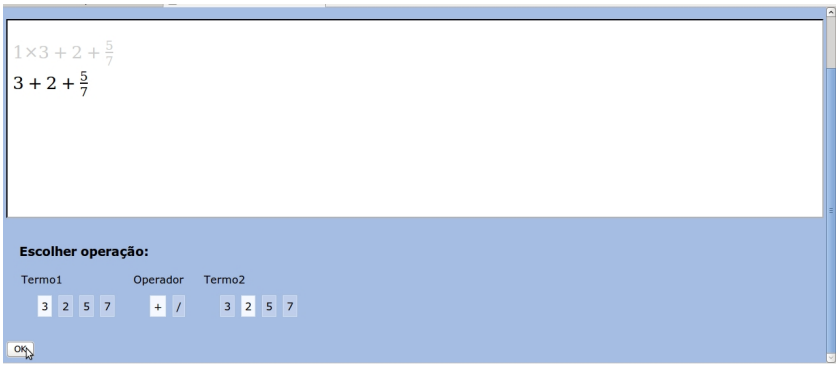

Figura 4.11: Próxima escolha da operação a ser resolvida

sobre os outros aprendizes inseridos no sistema.

Quando o sistema trouxer uma recomendação para o aprendiz a ser ajudado, será aberta uma janela de *chat*, como pode ser visto na Figura 4.17, entre os dois aprendizes e, enquanto eles conversam, um serviço de filtragem de conversa irá observar se eles estão realmente conversando sobre o problema em questão (fazendo uma pesquisa nas palavras-chave na ontologia daquele currículo).

4.6 Caso de Uso 6: Adaptação da Interface Gráfica

Acerca da interface gráfica, através das telas apresentadas pode-se perceber o quão confortável é para o aprendiz aprender fração no FraW, visto que o aprendiz poderá fazer manipular as frações da mesma forma que ele faz no caderno. Além disso, a possibilidade de resolução passo-a-passo pelo aprendiz e o acompanhamento do tutor para informar se o passo dado foi correto ou não é de grande ajuda na aprendizagem do aprendiz. No caso do aprendiz errar um passo, o FraW mantém uma base de regras de erros comuns, o que possibilita identificar um erro comum que o aprendiz pode cometer e, assim,

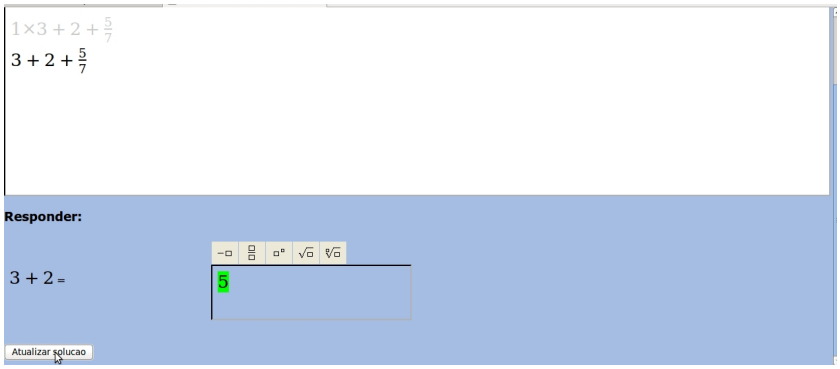

Figura 4.12: Resposta do aprendiz à nova operação escolhida por ele

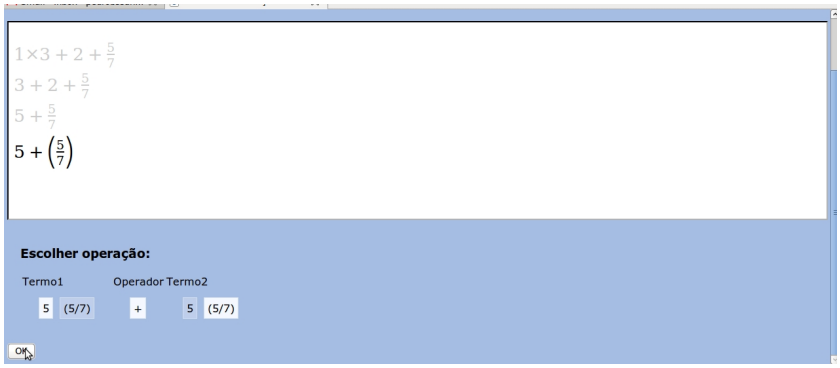

Figura 4.13: Escolha da operação a ser resolvida pelo aprendiz

dar uma dica baseada nessa informação.

Além disso, o FraW pretende dar a opção do ambiente aprentar a barra de entrada de fração de forma adaptada a questão, para que o aprendiz não use a entrada de raiz quadrada, ou de potencia, quando a questão não necessita disso para ser resolvida.

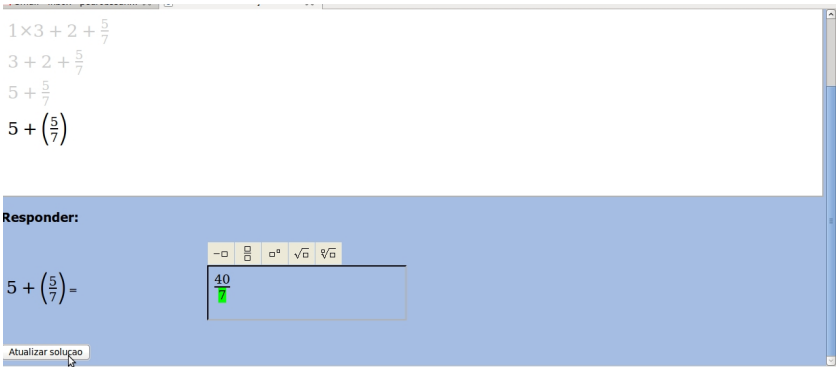

Figura 4.14: Resposta do aprendiz à operação escolhida por ele

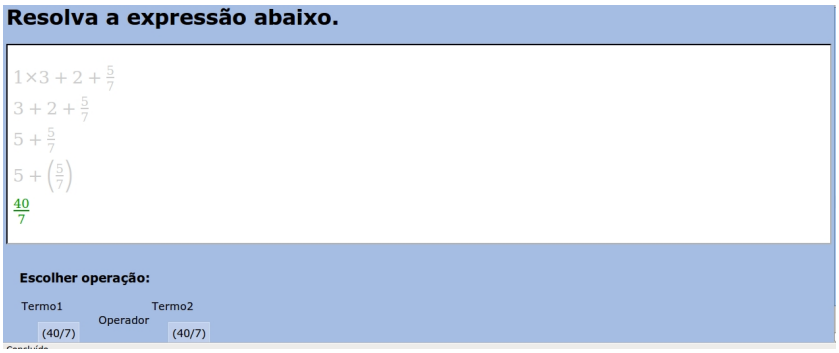

Figura 4.15: Final da resolução da questão proposta ao aprendiz pelo FraW

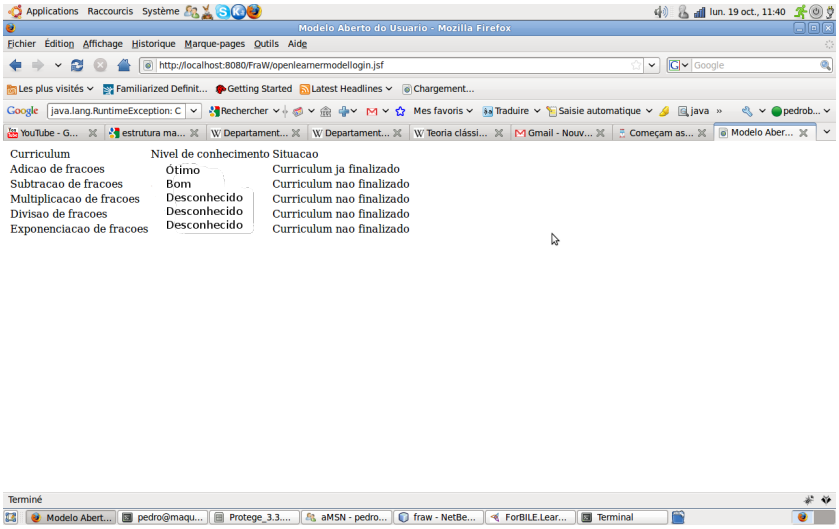

Figura 4.16: Modelagem Aberta do Aprendiz

Figura 4.17: Janela de *chat* entre os pares-complementar

Capítulo 5

O Sistema FraW e os Modelos Associados

O presente trabalho se situa no contexto do desenvolvimento do sistema FraW (Fração na *Web*), o qual corresponde a um Sistema Tutor Inteligente multiagente baseado no modelo Mathema [25], adotando a resolução de problemas como sua atividade pedagógica básica. Neste sentido, o aprendiz humano e o sistema cooperam a fim de chegar a uma solução de um problema posto mas focalizando todo o processo de solução. O sistema FraW, portanto, baseia-se no modelo Mathema, notadamente quando se considera a modelagem multiagente do domínio.

O sistema FraW, conforme Figura 5.1 está organizado da forma que se tem um aprendiz que pode interagir com uma sociedade de agentes ou com algum outro aprendiz, permitindo, assim, a cooperação entre o aprendiz e agentes computacionais ou humanos durante a resolução de problemas associado ao domínio de fração.

Como já foi dito, o sistema FraW tem como foco o ensino do domínio de fração e objetiva ajudar alunos das séries inicias do ensino básico na aprendizagem deste domínio. Esta aprendizagem é baseada na resolução de problemas e verificação do desempenho do aluno, através da correção da resolução de seus problemas. Este ambiente está habilitado a resolver problemas propostos pelo estudante, explicando cada passo utilizado na solução; assim como avaliar as soluções apresentadas pelo aluno para problemas propostos pelo sistema, oferecendo ajuda nos eventuais impasses no processo de elaboração da solução. A ajuda está fundamentada em apontar o erro exato cometido por ele e fazer a apresentação de dicas específicas para o tópico ao qual o aluno está com dificuldade. Caso o aprendiz, mesmo após a dica, persista no mesmo erro, a recomendação de um outro aprendiz para ajudá-lo é feita.

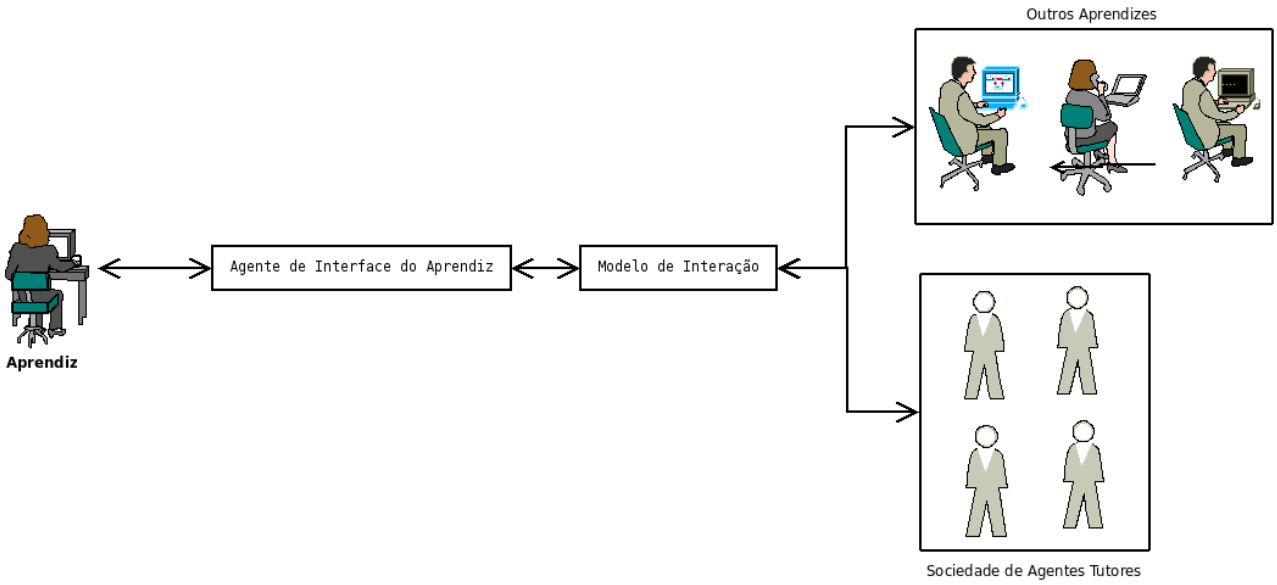

Figura 5.1: Visão de alto nível do FraW

Como todo STI, o FraW mantém a modelagem de cada aprendiz, possibilitando a cada um a visualização do seu perfil (modelagem) que foi criada pelo ambiente a partir da interação do aprendiz com este durante a resolução de questões. Observando o aprendiz o que foi modelado sobre ele, este pode concordar ou discordar. O segundo caso ocasionará uma negociação entre o aprendiz e o STI acerca da melhor informação a ser deixada como seu perfil.

5.1 Arquitetura do FraW

A arquitetura do FraW está apresentada na Figura 5.2, de forma abstrato/-conceitual, enquanto sua visão lógica é apresentada na Figura 5.4.

Observa-se inicialmente na Figura 5.2 a camada de Interface *Web*, a qual o aprendiz usará para interagir com o ambiente. Esta interface deve ser confortável e de fácil uso para o ensino matemático. As telas de comunicação do FraW foram feitas de forma que seus aprendizes pudessem inserir e ver suas questões e respostas da mesma forma que eles o fazem no caderno, como exemplo, os estudante poderão inserir '$\frac{1}{2}$' ao invés de '1/2' [24]. Sendo esta uma interface *Web*, o aluno poderá acessar o sistema em qualquer computador que tenha acesso a Internet. Detalhes sobre a Interface serão melhor apresentados na Seção 5.4.

Pode-se notar que a Interface está ligada ao *GatewayAgent*, o qual faz a interface entre a Interface gráfica e os outros agentes: Mediador e Tutores. O

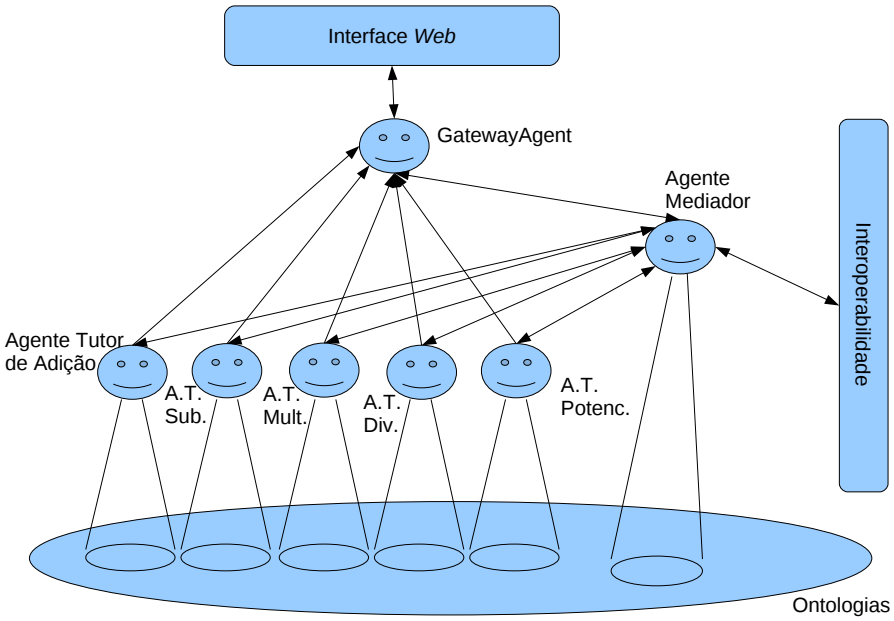

Figura 5.2: Visão lógica da arquitetura do FraW

uso deste *GatewayAgent* é aconselhado pelo Jade [1] (*Java Agent DEvelopment Framework*) para que os agentes funcionais (que executam alguma função) não interajam diretamente com a Interface gráfica.

O agente Mediador (Seção 5.3) faz a interação entre as entradas do aprendiz e os agentes Tutores, ou seja, é responsável por coordenar as informações recebidas pelos alunos e pelos agentes compostos na arquitetura e propagar para seu determinado receptor. É ele quem, caso o aprendiz apresente uma questão para que o sistema explique, irá dividir para cada agente Tutor sua parcela de resolução, além de unir as respostas para apresentar ao aprendiz. Também é o agente Mediador que recebe dos agentes Tutores as questões e material de estudo que serão passados ao aprendiz.

A camada de Interoperabilidade com a qual o Mediador se comunica é usada para a interação deste ambiente com outros ambientes multiagentes e serviços. Como o ambiente do FraW faz uso de serviços semânticos, tais serviços são especificados em ontologias e dispostos através da camada de Interoperabilidade para que, quem quiser fazer uso deles, poder usá-los, e os outros ambientes poderem também especificar seus serviços para que o FraW e outros ambientes possam fazer uso deles.

Já os agentes Tutores (Seção 5.2) são responsáveis pelo ensino de Adição, Subtração, Multiplicação, Divisão e Potenciação de Fração, são eles que, seguindo a estrutura padrão da arquitetura básica dos STI [26], mantém in-

[1] http://jade.tilab.com/

formações sobre o aprendiz, têm a modelagem do domínio a ser passado (como material de leitura e questões a serem propostas e resolvidas) e têm as estratégias de ensino que guiará o ensino do aprendiz.

Em relação aos agentes Tutores (pode ser visto na Figura 5.3), cada um é composto por esses 3 módulos básicos que formam um STI [26]:

- No Modelo do Aprendiz encontram-se informações sobre o aluno e a respeito do seu aprendizado. Podemos encontrar as questões por ele resolvidas, quais questões errou, quais acertou, suas dificuldades em determinados tópicos de estudo, entre outras informações pessoais e estudantis. Este modelo caracteriza individualmente cada estudante no seu processo de aprendizado. Sendo este um Modelo Aberto do Aprendiz, o estudante poderá ver qual foi a avaliação inferida pelo sistema sobre seu aprendizado, se o mesmo observar que o sistema o acusa de não saber determinado conhecimento que ele considera saber, ele poderá contestar esta afirmação, mas para que o estado de sua avaliação mude ele terá que provar para o sistema que sabe determinado tópico [23].

- No Módulo do Domínio estão as informações a respeito do que será passado para os alunos. No trabalho aqui apresentado o domínio corresponde a assuntos relacionados a frações com números inteiros. Tendo o FraW uma abordagem pedagógica de aprendizagem baseada em resolução de problemas, aqui tais problemas são expressões que podem conter as seguintes operações: soma, subtração, multiplicação, divisão e potenciação. Além de questões, o FraW também apresenta conteúdo para que o aluno estude sobre cada uma desta expressão.

  Este Módulo também é responsável por permitir que a estratégia de ensino seja separada por questões sobre um determinado tópicos (Currículos) e grau de dificuldade. É neste módulo onde também se encontram duas funcionalidades importantes: um módulo resolvedor de problemas e um módulo para a avaliar a solução de problemas. O primeiro módulo é usado quando o aluno passa uma expressão para que o sistema resolva. Quando o aluno vai resolver uma questão o segundo módulo é acionado [22].

- O Módulo Pedagógico é um módulo de gerenciamento em um STI. Sua função é coordenar o funcionamento do sistema. Esse Módulo contém estratégias de ensino, seleciona o conteúdo a ser apresentado ao aluno, monitora e critica o desempenho do aluno [53].

Dentro do Módulo Pedagógico há um bloco de ajuda ao aprendiz, caso este não esteja tendo um bom rendimento. Dentre as ajudas, tem-se dicas sobre uma determinada dificuldade que o aprendiz esteja passando e a recomendação de um outro aprendiz, para que este possa ajudá-lo em relação ao problema que ele está tentando resolver.

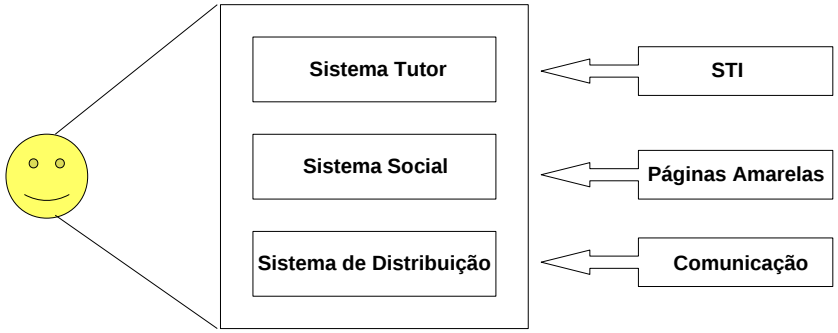

Figura 5.3: *Zoom* de um Agente Tutor (adaptado de [25])

A visão lógica da arquitetura do FraW está apresentada na Figura 5.4. Onde tem-se os componentes e conectores que interagem através de suas interfaces. Na parte superior da figura podemos ver que o componente de Interface com o Usuário requer serviços do componente *GatewayAgent* através do conector HTTP & Distribuição de Carga, o qual distribuirá acesso ao sistema em várias máquinas.

O conector Agente Facilitador faz a conexão entre o *GatewayAgent* e o banco de ontologias, pois o *GatewayAgent* passa uma requisição do aprendiz, a fim de localizar um dos Agentes Tutores irá atender. Uma vez colocado o tutor, o *GatewayAgent* pode interagir diretamente com ele através do conector P2P e também do *GatewayAgent* com o componente de Interoperabilidade (quando tal conector irá procurar por um serviço que ele estaja precisando e não tem em seu próprio ambiente).

Os Agentes Tutores buscam informações pedagógicas, de domínio e sobre os aprendizes diretamente do banco de ontologias, e também pode fazer uso dos Serviços do ambiente diretamente, além de poder procurar no componente

de Interoperabilidade por serviços extras que não contém em seu ambiente, através do conector de Localização de Serviços.

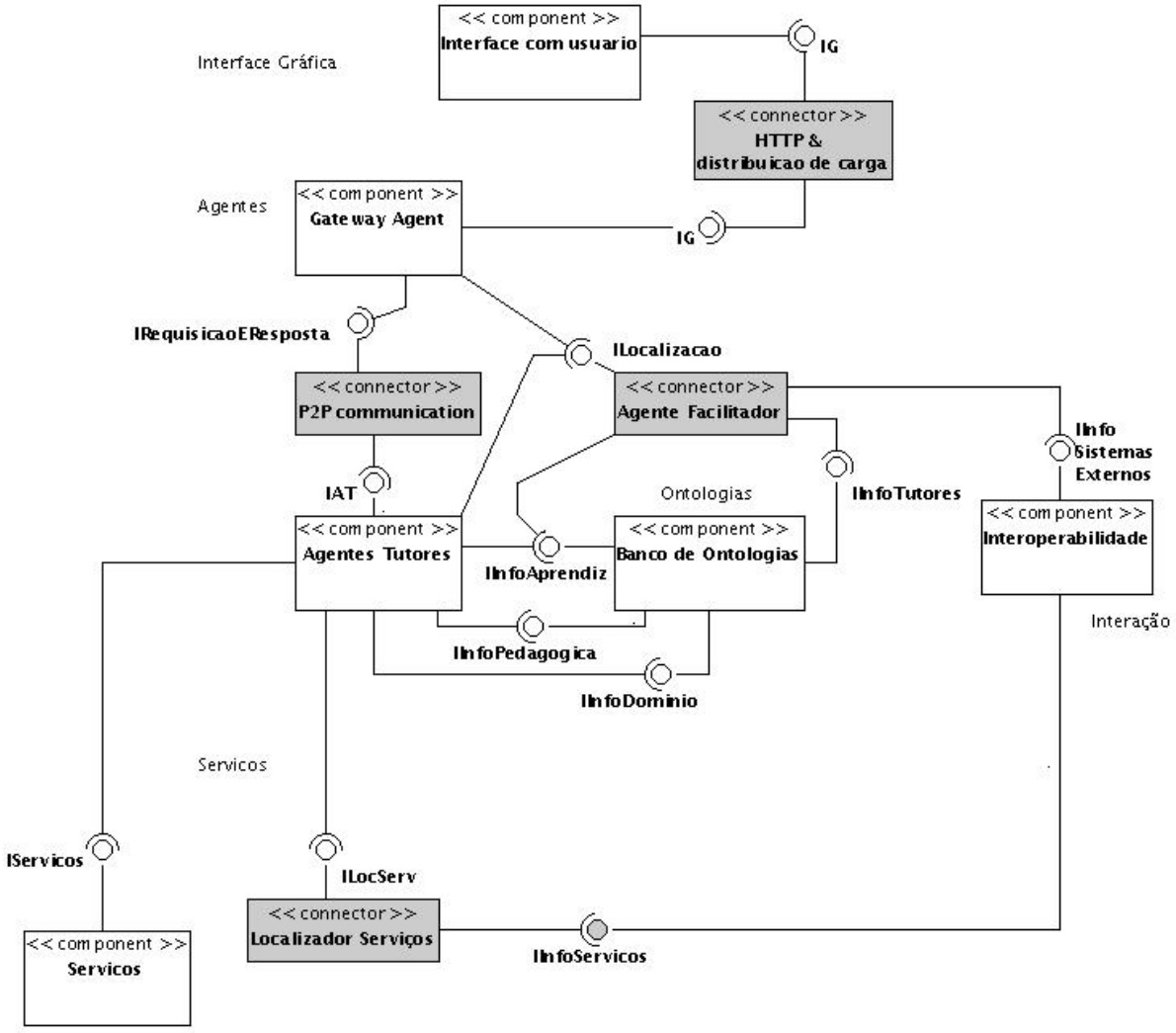

Figura 5.4: Visão lógica da arquitetura do FraW

A visão física simples do ambiente pode ser vista na Figura 5.5, onde temos as camadas de: Interface Gráfica, Agentes/Ontologias, Serviços, Interação e Sistemas Externos. Já na Figura 5.6 se pode ver a visão física com redundância, onde as camadas de Agentes/Ontologias e Serviços devem ser replicados para subsidiar os vários acessos dos aprendizes, possibilitando, assim, a escalabilidade.

A seguir, cada agente e camada da arquitetura aqui apresentada será detalhadamente apresentado.

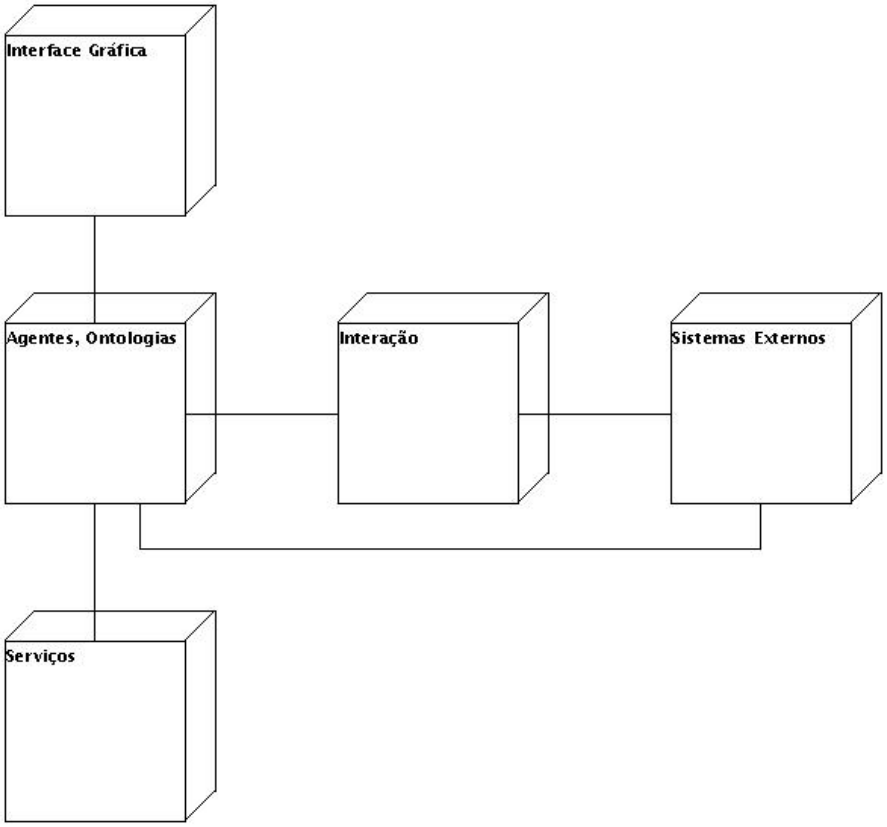

Figura 5.5: Visão física simples do FraW

5.2 Agentes Tutores

Existem cinco Agentes Tutores no FraW, são eles: Tutor de Adição, Tutor Subtração, Tutor de Multiplicação, Tutor de Divisão e Tutor de Potenciação. Cada agente Tutor está implementado da forma que apresenta a Figura 5.3, em que o Sistema de Distribuição, responsável pela comunicação deste agente com outros, e o Sistema Social, também conhecido como Páginas Amarelas, que é onde fica guardada as informações e localização dos outros agentes do ambiente, são responsabilidade do Jade [40].

A parte nomeada como Sistema Tutor da figura, está a parte de raciocínio de um agente. Portanto, é nesta parte onde se tem o acesso às ontologias (do Aprendiz, do Domínio e Pedagógico), inferências e resolução das questões. É nesta parte onde temos os três módulos já citados que serão melhor apresentados a seguir.

5.2.1 Modelagem do Aprendiz

A modelagem do aprendiz é adquirida através da interação do aprendiz com o sistema FraW, através do diagnóstico da resolução dos problemas, pelo apren-

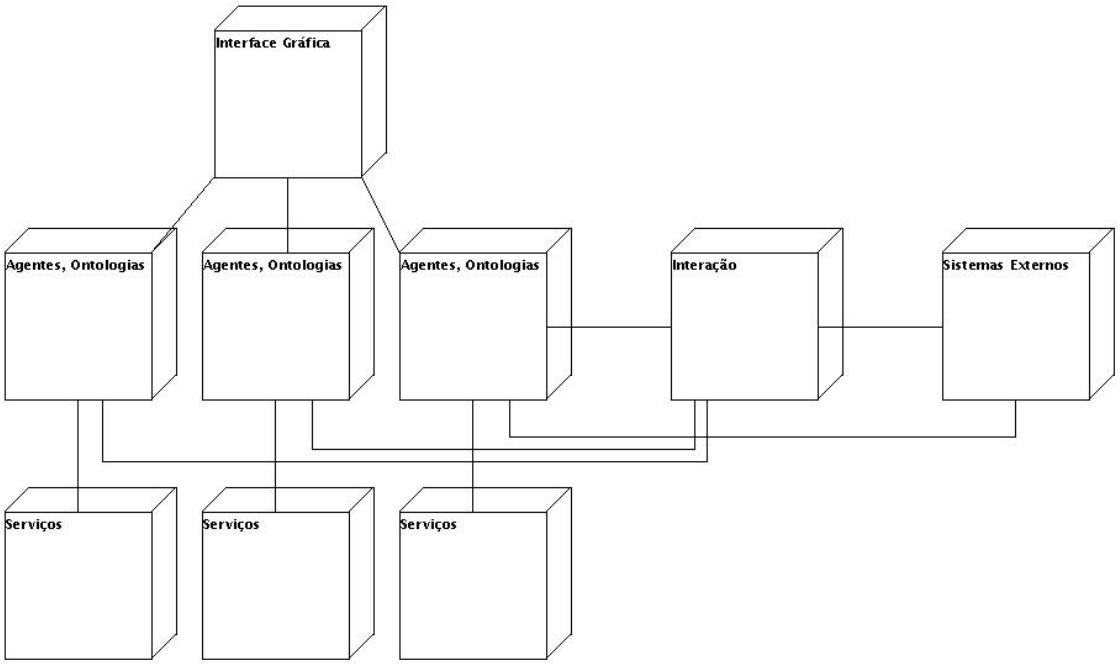

Figura 5.6: Visão física redundante do FraW

diz, o qual está melhor descrito em uma Subseção de 5.2.1. As informações que são armazenadas em tal modelo são acerca do estado cognitivo do aprendiz em cada tema, que será chamado de currículo, estudado sobre fração. Informações como: qual currículo o aprendiz já estudou ou está estudando, quais questões ele fez e quais ele errou, estão contidas em sua modelagem.

O módulo do aprendiz no sistema FraW foi feita seguindo uma taxonomia guiada pelo padrão IMS, que é um padrão usado para modelar o aprendiz [15], e modelada em ontologias. Nas ontologias utilizadas nessa dissertação estão estruturadas informações importantes relativas ao aprendiz. A especificação IMS define vários tipos de informações que devem ser obtidas do aprendiz de forma a ter um melhor conhecimento das capacidades, ações e intenções deste [54]. Neste trabalho, foram utilizados os seguintes tipos de informação:

- Identificação: dados pessoais do aprendiz;
- Objetivos: objetivos da aprendizagem, aspirações. Neste caso, sobre o aprendizado de fração;
- Atividades: processos de qualificações e/ou aprendizagem que o usuário está ou esteve envolvido. Neste trabalho, quais os tópicos (currículos) do domínio que o aprendiz já estudou e qual está estudando.

Os itens citados acima sobre o padrão IMS são importantes para este trabalho visto que dá base para que os questionamentos a seguir sejam respon-

didos:

- Qual o nome do aprendiz?
- O aprendiz é do sexo masculino ou feminino?
- Qual a idade do aprendiz?
- Qual o nível educacional do aprendiz?
- Quais tópicos (currículos) do domínio o aprendiz deve estudar?
- Quais tópicos (currículos) o aprendiz já estudou?
- Qual tópico (currículos) o aprendiz está estudando?
- Qual nível o aprendiz está no tópico (currículo) que ele está estudando?
- O que o aprendiz ainda não estudou?
- Qual o nível de dificuldade (iniciante, intermediário, avançado) do problema em questão?
- Quantas questões o aprendiz acertou em um determinado nível de um tópico (currículo)?
- Quantas questões o aprendiz errou em um determinado nível de um tópico (currículo)?

Assim, através das ontologias, o ambiente educacional pode interpretar semanticamente os conceitos definidos pelo padrão IMS [35].

O módulo do aprendiz foi criada aqui através de três ontologias, as quais formalizam o modelo do aprendiz com informações necessárias para se ter um conhecimento válido do aprendiz: informações pessoais, cognitivas e metas. As ontologias usadas aqui foram adaptadas de [6], que seguiu as determinações do padrão IMS [15], e é composto pelas seguinte ontologias:

- *Learner* - ontologia que contém, em sua maioria, informações que o usuário informa explicitamente, os quais o sistema não pode modificar, apenas o aprendiz. Na ontologia apresentada na Figura 5.7 temos as seguintes classes e seus respectivos atributos:
  - *User*: classe que caracteriza um aprendiz. Ela tem relação com a classe *Identification* através do atributo *hasIdentification* e que tem atributos como:

    - *rememberPassDescription*: descrição da senha, para que ela possa ser lembrada por essa descrição;
    - *registerDate*: data em que o aprendiz se cadastrou no sistema;
    - *login*: nome que o aprendiz usará para entrar no sistema;
    - *password*: senha do usuário no sistema.
- *Identification*: classe que guarda a identificação do usuário em diferentes pontos. Esses pontos são visto através desses atributos de relação que os liga à cada classe responsável:
    - *hasAddress*: faz referência para a classe *Address*. Esse atributo é uma lista, onde pode ser guardado não apenas informações sobre onde mora o aprendiz, mas outros endereços, como o de trabalho, entre outros;
    - *hasDemographics*: faz referência para a classe *Demographics*;
    - *hasContactInfo*: faz referência para a classe *ContactInfo*;
    - *hasName*: faz referência para a classe *Name*.
- *Address*: classe que mantém informações sobre onde o aprendiz mora, trabalha *etc*. Alguns atributos usados constantemente são:
    - *country*: país;
    - *statepr*: estado do país;
    - *city*: cidade;
    - *locality*: bairro.
- *ContactInfo*: classe que guarda os contatos do aprendiz, para cada contato podemos inserir uma lista de contatos daquele tipo, são eles:
    - *webSite*: site(s) que o aprendiz tem;
    - *mobilePhone*: número do(s) telefone(s) móvel(is) do aprendiz;
    - *telephone*: número do(s) telefone(s) do aprendiz;
    - *email*: *e-mail*(s) do aprendiz.
- *Demographics*: classe em que se encontram tais informações pessoais do aprendiz:
    - *dateOfBirth*: data de nascimento;
    - *gender*: sexo.
- *Name*: classe em que é armazenada o nome do aprendiz nos seguintes atributos:
    - *firstName*: primeiro nome;

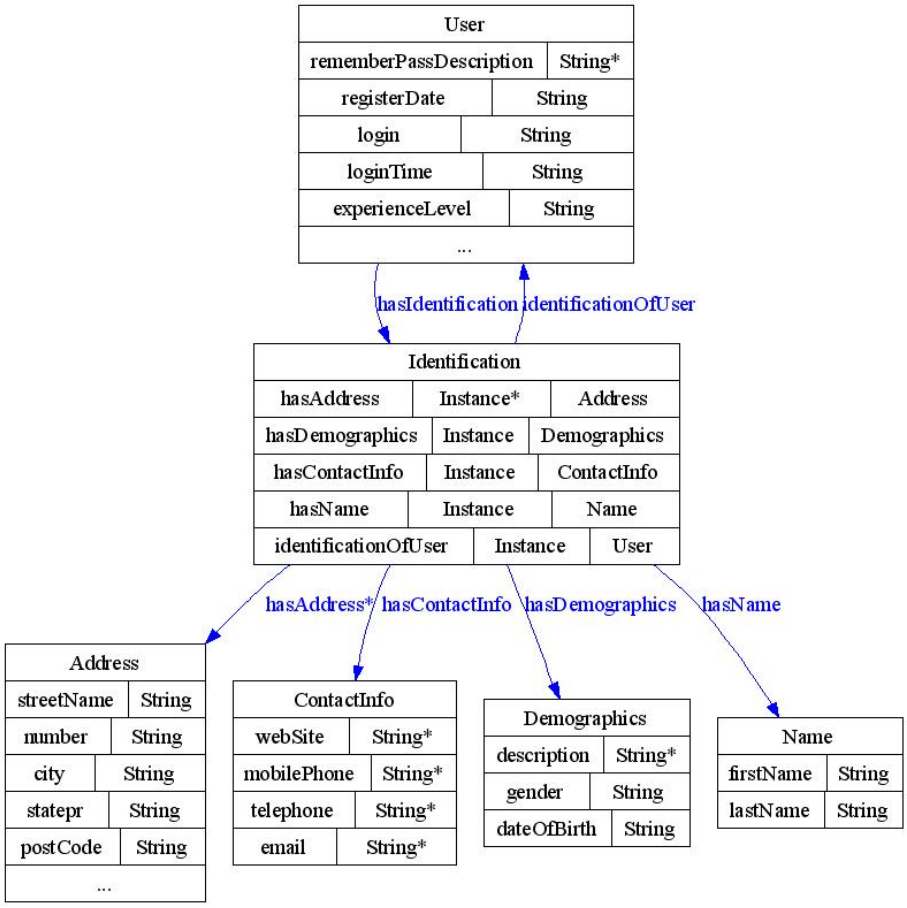

Figura 5.7: Ontologia *Learner*

  - *lastName*: último nome.

- *Interaction.Cognitive* (na Figura 5.8 é apresentada esta ontologia com uma instância) - é nesta ontologia que são depositadas informações sobre a interação do aprendiz com o sistema enquanto ele está aprendendo algum ramo (Currículo) da sub-divisão do domínio. O perfil do aprendiz sobre aquele assunto está sendo construído sem que ele note. Com essa ontologia são guardadas informações sobre o que o aprendiz está estudando (qual Currículo), o que o aprendiz já estudou e o que ele estudará assim que terminar o Currículo que ele está estudando. Esta ontologia também referencia uma ontologia do modelo de domínio. Segue algumas minúcias desta ontologia: a classe *CognitiveInformation* é composta por dois relacionamentos, como podemos ver na Figura 5.8, onde temos uma instância de tal classe para Maria: *hasUser* e *isComposedForInteractionLearnerData*, o qual segue maiores descrições.

  - *InteractionLearnerData* (5.9): classe que mantém dados sobre a interação do aprendiz durante a aprendizagem de um determinado Cur-

rículo. Esta classe contém as seguintes relações:

  * *isComposedForAggregatedData*: relação que compõe os dados sobre o processo de aprendizagem do aluno no Currículo que ele está estudando no momento. Contém os atributos:
    - *correctAnswerNumber*: quantidade de respostas corretas às questões propostas;
    - *incorrectAnswerNumber*: quantidade de respostas erradas às questões propostas;
    - *failureRate*: taxa de erro do aprendiz em relação às questões que ele resolveu até o momento;
    - *successRate*: taxa de acerto do aprendiz em relação às questões que ele resolveu até o momento.
  * *isComposedForProcessLearningPhase*: classe que compõe as informações sobre o que o aprendiz está aprendendo naquele momento. Esta classe tem os relacionamentos seguintes:
    - *hasCurriculumUnit*: Currículo que o aprendiz está estudando naquele momento;
    - *hasPedagogicalUnit*: Unidade Pedagógica que o aprendiz está estudando naquele momento. Lembrando que essa Unidade Pedagógica faz parte do Currículo que ele está estudando naquele momento;
    - *hasSequencing*: informa qual o tipo de sequência de atividade os aprendizes estão interagindo naquele momento;
    - *learnedCurriculums*: apresenta quais Currículos aquele aluno já estudou.

É nesta ontologia onde fica o atributo *questionsTrace*, onde fica armazenado quais questões o aprendiz fez e, para cada uma, informa se ele acertou-a ou não.

- *Goal* (Figura 5.10) - essa ontologia traz referência à ontologia *Learner*, e a uma ontologia do modelo do domínio. O uso das duas ontologias citadas se deve ao fato de nesta ontologia serem mantidas informações como nível de conhecimento e sua taxa de aprendizagem em um determinado Currículo que o aprendiz já estudou. As informações contidas nesta ontologia são armazenadas como forma de *log* para que o sistema saiba o aproveitamento da aprendizagem do aprendiz nos Currículos estudados. Alguns atributos dessa ontologia são:

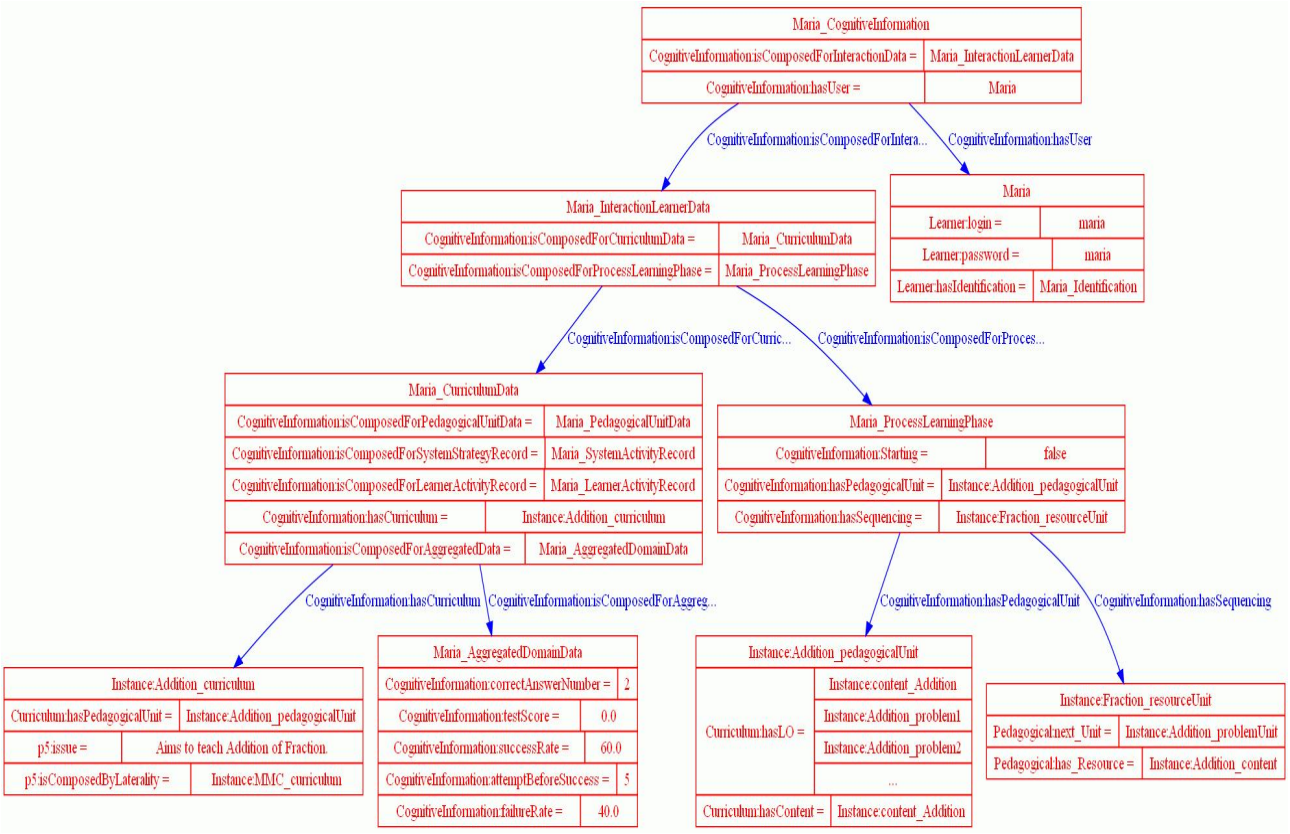

Figura 5.8: Instância a partir classe *CognitiveInformation* da ontologia *Learner.Interaction.Cognitive*

- *hasUser*: relacionamento que faz a ligação do aprendiz aqui descrito, com os resultados do aprendiz em um determinado Currículo;
- *hasCurriculum*: relacionamento que apresenta o currículo que o aprendiz estudou;
- *level_Of_Knowledge*: nível de conhecimento acerca de um currículo que o aprendiz quer chegar: avançado, básico ou intermediário;
- *rate_Of_Knowledge*: taxa de conhecimento que o aprendiz alcançou durante o estudo do Currículo descrito no relacionamente *hasCurriculum*;
- *desiredKnowledge*: taxa de conhecimento considerado como um bom aprendizado no Currículo descrito.

Como pode ser observado, as informações como a taxa de acertos de questões de um aprendiz, só será apresentado enquanto ele estiver estudando uma unidade pedagógica. Depois que o aprendiz já estudou um Currículo completo, a informação que será armazenada em seu modelo será um nível de aprendizagem que ele teve naquele Currículo, podendo ser classificado de acordo com a porcentagem de acerto de questões.

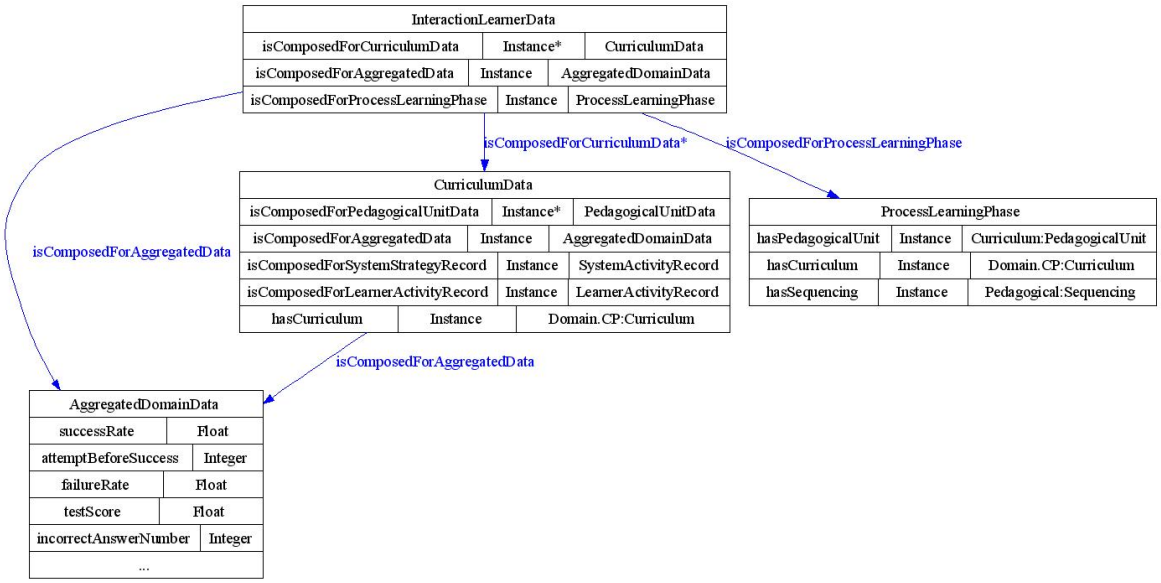

Figura 5.9: Ontologia *Learner.Interaction.Cognitive*

Diagnóstico

Essas duas últimas ontologias apresentadas mais acima (*Interaction.Cognitive* e *Goal*) são preenchidas pelas informações referentes ao resultado das interações do aprendiz com o tutor, o qual faz o diagnóstico do aprendiz. A interação do aprendiz com o sistema é gerenciado pelo agente mediador que transfere a requisição do aprendiz para os agentes tutores. Quando o aprendiz está estudando um determinado currículo, ele interage com um agente tutor que o guiará durante aquele currículo de aprendizagem. Em cada agente tutor há um raciocinador que faz inferência através de regras, que é responsável por duas formas de interação entre o tutor e o aprendiz: (i) quando o aprendiz resolve um problema de fração que o sistema propõe a ele e (ii) quando o aprendiz propõe uma questão para o STI resolver e explicar a ele sua resolução.

No primeiro caso, o aprendiz irá receber um problema para solucionar e sua resolução será feita passo-a-passo. A resolução passo-a-passo é utilizada aqui para que o tutor possa saber exatamente se o aprendiz está sabendo certos pontos da questão ou, também, saber em qual ponto exato o aprendiz está com dificuldade. A cada interação do passo dado pelo aprendiz, a informação de seu acerto ou erro é usado para modelar seu perfil do currículo que o passo dado pertence.

Outra forma do tutor saber acerca do processo de aprendizagem do aprendiz é através da aplicação de questionários de múltipla escolha com questões sobre o currículo de estudo. O tipo das questões que preenche este questio-

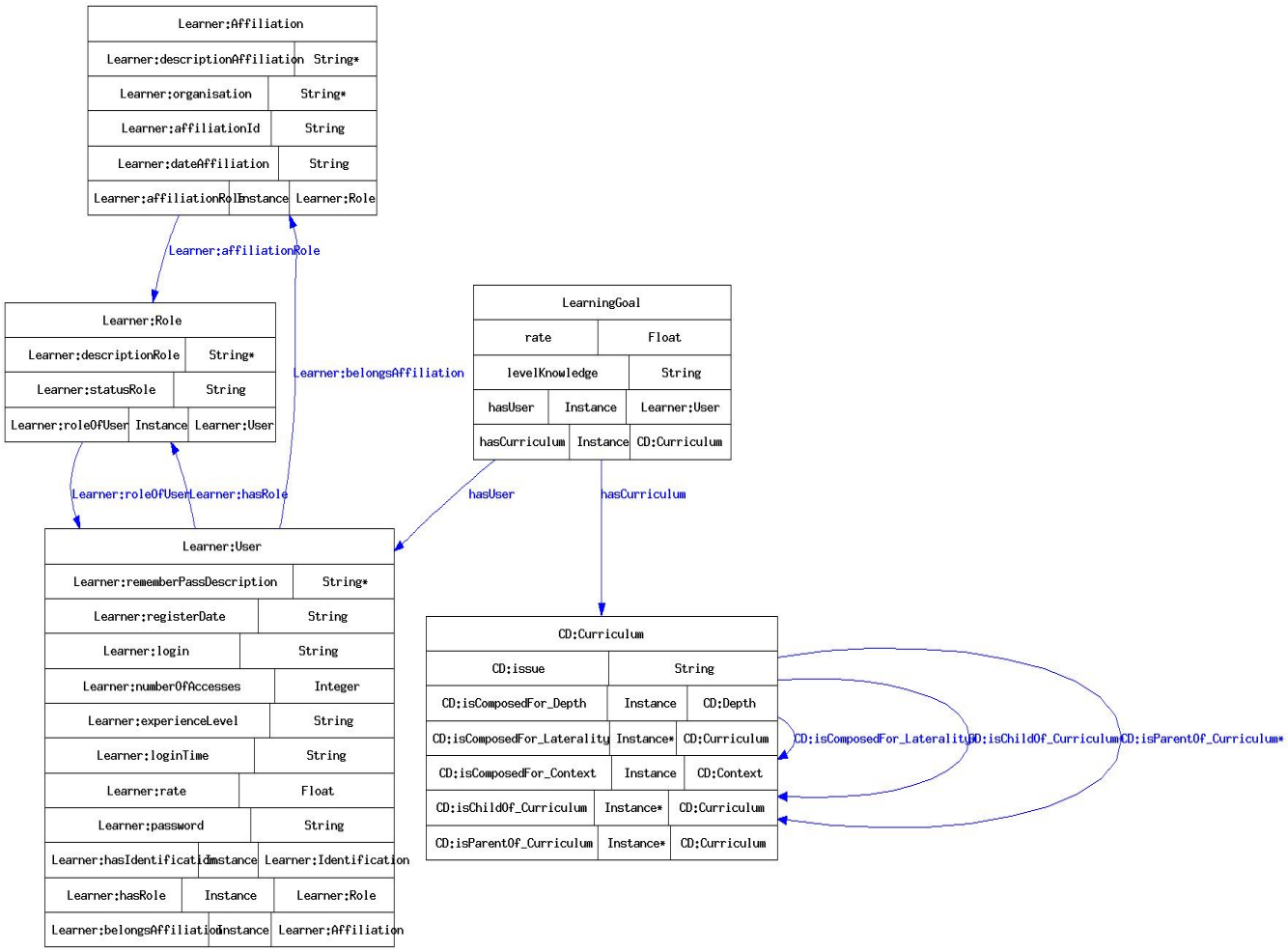

Figura 5.10: Ontologia *Learner.Goal* com suas relações e atributos

nário é exclusivamente sobre o tema do currículo em estudo.

Modelo Aberto do Aprendiz

A modelagem aberta do aprendiz é o nome dado à apresentação ao aprendiz da modegalem feita pelo sistema através da inferência realizada a partir das interações do aprendiz com o sistema. O módulo diagnosticador do tutor tem uma grande responsabilibidade sobre a modelagem do aprendiz, pois é o diagnóstico da resolução de um problema feito pelo aprendiz que irá informar se este foi feito de forma correta ou não, como está ilustrado na sequência 1, 1.1 e 1.1.1 da Figura 5.11.

Quando o aprendiz solicita a observância da sua modelagem, o FraW responde a sua solicitação de forma a apresentar o que o sistema inferiu sobre sua aprendizagem em cada um dos currículos que ele já estudou e também sobre o currículo que ele está estudando, como podemos ver na sequência 2 e 2.1 da mesma figura.

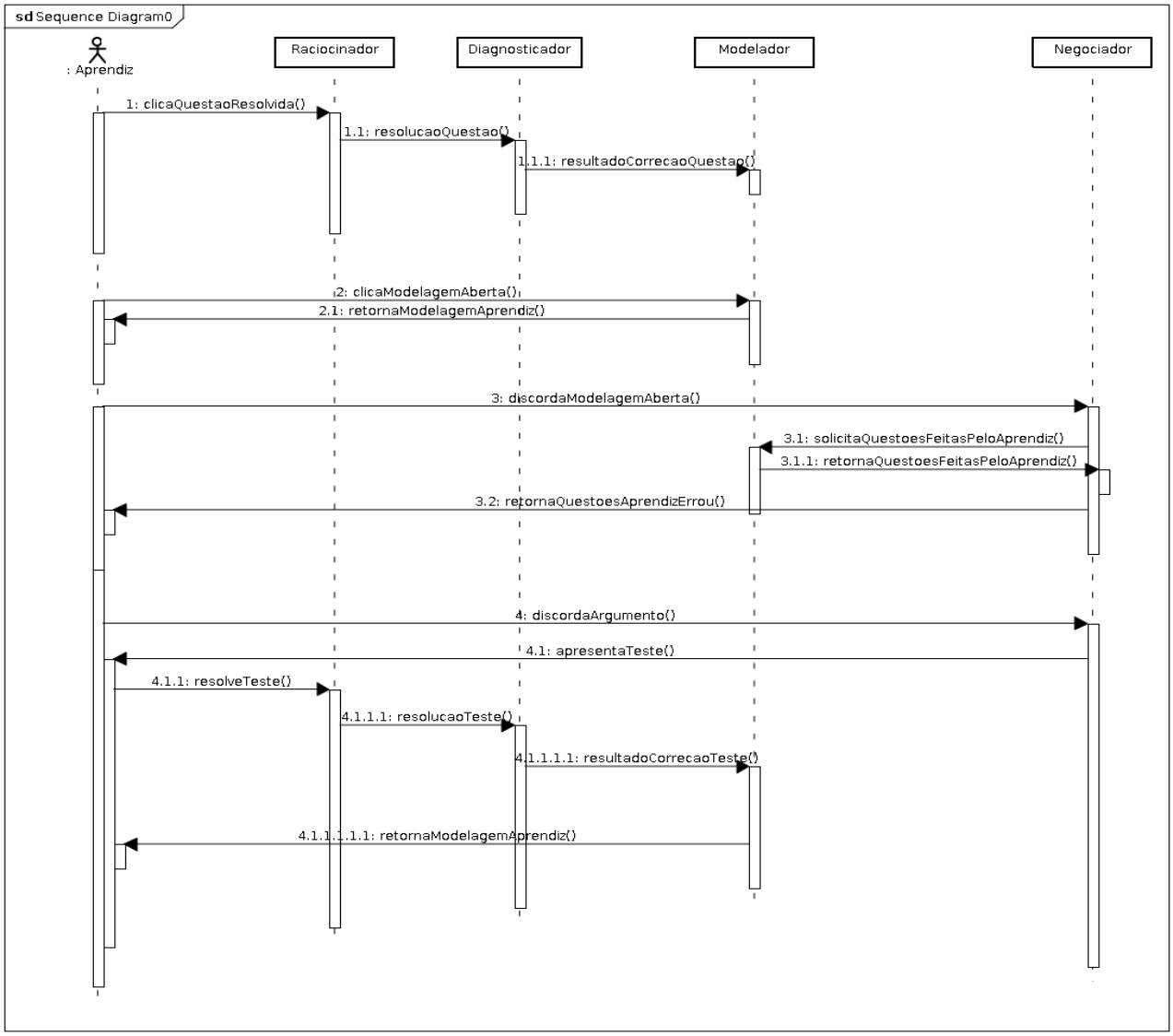

Figura 5.11: Diagrama de sequenciamento do diagnóstico e modelo aberto do aprendiz

Após observar a modelagem feita sobre si, o aprendiz pode concordar com ela, ou discordar (sequência 3 da figura apresentada). A discordância pode ocorrer devido a erro de inferência do próprio sistema, ou porque o aprendiz pode realmente ter uma visão diferente do que é apresentado a ele sobre sua aprendizagem. No caso de discordâncias, o FraW possibilita que o aprendiz externe isso e, a partir de então, começa uma negociação entre o aprendiz e o sistema sobre qual seria a modelagem que melhor transparecesse o verdadeiro estado de aprendizagem do aprendiz.

Para que o sistema possa argumentar sobre o porquê de tal estado de modegem do aprendiz, este busca as questões que o aprendiz fez, e apresenta ao aprendiz as questões que ele solucionou erroneamente (sequência 3.1, 3.1.1 e

3.2 da Figura 5.11). O aprendiz pode ficar satisfeito com o que foi apresentado a ele, e, então, concordar com o que é apresentado sobre ele. Ou o aprendiz pode, mesmo assim, continuar discordando de sua modelagem feita pelo sistema. Neste caso, o FraW apresentará ao aprendiz alguns novos problemas semelhantes aos que ele já haviam sido resolvidos durante o estudo daquele currículo (sequência 4 e 4.1 da sequência na Figura 5.11).

Se o aprendiz responder corretamente as questões propostas, seu estado de aprendizagem pode ser modificado para o que ele sugeriu, caso o estado que ele tenha sugerido tenha sido um nível superior ao que já estava na modelagem dele.

Para operacionalizar tais etapas da apresentação da modelagem do aprendiz e sua negociação, usa-se os agentes tutores, cada um responsável pelo ensino de um currículo, em que cada qual tem seu raciocinador e diagnosticador de acordo com seu currículo. E também faz parte do agente tutor modelar seu aprendiz de acordo com os resultados dos diagnósticos feitos para cada problema resolvido, bem como o negociador.

Através do diagnosticador, o FraW consegue modelar o perfil do aprendiz para, posteriormente, apresentar ao aprendiz, caracterizando a modelagem aberta do aprendiz. O processo de aprendizagem de qualque conhecimento no sistema FraW é suportada pelas ontologias *Learner.Interaction.Cognitive* e *Learner.Goal*, como foram apresentadas. Seu conteúdo pode ser apresentado ao aprendiz para que ele saiba como está o andamento de sua aprendizagem.

Apesar de ser guardada nas ontologias as questões que o aprendiz fez (tanto as que acertou, quanto as que errou), assim como também sua porcentagem de acerto (que é feito através da quantidade de questões que o aprendiz errou e acertou em uma amostra de um banco de problemas de um determinado currículo), o que é apresentado para o aprendiz são os seguintes conceitos, de acordo com a porcentagem de acerto: ótimo (porcentagem de acerto entre 90% - 100%), bom (70% - 89%), razoável (60% - 69%), ruim (40% - 59%), muito ruim (< 39%).

5.2.2 Módulo do Domínio

O Módulo do Domíno do FraW segue o modelo descrito por [25], conhecido por modelo Mathema. A modelagem do domínio de Fração está descrito em ontologias de acordo com a instância da ontologia apresentada na Figura 5.12 e, em forma de árvore, na Figura 5.13, onde se pode ver o domínio de Fração.

De acordo com as figuras apresentadas, temos dentro do Domínio, seus Currículos de: Adição, Subtração, Multiplicação, Divisão e Potenciação. Para

cada currículo, temos uma Unidade Pedagógica relacionada ao assunto do currículo e, para cada Unidade Pedagógica do Currículo, temos pelo menos um recurso (que pode ser vídeo ou material de leitura), uma lista de questões que os aprendizes deverão responder quando estiver estudando o currículo, e uma outra lista de questões que dará suporte ao modelo aberto do aprendiz, quando um aprendiz não concordar com seu *status* de aprendizagem que está descrito lá, e tiver que passar por um teste para provar seu conhecimento sobre aquele determinado currículo.

A descrição dos termos contidos na ontologia de Domínio, termo usado nesta ontologia para designar uma determinada área de estudo, está apresentada abaixo:

- *domainName*: nome do domínio que o tutor irá ensinar, neste caso: Fração;
- *isComposedByContext*: o contexto do domínio que será ensinado, aqui: Simbólico, mas também poderia ser gráfico;
- *isComposedByCurriculum* (a Figura 5.14 apresenta a instância do Currículo de Adição e suas classes subsequentes): os Currículos relacionados ao domínio que serão ensinados aos estudantes, ou seja, as divisões do domínio em áreas menores.
  - *hasPedagogicalUnit* (a Figura 5.15 apresenta a instância da Unidade Pedagógica de Adição e seus atributos/relacionamentos): são as unidades pedagógicas que um Currículo tem, ou seja, uma instanciação de uma parte do Currículo ou dele todo, já com seu conteúdo a ser ensinado.
    - *hasLO*: apresenta os objetos de aprendizagem que a Unidade Pedagógica contém, podendo esses objetos de aprendizagem ser de diferentes tipos, como: conteúdo (*Content*), problema (*Problem*), entre outros.
      - *description*: descrição do objeto de aprendizagem;
      - *URIResource*: a URI (*Uniform Resource Identifier*) em que o objeto de aprendizagem está localizado;
      - *URIType*: o tipo de arquivo do recurso usado.
  - *isComposedByLaterality*: conteúdo lateral ao que está sendo estudado, ou seja, algum assunto que tem envolvimento com o Currículo, mas que não seja o foco dele. No caso de um Currículo de Fração ser Adição, um assunto lateral poderia ser Mínimo Múltiplo Comum.

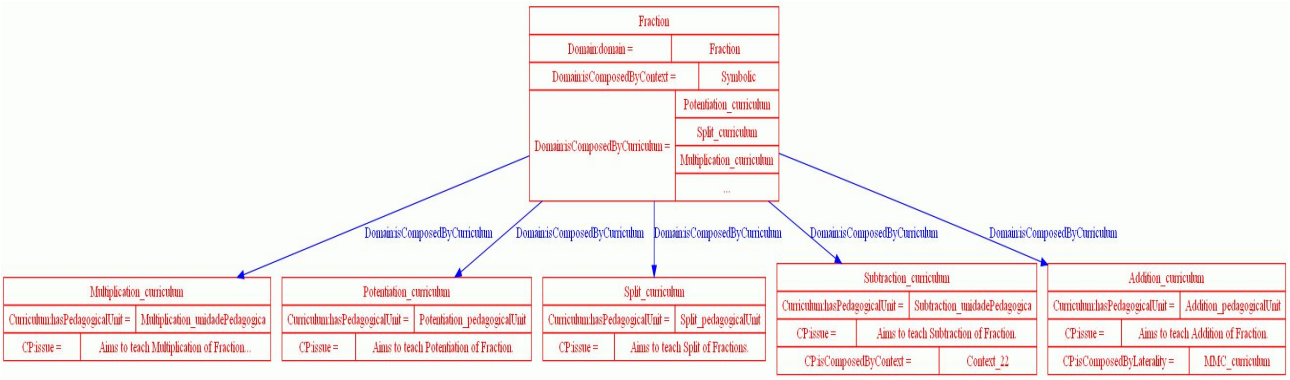

Figura 5.12: Instância da ontologia para o Domínio de Fração

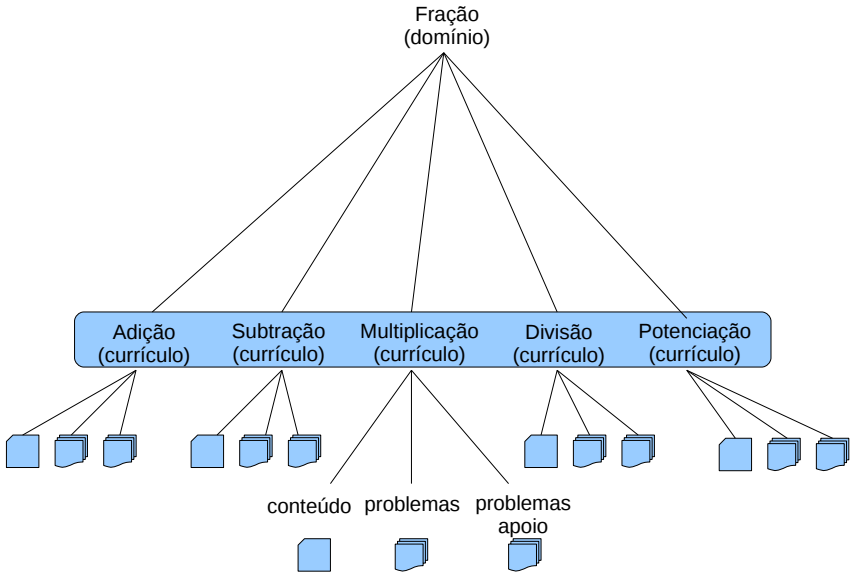

Figura 5.13: Árvore do Modelo do Domínio

Para adicionar um objeto de aprendizagem do tipo Problema, ou Conteúdo, temos que antes, criar uma Unidade para o Problema ou Conteúdo, o qual tem tais características:

- *has_ Problem*: relacionamento para com uma instância de Problema que aquela Unidade de Problema tem;
- *has_ ProblemNext*: o próximo Problema que virá depois daquele;
- *isExplained*: se há uma explicação para aquele Problema;
- *previous_ Unit*: a Unidade de estudo anterior a essa.

E, para cada Problema (*Problem*), devemos escolher o tipo dele dentre as seguintes opções: preenchimento de lacunas em branco (*FillBlanks*), ligação

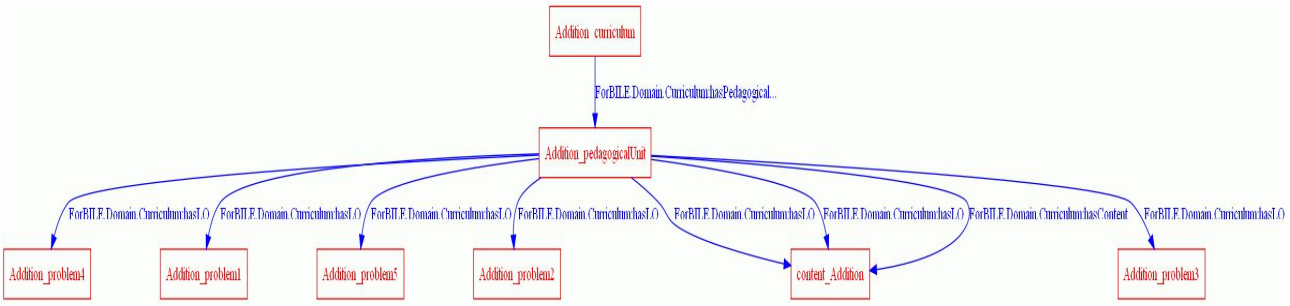

Figura 5.14: Instância da ontologia do Domínio para o Currículo de Adição

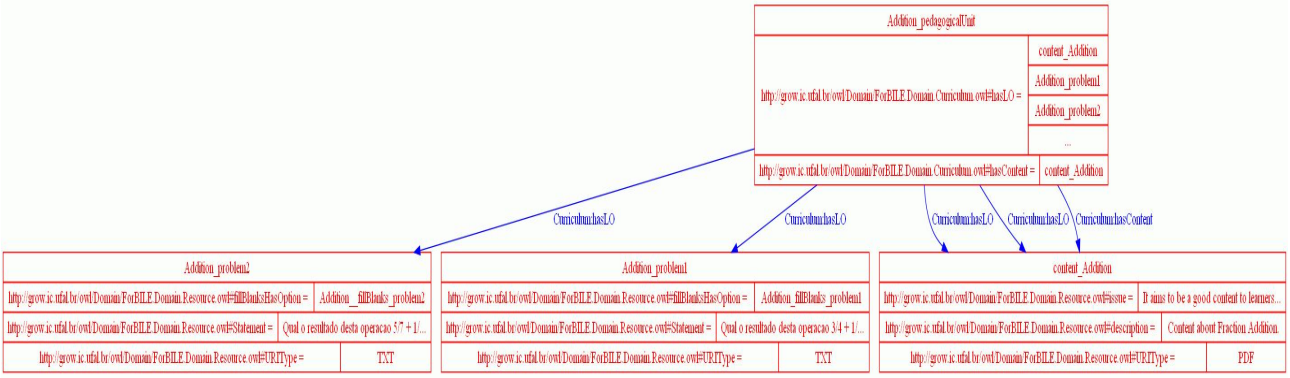

Figura 5.15: Instância da ontologia do Domínio para a Unidade Pedagógica de Adição

de colunas (*MatchColumn*), múltipla escolha (*MultipleChoice*) e verdadeira ou falsa (*TrueOrFalse*). A escolha da opção do tipo de Problema para o FraW foi o de múltipla escolha de operações de fração, além da resolução passo-a-passo que o aprendiz terá que resolver, o qual será diagnósticado seu resultado e apontado sua dificuldade de forma exata através do módulo de raciocínio do agente Tutor irá verificar se ele está correto ou não e quais suas dificuldades. A seguir temos a descrição das características de um Problema:

- *statement*: declaração sobre a questão que será apresentada;
- *URIType*: o tipo de arquivo do recurso usado.

Os atributos para um Conteúdo (*Content*), seja este um arquivo de leitura, um vídeo, um áudio, são os seguintes:

- *description*: descrição do conteúdo usado;
- *URIResource*: a localização do conteúdo usado;
- *URIType*: o tipo de arquivo do recurso usado.

Na Figura 5.16 vemos a instância da classe de Problema (*Problem*) e de Conteúdo (*Content*) para um problema e conteúdo de Adição de Fração:

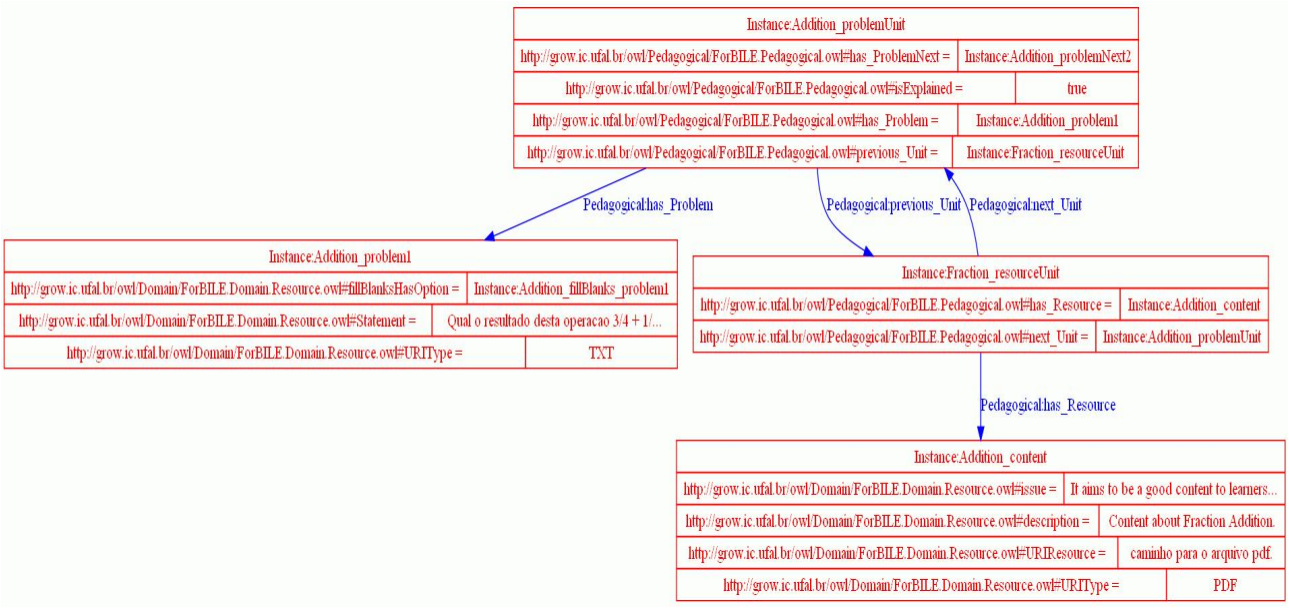

Figura 5.16: Sequenciamento de um Problema

5.2.3 Detalhes de Implementação

As expressões apresentadas para que o aprendiz resolva ou apresentadas pelo aprendiz para que o sistema resolva podem ser de dois tipos: (i) simples ou (ii) complexa. Uma expressão simples é uma expressão em que nela só contém operações do Currículo que ele está estudando, por exemplo: adição: $\frac{3}{2} + \frac{4}{5} + \frac{1}{3}$. Já uma expressão complexa, esta contém operações de vários Currículos, como a seguinte expressão: $\frac{3}{2} - \frac{4}{5} + (\frac{1}{3}/\frac{3}{2}) + \frac{4}{5} * \frac{1}{3}$.

Como apresentado em [22], a resolução de questões que são apresentadas pelo aprendiz segue a sintaxe exposta na BNF [2] abaixo, que é a gramática que gera as expressões válidas.

<expressao> ::= <termo><op-complexa> | <termo><mais-op-simples>

<termo> ::= <numero> | (<expressao>)

<numero> ::= <natural> | (-<natural>)

<natural> ::= <algarismo><mais-alg>

<mais-alg> ::= λ | <natural>

<algarismo> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<mais-op-simples> ::= λ | <op-comlexa><mais-op-simples> | <op-simples><termo><mais-op-simples>

<op-simples> ::= + | - | / | *

<op-complexa> ::= ^<numero> | :<natural>

A base do conhecimento é o conjunto das sentenças lógicas, aqui, estas

[2] Backus-Naur Form

sentenças são, essencialmente, sentenças da lógica proposicional. As sentenças lógicas da base do conhecimento têm que seguir um padrão, o qual caracteriza uma regra formada por um conjunto de símbolos proposicionais (premissas) que implicam em um outro símbolo proposicional (conclusão). Duas regras da adição de fração seguem abaixo:

Código 5.1: Exemplos de regras da adição de fração

```
<regra>
        <premissa>
                <valor>true</valor>
                <descricao>OperacaoDeAdicao</descricao>
        </premissa>
        <premissa>
                <valor>true</valor>
                <descricao>DenominadoresIguais</descricao>
        </premissa>
        <conclusao>
                <valor>true</valor>
                <descricao>(n1+n2)/d1</descricao>
        </conclusao>
</regra>

<regra>
        <premissa>
                <valor>true</valor>
                <descricao>OperacaoDeAdicao</descricao>
        </premissa>
        <premissa>
                <valor>false</valor>
                <descricao>DenominadoresIguais</descricao>
        </premissa>
        <conclusao>
                <valor>true</valor>
                <descricao>((mmc/d1)*n1+(mmc/d2)*n2)/mmc</descricao>
        </conclusao>
</regra>
```

Escritas no formato XML (*Extensible Markup Language*) [16], seguem o padrão das premissas implicando em uma conclusão. A primeira regra, por exemplo, pode ser lida da seguinte maneira: se é uma adição de fração e os denominadores são iguais, então a adição será transformada para uma ex-

pressão do tipo “(n1+n2)/d1”, ou seja, o denominador da primeira fração é conservado e os numeradores são somados.

5.2.4 Modelo Pedagógico

O FraW apresenta para o seu aprendiz uma sequência de conteúdo e questões, como podemos ver na Figura 5.17, sempre temos um conteúdo sobre a Unidade Pedagógica primeiro, para depois vir as questões sobre tal unidade. A sequência apresentada na figura é a seguinte: Adição, Subtração, Multiplicação, Divisão e Potenciação de Fração, começando sempre por seu conteúdo seguido de questões.

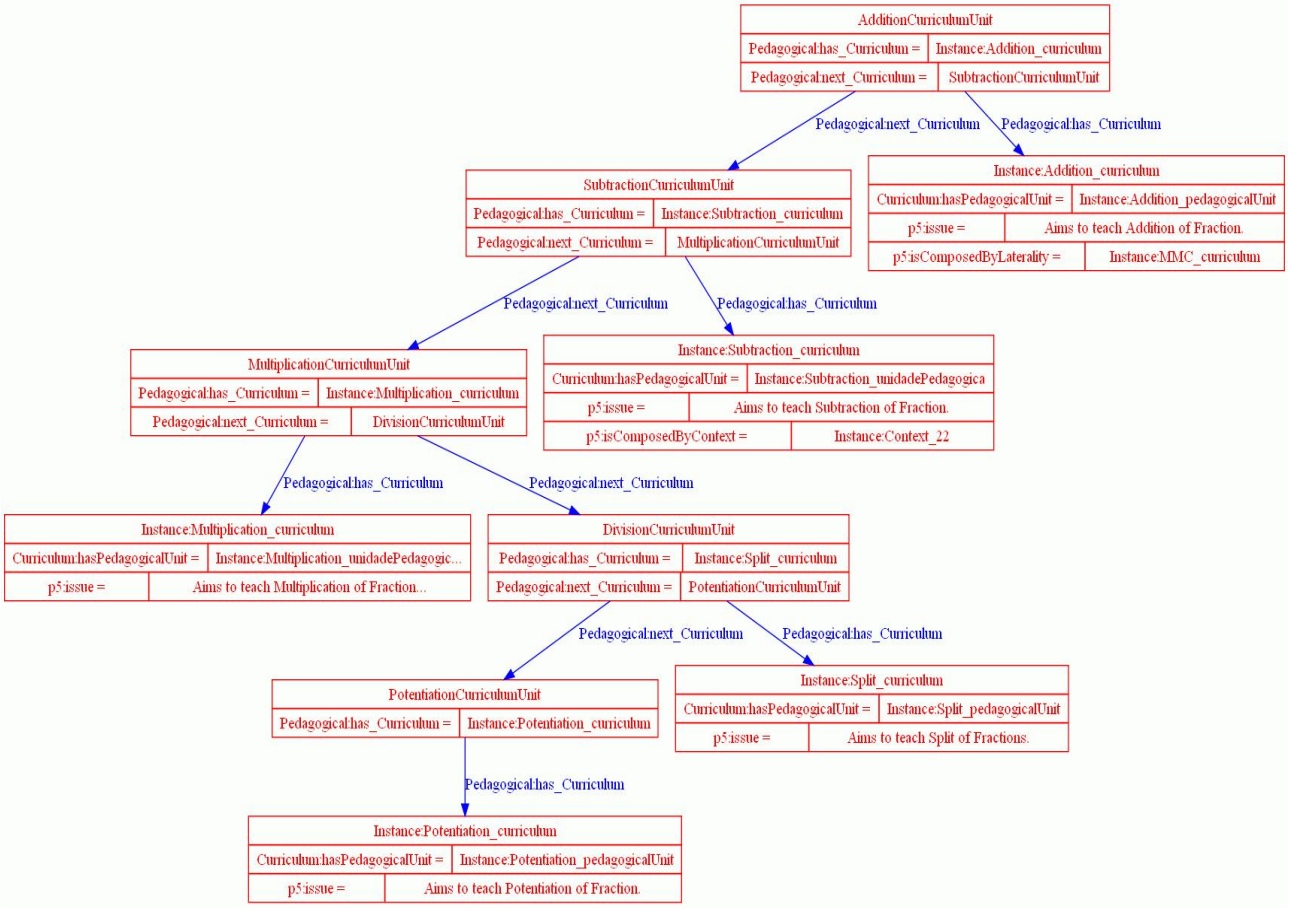

Figura 5.17: Sequenciamento de recursos apresentados ao aprendiz do FraW

Será a partir da sequência de questões (Problemas) que o aprendiz será avaliado. Para isso, foi criado uma classificação de questões dentro de uma Unidade Pedagógica, portanto, temos, dentro de uma certa Unidade, questões fáceis, mediadas e difíceis.

Além disso, também é armazenado nas ontologias outras sequências de Problemas que serão apresentados para o aprendiz apenas no momento de

Negociação acerca de sua modelagem do aprendiz, caso ele não concorde com a inferência do FraW sobre seu status de aprendizagem.

O FraW, como um STI, ele pode resolver algumas questões propostas pelo aprendiz, assim como também propor questões para o aprendiz, para que este passo-a-passo responda e o tutor possa, em cada passo, informar se sua resposta está correta ou não, e, caso não esteja, dar dicas sobre como resolver. Tais dicas podem ser oriundas das "dicas de erro comum", que são dicas baseadas no erros que a grande maioria dos aprendizes cometem.

O FraW também guarda, além das dicas e dicas de erros comum, um outro recurso de ajuda do módulo Pedagógico, como a recomendação de um par-complementar (seja esse par um agente virtual ou humano, focaremos o agente humano), para que este possa dar suporte ao aprendiz em sua dificuldade. Sobre tal recomendação é falado a seguir, na Seção 5.2.4.

Serviço de Recomendação de um Par-Complementar

Hoje, com o desenvolver das tecnologias, da *Web* e dos ambientes educacionais, é notável que a aprendizagem não se restringe aos muros das instituições educativas. Ainda mais: que relações não são criadas de forma direta, mas através de, entre outras coisas, ferramentas (instrumentos), passando a ser mediada por esse elemento [3]. Além disso, a educação deve sempre ser repensada, de forma a animar, facilitar e promover a aprendizagem, integrando seus diferentes aprendizes [19].

Portanto, no contexto de um STI, o aprendiz, apesar de ter um tutor virtual para lhe dar assistência sobre o assunto que está sendo estudado, tirando dúvidas e lhe passando novos conhecimentos, ainda assim pode ter um rendimento não satisfatório sobre aquele assunto. O rendimento do aprendiz pode ser observado pelo sistema através da observação da modelagem feita sobre aquele aprendiz, o qual construiu seu Modelo do Aprendiz.

Assim, tal serviço de recomendação de um par-complementar observa os modelos dos aprendizes, focando o *status* da aprendizagem do aluno em um determinado tópico do assunto, para que, encontrando um aprendiz que não esteja tendo um bom rendimento neste tópico [34], esse serviço possa lhe ajudar, assim como será apresentado a seguir.

Caso o aprendiz, mesmo interagindo com o tutor virtual, não esteja apresentando um bom entendimento sobre um certo Currículo de aprendizagem (sendo observado isso na resolução de um problema correspondente ao assunto do Currículo), o sistema recomendará a ele um agente humano. O cenário da necessidade de se ter um outro aluno para ajudar ao aprendiz com

deficiência na aprendizagem do assunto em questão é o que se segue:

- O aprendiz está resolvendo uma questão de um determinado Currículo (Figura 5.18);
- Caso o aprendiz, mesmo com as dicas dadas pelo tutor do Currículo em relação à questão, não consiga resolver a problema corretamente, o Módulo Pedagógico do agente Tutor o qual o Currículo pertence observa se a dúvida é sobre algo do cunho do próprio Currículo, ou não. Caso não seja, o aprendiz será colocado em contato com outro agente Tutor virtual (Figura 5.19), este tutor pode estar no mesmo ambiente de interação, ou em outro ambiente, o qual foi encontrado através da camada de Interoperabilidade do FraW, e este será um agente especializado no assunto correlato, para que este possa guiar o aluno na aprendizagem ou revisão do assunto correlato, para que ele, depois dessa interação, possa continuar a aprendizagem do tópico de estudo inicial;
- Caso o erro seja em relação a algo do próprio Currículo, o tutor ativa o serviço de recomendação, passando como referência a modelagem do aprendiz a ser ajudado. O algoritmo deste serviço está implementado como se segue:

  1 O serviço recebe o aprendiz que necessita de ajuda e o Currículo que deve ser ajudado;

  2 É procurado na ontologia da modelagem dos aprendizes pelos aprendizes que já estudaram o Currículo dado como entrada e que obteve um bom rendimento nele (acima de 80% de rendimento) [41];

  3 Caso nenhum aprendiz seja encontrado sob esta condição, o sistema deve retornar algum outro recurso (material de leitura, vídeo);

  4 Caso sejam encontrados um ou mais aprendizes, é observado quais estão *on-line* e enviado um pedido de ajuda a este. Caso todos se recusem a ajudar, acontece o que está exposto no item [3].

  5 Caso uma pessoa se disponibilize a ajudar, acontecerá o caso [7];

  6 Caso mais de uma pessoa se disponibilize a ajudar, é apresentado ao aprendiz a lista de pessoas disponíveis em ajudá-lo, e este poderá escolher de acordo com seus próprios pesos (qual estudante ele gosta mais, qual o mais geograficamente próximo a ele, entre outros) e, então, o item [7] acontecerá;

  7 O par de aprendizes, o que necessita de ajuda e o que irá ajudar, serão colocados em contato.

- Assim, como resultado da recomendação o serviço retornará uma pessoa que poderá ajudar o estudante que está com dificuldade (Figura 5.20);
- Para que a ajuda se torne mais fácil, o FraW abre uma janela de *chat* entre as partes envolvidas;
- Enquanto os aprendizes estiverem interagindo através da ferramenta de *chat*, haverá um serviço que verificará se de fato os aprendizes estão falando sobre operações de Frações. Tal serviço faz a verificação da conversa dos aprendizes através de seu arquivo de log. Algumas palavras-chave sobre o assunto em questão são guardados na ontologia de Currículo, para que a verificação do diálogo seja observada de acordo com o que seria comum conversar sobre um certo Currículo.

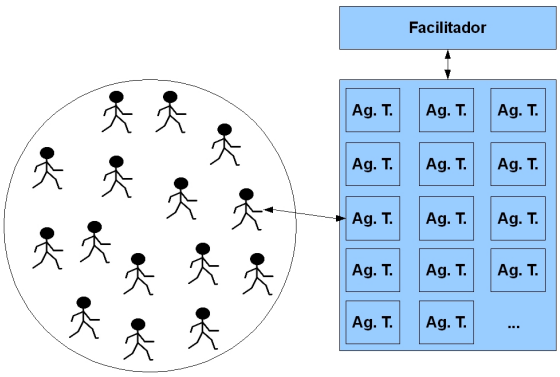

Figura 5.18: Aprendiz interagindo com um agente Tutor

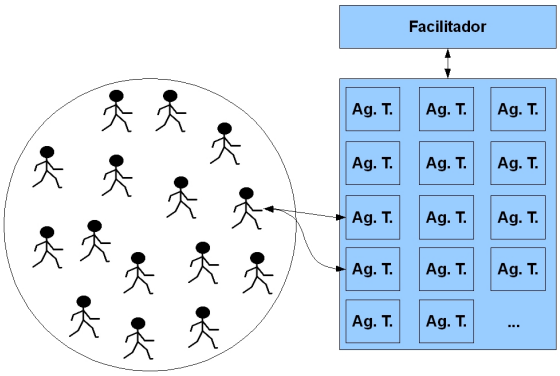

Figura 5.19: Aprendiz em comunicação com um outro agente Tutor para aprender um assunto correlato a o que ele já estava estudando

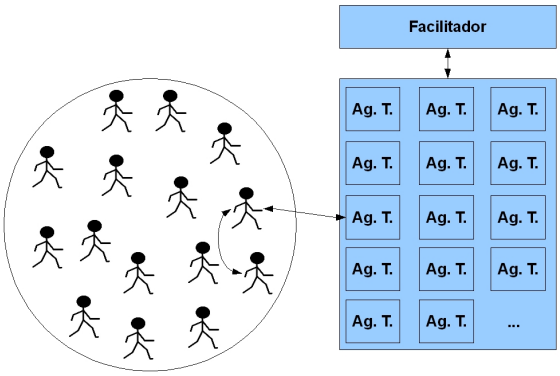

Figura 5.20: Aprendiz interagindo com um outro aprendiz que poderá ajudá-lo com suas dúvidas

A busca por um par-complementar é feita através de um serviço *Web* semântico em que é apresentado como entrada a URI do aprendiz que irá receber a ajuda, e, como saída, o aprendiz que está *on-line* e irá ajudar. A busca por esse outro aprendiz que irá ajudar o aprendiz com deficiência em um determinado currículo será feita entre os outros estudantes que estão estudando fração que já tenham estudado este currículo e que tenham conseguido uma boa conceituação neste.

O processo de criação de laços entre pessoas distribuída em diferentes espaços geográfico e que não se vêem é um desafio grande. A parceria entre um aprendiz que necessita de ajuda e outro que vem a ajudá-lo facilita a criação de tais laços, devido a confiança que um deposita no outro durante o processo de troca de conhecimento [30].

Serviço de Mineração de Texto

Quando um aprendiz é colocado em contato com outro, como já foi apresentado na Subseção 5.2.4 (Serviço de Recomendação de um Par-Complementar), é aberta uma janela de *chat* para que um aprendiz possa aprender com o outro. Como, porém, os aprendizes podem interagir nesta ferramenta de forma a não dar ênfase ao assunto do estudo, há uma outra ferramenta, a de mineração de texto, que busca saber se a interação entre os aprendizes foi efetiva, ou seja, se realmente conversaram sobre o assunto do currículo em questão.

A solução usada para minerar o *chat* entre os pares é feita por casamento de palavras-chave, a qual foi adotada por sua maior facilidade de implementação e é apresentada a seguir:

- 1 É recebido o *log* da conversa entre os pares;

- 2 Procura-se as palavras-chave correspondentes ao Currículo discutido em questão;
- 3 É feito o *match* das palavras-chave para cada linha do *log*;
- 4 É armazenado a quantidade de linhas em qual houve casamento de palavras;
- 5 É calculada a porcentagem de linhas em que houve casamento de palavras;
- 6 É retornada a porcentagem calculada.

Assim, esta ferramenta de mineração de texto recebe como entrada a URI do currículo em questão, bem como também o *log* da conversa entre os estudantes. Na ontologia do Currículo há o campo "palavras-chave" para o uso deste serviço, este campo armazena as palavras que devem ser procuradas no texto para se saber se os aprendizes estão ou não conversando sobre o tema do Currículo, por isso que este serviço recebe o Currículo e o *log* da conversa. Um exemplo das palavras-chave que conteriam no Currículo de Adição de Fração poderia ser: adição, soma, +, mmc, m.m.c., mínimo múltiplo comum.

Este serviço de mineração funciona da seguinte forma: em cada linha do *log* de entrada é procurado se há pelo menos uma palavra que esteja em seu conjunto de palavras-chave. O retorno deste serviço será a porcentagem de linhas que tenham alguma dessas palavras-chave em relação ao total de linhas do *log*. Se a porcentagem retornada for maior ou igual a 40%, o sistema considera que o aprendiz ajudado está em condições de continuar resolvendo as questões do Currículo discutido.

5.2.5 Modelagem GAIA dos Agentes Tutores

Mais afrente será apresentado a necessidade de se ter a modelagem em GAIA [65] dos Agentes Tutores, portanto, apresenta-se aqui tal modelagem:

Ambiente:

- Ler
  - Regras[i], i=1, númeroTotalDeRegras (regras para realizar a inferência dos agentes tutores)

  - ObjetosDeAprendizagem[i], i=1, númeroTotalDeObjetosDeAprendizagem (objetos de aprendizagem, recursos que dispõem ao aluno conteúdo sobre um determinado domínio, bem como questões)

- Modificar

  - Ontologias[i], i=1, númeroTotalDeOntologias (ontologias que armazenam e expressam informações acerca dos estudantes do sistema, do modelo pedagógico de ensino e do domínio que está sendo dado pelo sistema)

Pela similaridade entre os agentes tutores, será apresentado apenas os Papéis dos agentes tutores de Adição e de Multiplicação.

Papéis:

- Papel: **TutorAdição**

  - **Descrição**: Este papel tem como objetivo auxiliar o aluno no aprendizado da operação de adição envolvendo Frações.
  - **Protocolos e atividades**: recebeRespostaAprendiz, realizaDiagnóstico, devolveInstrução, solicitamConteúdo, obtémObjetoAprendizagem, retornaObjetoAprendizagem, recebeProblema, resolveProblema, devolveSoluçãoDoProblema, solicitaMMC, recebeMMC, recebeSolicitaçãoParComplementar, pesquisaParComplementar, recomendaParComplementar.

    Os Protocolos e Atividades desse agente foram substituídos por serviços semânticos que realizam tais funções, portanto, só restou tais protocolos: *chamarServiçoSemântico* e *receberServiçoSemântico*
  - **Permissões**:
    - Ler
      - Regras (somente as regras relacionadas a Adição de Fração)
      - ObjetosDeAprendizagem (somente os objetos de aprendizagem voltados ao estudo/ensino de Adição de Fração)
    - Modificar
      - Ontologias (modificar as ontologias para atualizar o progesso do aluno em relação ao assunto de Adição de Frações)
  - **Responsabilidades**:

    * Liveness:
        · (recebeProblema.resolveProblema.[devolveSoluçãoDoProblema])^ω (**substituída pelo serviço semântico: *serviçoResolverProblemaAdição***) | |
        · (recebeRespostaAprendiz.realizaDiagnóstico.[devolveInstrução])^ω (**substituída pelo serviço semântico: *serviçoReceberRespostaAlunoAdição***) | |
        · (solicitamConteúdo.[obtemObjetoAprendizagem .retornaObjetoAprendizagem])^ω (**substituída pelo serviço semântico: *serviçoRetornarObjetoAdição***) | |
        · (solicitaMMC.[recebeMMC])^ω (**substituída pelo serviço semântico: *serviçoSolicitarMMCAdição***) | |
        · (recebeSolicitaçãoParComplementar.pesquisaParComplementar .[recomendaParComplementar])^ω (**substituída pelo serviço semântico: *serviçoRecomendaçãoParComplementar***)
        Cada Responsabilidade *Liveness* foi substituída por um serviço semântico.

- Papel: **TutorMultiplicação**
    - **Descrição**: Este papel tem como objetivo auxiliar o aluno no aprendizado da operação de multiplicação envolvendo Frações.
    - **Protocolos e atividades**: recebeRespostaAprendiz, realizaDiagnóstico, devolveInstrução, solicitamConteúdo, obtémObjetoAprendizagem, retornaObjetoAprendizagem, recebeProblema, resolveProblema, devolveSoluçãoDoProblema, recebeSolicitaçãoParComplementar, pesquisaParComplementar, recomendaParComplementar.
      Os Protocolos e Atividades desse agente foram substituídos por serviços semânticos que realizam tais funções, portanto, só restou tais protocolos: *chamarServiçoSemântico* e *receberServiçoSemântico*
    - **Permissões**:
        * Ler
            · Regras (somente as regras relacionadas a Multiplicação de Fração)
            · ObjetosDeAprendizagem (somente os objetos de aprendizagem voltados ao estudo/ensino de Multiplicação de Fração)
        * Modificar

· Ontologias (modificar as ontologias para atualizar o progesso do aluno em relação ao assunto de Multiplicação de Frações)

– **Responsabilidades**:

* Liveness:

· (recebeProblema.resolveProblema.[devolveSoluçãoDoProblema])^ω (**substituída pelo serviço semântico: *serviçoResolverProblemaMultiplicação***) | |

· (recebeRespostaAprendiz.realizaDiagnóstico.[devolveInstrução])^ω (**substituída pelo serviço semântico: *serviçoReceberRespostaAlunoMultiplicação***) | |

· (solicitamConteúdo.[obtémObjetoAprendizagem .retornaObjetoAprendizagem])^ω (**substituída pelo serviço semântico: *serviçoRetornarObjetoMultiplicação***)

· (recebeSolicitaçãoParComplementar.pesquisaParComplementar .[recomendaParComplementar])^ω (**substituída pelo serviço semântico: *serviçoRecomendaçãoParComplementar***)
Cada Responsabilidade *Liveness* foi substituída por um serviço semântico.

* Safety:

· num_de_receberFrações = num_de_devolverSoma

· num_de_solicitarMMC = num_de_receberMMC

Pela similaridade entre os agentes Tutores e seus serviços semânticos, serão apresentados, a seguir, apenas os serviços do agente Tutor de Adição:

- ***serviçoResolverProblemaAdição***
  - Entrada: recebeProblemaAdição (*String*)
  - Saída: devolveSoluçãoProblemaAdição (*String*)
- ***serviçoReceberRespostaAlunoAdição***
  - Entrada: recebeRespostaAprendizAdição (*String*)
  - Saída: devolveInstruçãoAdição(*String*)
- ***serviçoRetornarObjetoAdição***
  - Entrada: solicitamConteúdoAdição (*String*)
  - Saída: retornaObjetoAprendizagemAdição (*String*)

- ***serviçoSolicitarMMCAdição***
  - Entrada: solicitaMMCAdição (*String*)
  - Saída: recebeMMCAdição (*String*)

A modelagem GAIA do ambiente FraW e do agente Tutor de Adição está especificado na Figura 5.21, onde podemos ver a descrição do ambiente FraW, suas permissões, já citadas aqui, e seus recursos, também já citados aqui. Fração é a sub-organização do FraW, com seu nome, descrição, seus agentes (Adição, Subtração, Multiplicação, Divisão e Potenciação) e suas permissões, como modificar ontologia e ler objetos de aprendizagem. Na figura é apresentada também o agente de Tutor de Adição com seu nome, descrição, quais serviços ele provê, suas responsabilidades e permissões.

5.3 Agente Mediador

O Agente Mediador é o responsável pelo recebimento das interações do aprendiz com o ambiente e, a partir disso, repassa aos agentes Tutores as devidas informações.

É o agente Mediador que receberá a questão do aluno a ser resolvida e passará para cada Tutor uma parte da questão para que ele resolva. A medida que os Tutores vão respondendo a parte da questão determinada a ele, ele o envia ao agente Mediador, o qual vai criando a solução para a questão inserida pelo aprendiz, além da explicação da resolução da questão. Lembrando que tais questões apresentadas pelo aprendiz podem ser tanto expressões simples, quanto complexas, portanto, é o Mediador que separará as operações a serem feitas e sua ordem de resolução, para, então, convocar os Tutores para resolver tais operações.

O agente Mediador também acompanha o aprendiz durante seu processo de aprendizagem com algum agente Tutor, repassando as questões apresentadas pelo Tutor ao aprendiz, ou as informações de forma geral, e, também, recebendo as informações do aprendiz, para passar para o agente Tutor preencher as informações sobre o aprendiz.

A modelagem aberta do aprendiz é apresentada pelo agente Mediador, onde ele resgata das ontologias as informações sobre o aprendiz que cada agente Tutor depositou sobre seu determinado Currículo, para, então, passar para o estudante.

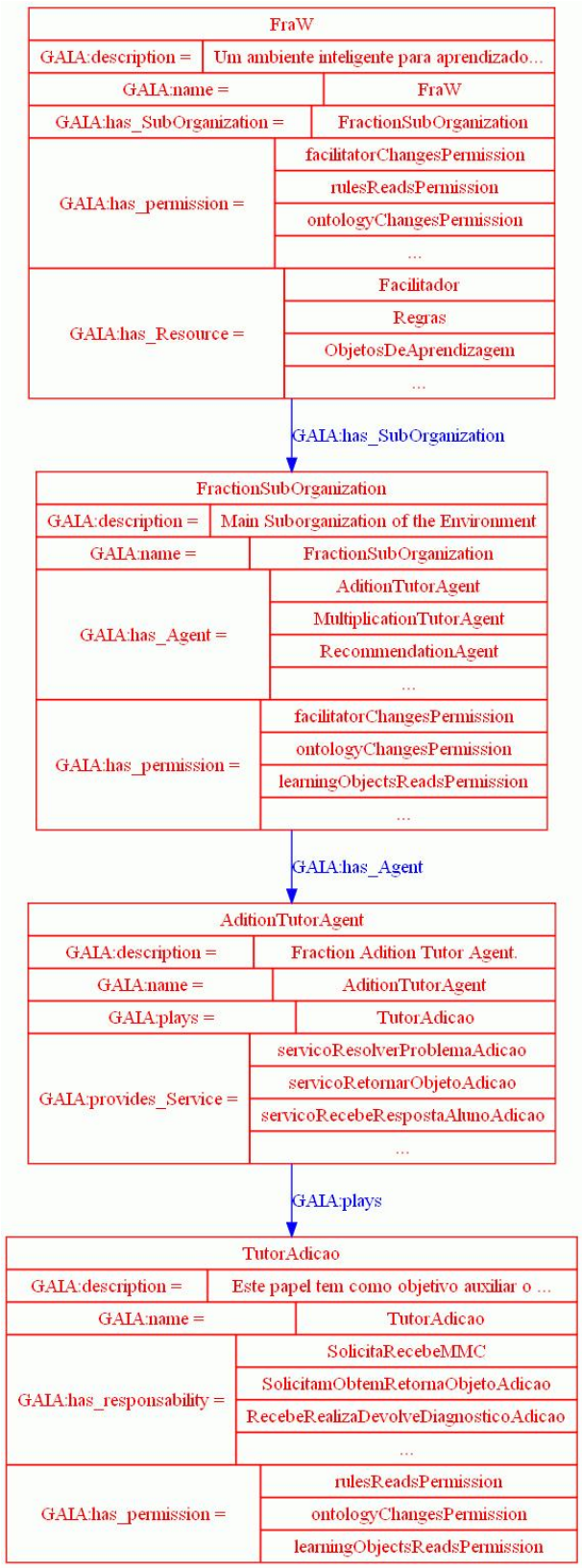

Figura 5.21: Modelagem GAIA do ambiente FraW e o agente Tutor de Adição

5.4 Interface

Grande parte do conforto do aprendiz no ambiente se dá a forma de inserção e recebimento da informação. A interface Web do FraW faz uso do trabalho [24]

que proporciona um melhor comodismo ao estudante de Fração que irá fazer uso das expressões matemáticas com a mesma notação vista em sala de aula.

O trabalho de [24], já citado aqui, teve a preocupação em responder as duas perguntas seguintes, em relação a interface gráfica do FraW: "como os usuários de um sistema *Web* fornecerão suas expressões matemáticas como entrada?"e "eles precisarão escrevê-las usando a sintaxe de uma linguagem de marcação?". Para que tais questões fossem resolvidas foi utilizado um editor visual de expressões matemáticas, conhedido por MathEdit, que pode ser incorporado a qualquer página *Web* e foi desenvoldido por [58, 57].

A arquitetura da API MathEdit é apresentada na Figura 5.22. As ações do aprendiz, como cliques de *mouse* e digitação são tratados como comandos que invocam funções JavaScript. Cada função basicamente ajusta a árvore DOM (*Document Object Model*) do código MathML que representa a expressão editada. MathML é uma linguagem destinada à descrição da notação matemática. Ela é uma extensão do XML (*eXtensible Markup Language*), onde se tem o MathML *Content*, que temos a semântica da expressão, e o MathML *Presentation*, o qual descreve a aparência da expressão matemática.

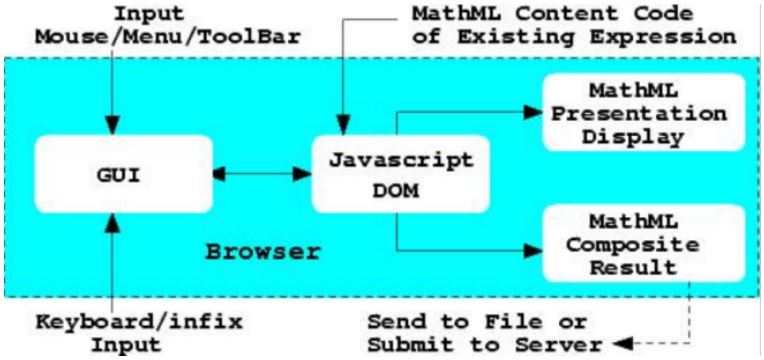

Figura 5.22: Arquitetura MathEdit [24]

5.5 Geração do FraW no Massayo

O FraW foi construído através das tecnologias de agentes, ontologias e serviços semânticos. Para a construção do ambiente FraW, foi utilizado também o Massayo, sistema educacional mediado por computador dotados de adaptabilidade e baseados na Web Semântica [47].

Para se utilizar o Massayo, foi preciso modelar os agentes na metodologia GAIA e passar essa modelagem para ontologias, para, então, a partir de tais ontologias, o Massayo poder transformar a modelagem nos agentes e configuração Jade [7]. Esses passos foram seguidos e estão descritos a seguir:

1. Mapeamento Domínio-GAIA e suas regras: o domínio da aplicação deve ser especificado em ontologia, a qual gerará, de cada currículo do domínio, um agente tutor. Após a especificação do domínio, as regras utilizadas no domínio dos agentes Tutores devem ser executadas, as quais gerará os recursos dos papéis dos Tutores, estes apresentados na Figura 5.23

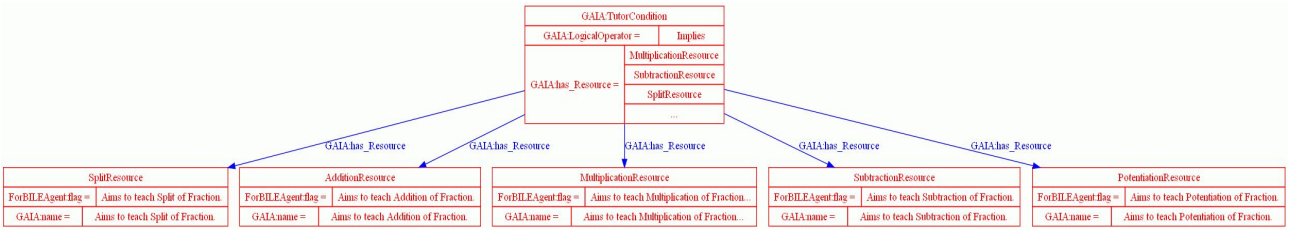

Figura 5.23: Alguns recursos gerados a partir das regras de mapeamento [47]

2. Ontologia de especificação GAIA: a ontologia de GAIA, apresentada na Figura 5.24, nos permite ajustar em uma ontologia a modelagem GAIA do sistema multiagente. Na ontologia de GAIA há um ambiente (*Environment*), o qual especifica o ambiente em que os agentes irão interagir, descritos através de sua descrição, premissões (*Permissions*) e sub-organizações (*Sub-organization*), que podem ser de agentes tutores e de agentes que não são tutores (Agente de Evolução, Agente Controlador e Agente Mediador), tais agentes foram descritos na Seção 2.6 e apresentados na Figura 2.7. Em cada sub-organização temos seus agentes (*Agent*) e permissões (*Permission*). Cada agente tem seu papel (*Role*) e cada papel tem suas responsabilidades (*Responsability*) e permissões também. Cada responsabilidade pode ser do tipo *LivenessResponsability*, que será uma atividade, e do tipo *SafetyResponsability*, o qual precisa de pré e pós condições para a execução de um serviço.

   Na Seção 5.2.5 pode ser vista a modelagem GAIA dos agentes Tutores do FraW; após a especificação desses agentes na ontologia de GAIA, suas configurações em Jade podem ser gerados através das regras contidas nesta ontologia. Na Figura 5.25 pode ser visto parte da modelagem do agente Tutor de Adição e seus serviços, sendo apresentado também as entradas (*input*) e saídas (*output*) dos serviços deste Tutor.

3. Regras de Mapeamento GAIA-JADE/ForBILEAgent: com as instâncias da ontologia GAIA criada, mapeou-se as regras de GAIA para um modelo de construção e execução de agentes. Assim, as instâncias de GAIA foram mapeadas para a ontologia FORBILEAgent, como podemos ver na Figura 5.26.

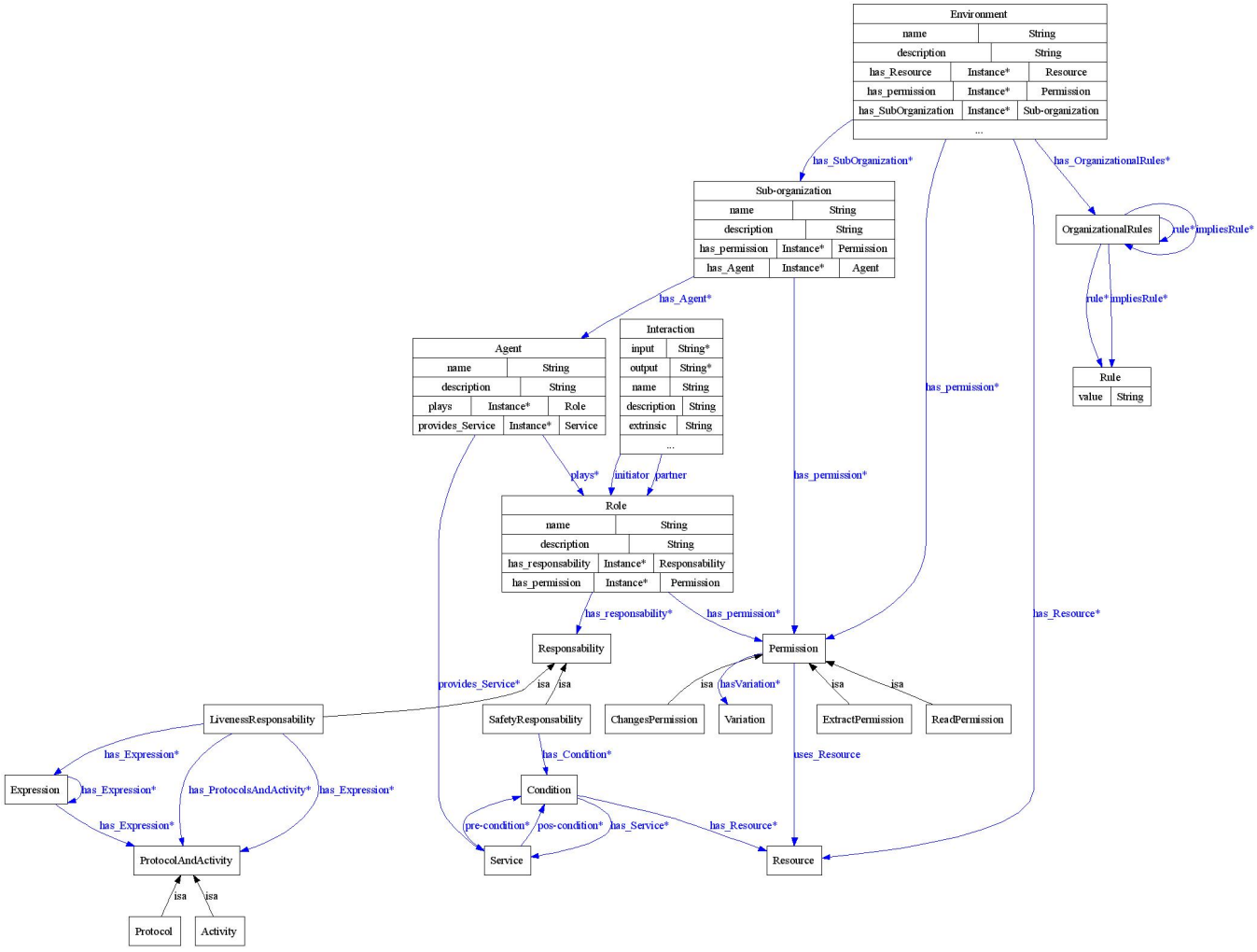

Figura 5.24: Ontologia para mapeamento de um sistema multiagente em GAIA do MASSAYO [47]

4. Ontologia de ForBILEAgent: esta ontologia importa tanto a ontologia de GAIA, aqui já apresentada, quanto a de JADE (Figura 5.27), a qual mantém informações sobre um agente JADE (*JADEAgent*), como o identificador do agente (AID), quais seus comportamentos (*Behaviour*), mensagens (*ACLMessage*), o container (*Container*) que ele está inserido, e sua plataforma (*AgentPlatform*), podendo tal plataforma estar em uma única máquina ou mais de uma.

   Após a execução das regras de mapeamento, a ontologia ForBILEAgent, apresentada em 5.28, é gerada, sendo necessária a configuração de algumas propriedades dos agentes nesta ontologia, como a união dos recursos oriundos da ontologia de Domínio com a de JADE.

   A ontologia ForBILEAgent para o agente Tutor de Adição é apresentado na Figura 5.29.

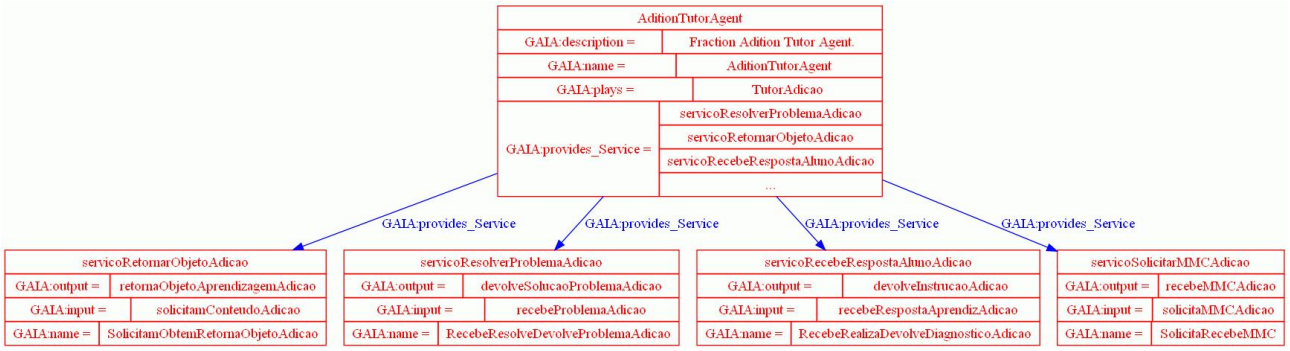

Figura 5.25: Ontologia do mapeamento GAIA do agente Tutor de Adição e seus serviços

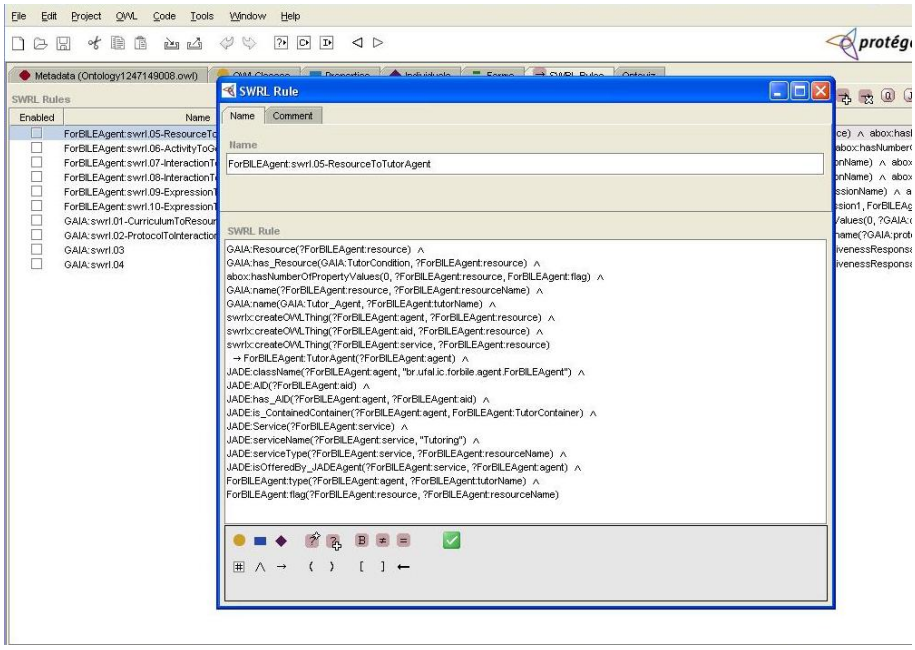

Figura 5.26: Exemplo de regra de mapeamento GAIA-JADE/ForBILEAgent no FraW

Com a ontologia contendo as configurações em Jade do sistema multiagente, foram implementados os códigos em Java.

É bom lembrar que a tela de *login* e de registro no ambiente também foi fornecido pelo Massayo/ForBILE. Quanto aos serviços semânticos, eles não são gerados pelo Massayo/ForBILE. Como foi visto nesta seção, os agentes Tutores e algumas telas foram geradas pelo Massayo/ForBILE.

Como uma breve análise do uso do *framework* Massayo, para criar uma instância dele foi preciso, como foi apresentado, modelar em GAIA os agentes tutores, bem como adaptar as ontologias do sistema ao trabalho em questão. Em relação aos serviços *Web* semânticos, já havia alguns prontos, os que são frequentemente usados em STIs, porém, os serviços de exclusividade do ambiente criado devem ser implementados. Em relação ao uso de serviços

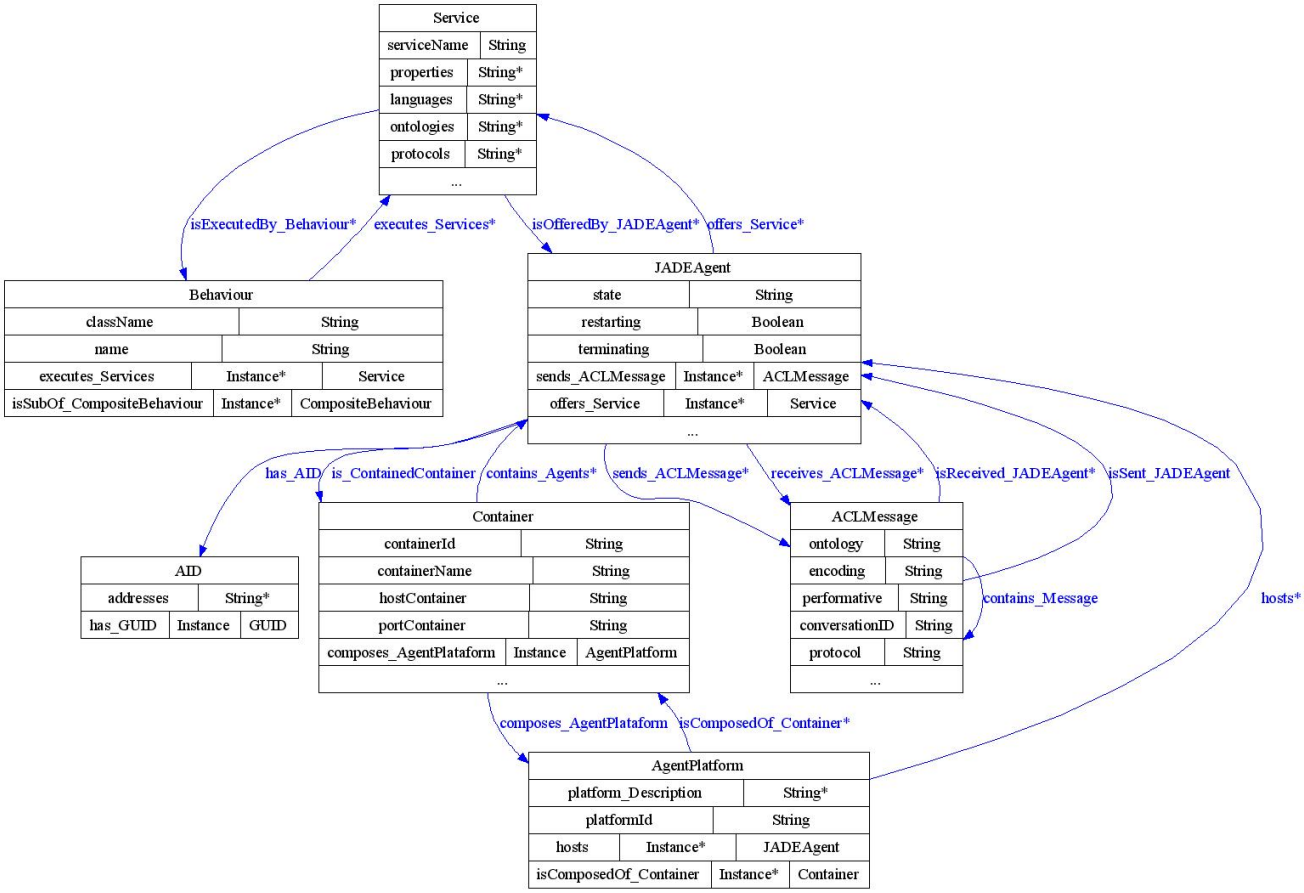

Figura 5.27: Ontologia para mapeamento em JADE do Massayo [47]

Web semânticos, o uso destes tornou o ambiente lento devido à pesquisa de cada agente pelo serviço que usará, bem como a execução de tais serviços.

5.6 Conclusão

Este capítulo apresentou o FraW e como ele foi desenvolvido: sua arquitetura, apresentando seu nível lógico e físico; especificações em ontologias, tanto as ontologias de descrição dos agentes contidos no FraW, os quais o MASSAYO disponobiliza, que mapeam as descrições dos agentes em GAIA para uma outra ontologia de configurações JADE, quanto as ontologias dos módulos dos agentes tutores: modelo do aprendiz, pedagógico e de domínio. Além disso, também foram apresentados os serviços semânticos criados e que estão no ambiente FraW para que seus agentes possam fazer uso deles.

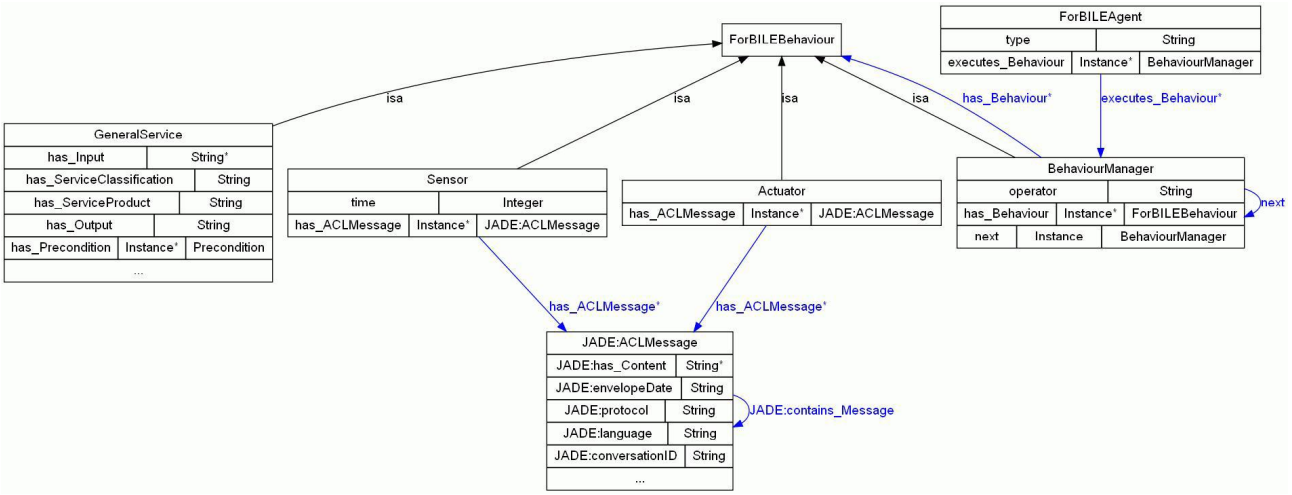

Figura 5.28: Parte da ontologia ForBILEAgent [47]

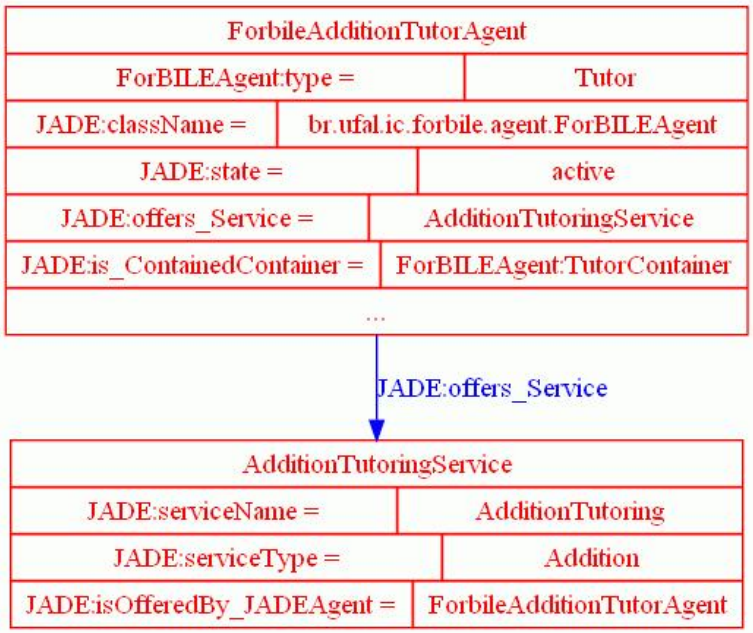

Figura 5.29: Exemplo de regra de mapeamento GAIA-JADE/ForBILEAgent no FraW

Capítulo 6

Avaliação e Refinamento dos Resultados

6.1 Pré-requisitos para o FraW

Alguns requisitos importantes para o bom funcionamento do FraW estão descritos a seguir:

- **Suporte a diferentes perfis de usuários**: possibilidade de haver vários perfis de usuários no sistema, como: aluno, professor, monitor, pais;
- **Eficácia Pedagógica**: diversidade de pedagogia para ensinar um mesmo assunto;
- **Envolvimento de pares humanos e computacionais**: interação entre pares humanos ou computacionais para a resolução de alguma tarefa;
- **Usabilidade**: facilidade dos usuários em utilizar o ambiente o qual estão interagindo;
- **Adaptabilidade**: capacidade do ambiente em se adaptar ao perfil do usuário;
- **Facilidade de Evolução**: capacidade de acrescentar facilmente novas funções ao ambiente;
- **Distribuição**: habilidade em distribuir a carga de acessos do ambiente para diferentes máquinas servidoras;
- **Escalabilidade**: habilidade em manipular uma crescente quantidade de máquinas clientes acessando o ambiente;

- **Interatividade**: possibilidade de manter interação entre pessoas ou serviços no ambiente;
- **Flexibilidade**: capacidade de uso de ferramentas ou de diferentes estratégias de localização/recomendação de agentes tutores/pares, ou de poder utilizar diversas ferramentas para tais interações;
- **Confiabilidade**: possibilidade de usar o sistema sem a ocorrência de falhas ou erros visíveis nele;
- **Desempenho**: ter um bom desempenho no funcionamento do sistema;
- **Disponibilidade**: possibilitar a disponibilidade do sistema apresentando redundância/réplica dos dados.

Portanto, pensando nessas características, foi pensado em uma melhor arquitetura que possibilitasse a melhora de algumas dessas características já pensadas, ou a implantação de algumas dessas características.

6.2 Arquitetura Refinada do FraW

Na Figura 6.1 é apresentada uma arquitetura idealizada para o sistema FraW. A descrição dela é como se segue. Em (1), tem-se um aprendiz que interage através da interface, representada na figura por IHC (Interface Humano Computador), com o sistema. A IHC do sistema sana o requisito de **usabilidade**, pois este dá um grande suporte e comodidade ao aprendiz de Fração. É em (1) que deve ser sanado também o requisito **suporte a diferentes perfis de usuários**, podendo este ser um aprendiz, professor, tutor, entre outros. O padrão de projeto MVC (*Model View Controller*) é usado para desacoplar a Interface Gráfica do resto do sistema.

O agente que irá dar suporte a interação do ator, assim que ele entra no sistema, é o agente Mediador, o qual interage com os diferentes perfis e com os agentes tutores, além de fazer uso das ontologias e das Ferramentas de Interação. É em (2) que pode ser visto a **adaptabilidade**, a medida que o Mediador se adapta ao perfil do usuário e dá o suporte necessário a ele, e também a **facilidade de evolução**, como a adição de um novo agente Tutor, visto que no Mediador podem ser mudadas as regras de negócio, sem precisar modificar os agentes, já que é no Mediador que o controlador (*Controller* do MVC) está centralizado.

Inicialmente, quando o aprendiz, ou outros perfis, entra no sistema, o Mediador irá buscar através do Controlador de Localização (3) uma entidade que

irá dar suporte ao aprendiz. Neste caso inicial, a busca será feita pelo Localizador de Tutor Computacional, o qual retornará para o Mediador um Agente Tutor que o Aprendiz irá interagir no momento, a partir de então, a interação entre o Mediador e o Agente Tutor será *peer-to-peer* (P2P) (7), o qual será implementado por algum protocolo entre os agentes de *software*. Assim, em (3) pode ser visto o uso do requisito de **flexibilidade**, onde há várias estratégias de localização para localização de tutores e pares.

O mediador é um ponto único de acesso para a interface gráfica. Em outras palavras, ele pode ser visto como um controlador principal, que pode delegar tarefas para sub-controladores, tais como o Controlador de Localização.

É interessante observar que o Agente Tutor faz uso de três bases de dados, que são: modelo do aprendiz, do domínio e pedagógico, como pode ser visto em (4), isso possibilita a **confiabilidade** do ambiente, visto que quando um agente Tutor estiver indisponível, outro agente Tutor poderá guiar o aprendiz. Além disso, a replicação local dos dados sobre os aprendizes, modelo pedagógico e de domínio, torna a **disponibilidade** possível, pois a redundância de tais dados permite que esses dados sejam usados por outro tutor, mesmo que a base de dados principal esteja inativa. Por fim, assim como há vários agentes Tutores, sendo cada um responsável por um determinado Currículo, para que haja a **eficácia pedagógica**, pode-se reproduzir mais de um agente Tutor para cada Currículo, para que cada um possa usar sua pedagogia de forma diferente. Por exemplo, pode have um agente Tutor de Adição para criança, e outro para adultos.

A partir da interação entre o aprendiz (ou outro perfil) e um agente Tutor, quando há a necessidade do aprendiz de interagir com alguma outra entidade, o Agente Tutor informa ao Medidor da necessidade e este procurará através do Controlador de Localização um outro Tutor Computacional ou Humano. Assim como também, caso o próprio agente Tutor necessite da ajuda de um outro agente Tutor, este também informará ao Controlador de Localização, tornando assim possível o **envolvimento de pares humanos e computacionais**. No caso de se fazer necessário um Tutor Humano, é usado o Localizador de Tutor Humano e, a partir do encontro de um Tutor Humano para o aprendiz, o Mediador fará uso de uma Ferramenta de Interação (Fer. Int.) (5) através de uma conexão HTTP (6) para possibilitar a interação entre o Aprendiz e o Tutor Humano. Tal ferramenta de interação (5) pode ser usada de forma a dar **flexibilidade** de uso, podendo ser um *chat*, um fórum, ou qualquer outra ferramenta. Quanto a conexão HTTP em (6), faz-se presente os requisitos de **distribuição de carga** e de **escalabilidade**.

O conector assinalado como (6), é capaz de promover várias conexões en-

tre o Mediador e os servidores de Ferramentas de Interação. Dessa forma, é possível implementar um mecanismo de monitoramento dos servidores, de forma a evitar sobrecarga de acesso, balanceando o uso dos recursos entre um conjunto de servidores disponíveis. Além de aumentar a escalabilidade do sistema, esse mecanismo de distribuição de carga pode aumentar a qualidade do serviço oferecido, proporcionando inclusive melhor desempenho.

Em (8), tem-se que quando algum Tutor entrar no sistema, ele deve informar ao Controlador de Localização e apresentar a ele suas informações, como identificação e seu domínio de ensino, através de um Protocolo de Entrada/-Saída.

O Mediador, assim como os Agentes Tutores, também faz uso das ontologias referentes ao aprendiz, domínio e pedagógico. A base de informação na qual essas ontologias fazem parte é unificada, e para cada Agente que a usa, este deve replicá-la. Um outro ponto é que esta base está inserida em um componente baseado no padrão *Blackboard*, o qual é reativo e informa ao Mediador suas atualizações assim que elas acontecem, realizando, então, as devidas atualizações que os Agentes Tutores requerem, melhorando o **desempenho** destes agentes tutores.

Para implementação dos Agentes Tutores e das Ferramentas de Interação podem ser utilizados os Serviços *Web* Semânticos, o que possibilitaria a **interatividade** entre serviços, porém, esta forma de implementação pode prejudicar o desempenho e escalabilidade do sistema, devido ao custo da pesquisa e *matching* dos serviços, além das consultas feitas nas ontologias.

As interações representadas na Figura 6.1 serão apresentados nas próximas Seções: 6.3, 6.4 e 6.5.

6.3 Interação: Agente Tutor com outros Agentes Tutores do sistema

Um Agente Tutor pode interagir com outro Tutor diretamente (P2P), para isso é preciso que um dos tutores precise de algum serviço que um outro Tutor tenha. Este Tutor que necessita desde serviço fará a solicitação ao Controlador de Localização, o qual retornará um outro Tutor que faça o serviço que ele necessita, para, então, haver a conexão direta entre os tutores.

Um exemplo deste tipo de interação pode ser ilustrado com o seguinte cenário: o Tutor de Adição faz a soma de frações, mas, durante a soma de duas frações com denominadores diferentes, há a necessidade de se tirar o mínimo múltiplo comum (MMC) antes e o Tutor de Adição não é responsável por isso.

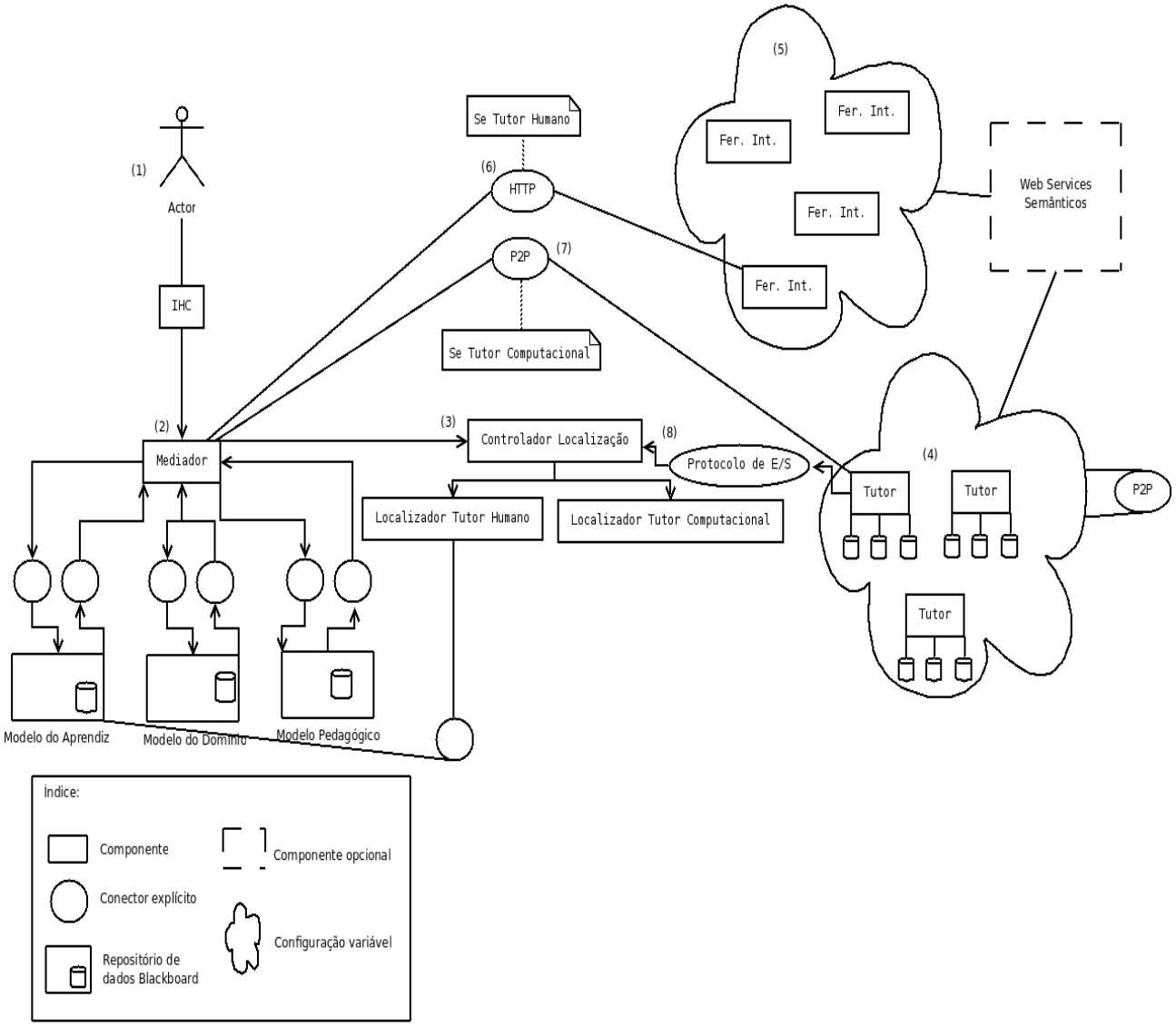

Figura 6.1: Arquitetura refinada do FraW

Para tanto, o Tutor de Adição iria fazer a requisição de tal Tutor que tenha esse serviço através do Controlador de Localização, o qual, recebendo o Tutor de MMC através do Localizador de Tutor Computacional, habilita uma interação P2P entre eles.

6.4 Interação: Agente Tutor com Aprendiz

A interação entre um Agente Tutor e um Aprendiz se dá de forma direta (P2P) assim que o agente Mediador faz a ligação entre eles. Ou seja, assim que o aprendiz entra no sistema, o Mediador observará sua modelagem e, a partir dela, fará a solicitação de um agente Tutor para que o aprendiz continue seus estudos. Também haverá essa interação quando o aprendiz termina os estu-

dos com um Tutor e começa a interagir com outro, este outro será indicado pelo Mediador, assim que o Agente Tutor inicial avisar que ele terminou seus estudos.

Um cenário que pode exemplificar é o seguinte: o aprendiz José entra no sistema e o Mediador poderá, a partir da visualização de sua modelagem, observar que ele começou os estudos com o Tutor de Adição, mas não terminou, portanto, José será colocado em contato com o Tutor de Adição. Logo após o término do estudo de José com o Tutor de Adição, este informará ao Mediador que José terminou o estudo de Adição, o qual fará a pesquisa pelo próximo Tutor responsável pelo próximo estudo de José.

6.5 Interação entre Aprendizes

A interação entre os aprendizes acontece quando, a partir da interação do aprendiz com um Tutor, o Tutor vê a necessidade de colocar o aprendiz que ele está ensinando em contato com outro aprendiz.

Um exemplo da interação entre aprendizes é o caso em quem o aprendiz José está estudando sobre multiplicação de fração e o tutor responsável por este currículo apresenta a José algumas questões. Em algum momento, José tenta fazer uma questão e não consegue, por algum erro de concepção na sua aprendizagem, ou outro motivo. Quando o Tutor de Multiplicação nota seu impasse, apresenta a ele uma dica de como fazer a questão e se, mesmo tendo recebido a dica, José não consiga resolver, o Tutor de Multiplicação informará ao Mediador que José está com o impasse o qual ele não consegue resolver. O Mediador irá solicitar ao Controlador de Localização uma ajuda, esta ajuda pode vir na forma de Tutor Humano. Para que ambos, aprendiz e Tutor Humano, possam discutir acerca do assunto, o Mediador irá colocá-los em contato através de uma Ferramenta de Interação, como, por exemplo, o *chat*.

Capítulo 7

Conclusão e Trabalhos Futuros

O trabalho aqui apresentado traz um ambiente que une as vantagens dos STIs, acrescentando-lhes novas funcionalidades para a construção do sistema educacional FraW voltado ao ensino de frações. O presente trabalho apresentou uma proposta diferenciada para ambientes interativos de aprendizagem na *Web*, partindo da arquitetura clássica de STI, acrescentando-lhe novas funcionalidades. Neste sentido, investiu-se numa abordagem que integra Sistemas Multiagentes e Serviços *Web* Semânticos para o desenvolvimento de tais sistemas.

Especificamente, apresentou-se uma arquitetura adequada para se conseguir reduzir os custos no desenvolvimento e manutenção dos sistemas em questão, oferecendo, assim, flexibilidade com bom suporte ao reuso. Além disso, apresentou-se o modelo aberto do aprendiz, juntamente com um mecanismo de suporte a negociação de conflitos entre a opinião do estudante e o conceito inferido pelo sistema.

Também foi apresentado um sistema multiagente com tutores que oferecem interações adaptativas às características cognitivas do aprendiz, oferecendo um mecanismo de recomendação que possibilite a interação entre pares, sendo tal interação feita através de uma ferramenta de *chat*, tendo apontado também uma posterior análise do resultado dessa interação para verificar a qualidade da interação.

Apresentou-se ainda como o sistema FraW foi desenvolvido, a partir de um estudo de caso com vistas a demonstrar a facilidade na construção de STIs a partir da utilização de um *framework* que possibilita o reuso de ontologias que representam bem a taxonomia do domínio de um STI. Vale ainda ressaltar que o estudo de caso demonstra as vantagens do uso de tal *framework* em relação ao desenvolvimento de agentes de *software*, dado que o *framework* provê os agentes básicos de manutenção e evolução do sistema, deixando ao

desenvolvedor a tarefa de especificar e implementar os agentes tutores, sendo tal tarefa amenizada pelo fato do *framework* oferecer uma estrutura básica para esses agentes tutores.

Portanto, o sistema FraW possibilita ao aprendiz o estudo sobre Adição, Subtração, Multiplicação, Divisão e Potenciação de fração, de forma que o aprendiz é acompanhado constantemente pelo sistema e suas interações são diagnosticadas para construir um modelo acerca dele. E, dependendo de sua necessidade, o aprendiz pode interagir ou não com outro aprendiz.

Observou-se, após a implementação do sistema FraW, que, devido aos constantes acessos às ontologias, à pesquisa e ao *matching* de *Web Services* semânticos, o sistema foi prejudicado em relação ao desempenho e escalabilidade. A resposta às interações do aprendiz está muito dispendiosa. Estudos para que seu desempenho e escalabilidade melhorem estão sendo feitos.

Como trabalho futuro, temos o teste com o público alvo, alunos das séries inicias do ensino fundamental, que será feito assim que o ambiente for finalizado. Além deste teste, também será feita uma avaliação sistemática do FraW, dando ênfase na observação de sua usabilidade/aplicabilidade.

Ainda como trabalho futuro, tem-se a melhora do serviço de recomendação do par-complementar, para que cada aprendiz possa descrever em seu perfil a disponibilidade em ajudar um outro estudante, além de poder também descrever se em situação de receber ajuda, não gostaria de receber auxílio de um certo aprendiz; bem como incentivar os aprendizes a ajudar. Outro aspecto é a recomendação não de apenas um aprendiz, mas de um grupo de estudantes, enriquecendo a interação e a troca de conhecimento entre os alunos.

Um outro trabalho futuro será a melhora no serviço de mineração de texto, que poderia agir ao longo da conversa entre estudantes e também influenciar na conversa, chamando a atenção para a volta do assunto em questão, por exemplo. Em relação ao modelo aberto do aprendiz, uma melhora na negociação pode ser feita implementando uma maneira mais sofisticada na interação entre o aprendiz e o tutor, algo como a criação de um *chatterbot*.

Por

As publicações relacionadas à pesquisa realizada no contexto desta dissertação são as seguintes: [23, 53, 18].

Bibliografia

[1] Aguera, A. S., Guerra, A. & Martínez, M. (2000), Memory based reasoning and agents adaptive behavior, *in* 'MICAI '00: Proceedings of the Mexican International Conference on Artificial Intelligence', Springer-Verlag, London, UK, pp. 610–620.

[2] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D. & Patel-Schneider, P. F., eds (2003), *The Description Logic Handbook: Theory, Implementation, and Applications*, Cambridge University Press.

[3] Barbosa, D. N. F., Sarmento, D. F., Barbosa, J. L. V. & Geyer, C. F. R. (2008), 'Em Direção a Educação Ubíqua: Aprender Sempre, em Qualquer Lugar, com Qualquer Dispositivo', *RENOTE - Revista Novas Tecnologias na Educação* **6**, 1–10.

[4] Barros, B., Verdejo, F., Read, T. & Mizoguchi, R. (2002), Applications of a Collaborative Learning Ontology, *in* 'MICAI '02: Proceedings of the Second Mexican International Conference on Artificial Intelligence', Springer-Verlag, London, UK, pp. 301–310.

[5] Bellifemine, F. L., Caire, G. & Greenwood, D. (2007), *Developing Multi-Agent Systems with JADE*, Wiley.

[6] Bittencourt, I. I. (2006), Plataforma para Construção de Ambientes Interativos de Aprendizagem Baseados em Agentes, Master's thesis, Universidade Federal de Alagoas.

[7] Bittencourt, I. I., Bispo, P., de Barros Costa, E., Pedro, J., Veras, D., Dermeval, D. & Luna, H. P. L. (2009*a*), Modeling jade agents from gaia methodology under the perspective of semantic web, *in* 'ICEIS09', pp. 780–789.

[8] Bittencourt, I. I., Costa, E., Silva, M. & Soares, E. (2009*b*), 'A computational model for developing semantic web-based educational systems', *Knowledge-Based Systems* **22**, 302–315.

[9] Bolzan, W. & Giraffa, L. M. M. (2002), Estudo comparativo sobre sistemas tutores inteligentes multiagentes web, Technical report, URCAMP.

[10] Brooks, C., Panesar, R. & Greer, J. (2006), Awareness and Collaboration in the iHelp Courses Content Management System, *in* 'in Proc. of 1st European Conference on Technology Enhanced Learning (EC-TEL 2006)'.

[11] Bull, S. & Kay, J. (2007), 'Student Models that Invite the Learner In: The SMILI Open Learner Modelling Framework', *International Journal of Artificial Intelligence in Education* **17**(2), 89–120.

[12] Bull, S., Gardner, P., Ahmad, N., Ting, J. & Clarke, B. (2009), Use and trust of simple independent open learner models to support learning within and across courses, *in* 'UMAP', pp. 42–53.

[13] Calado, I., Barros, H. & Bittencourt, I. I. (2009), An approach for semantic web services automatic discovery and composition with similarity metrics, *in* 'SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing'.

[14] Chaachoua, H., cois Nicaud, J.-F., Bronner, A. & Bouhineau, D. (2004), APLUSIX, A Learning Environment For Algebra, Actual Use and Benefits, *in* 'ICME 10 : 10th International Congress on Mathematical Education'. Retrieved from `http://www.itd.cnr.it/telma/papers.php`, January 2007.

[15] Consortium, I. G. L. (2001), 'IMS Learner Information Package Model Specification', Disponível em http://www.imsglobal.org/profiles/index.html. Acessado em 05, Mai. de 2009.

[16] Consortium, W. W. W. (2006), 'Extensible Markup Language (xml)', Disponível em http://www.w3.org/TR/REC-xml/. Acessado em 15, dez. de 2009.

[17] Cunha, D. (2002), 'Web Services, SOAP e Aplicações Web', Disponível em http://devedge-temp.mozilla.org/viewsource/2002/soap-overview/index_pt_br.html.

[18] da S. Santos, P. B., Sibaldo, M. A. A., de Barros Costa, E., Bittencourt, I. I., e Silva, D. V., Luna, H. P. L. & Filho, M. H. C. L. (2009), Suporte interacional adaptativo na web focalizando agentes humanos e artificiais em situações de resolução de problemas: Um estudo de caso no domínio de fração, *in* 'Anais do XX Simpósio Brasileiro de Informática na Educação'.

[19] da Silva Andre, M. C. & Franco, S. R. K. (2008), Aprendizagem Colaborativa: Estudo da Co-Operação de Uma Comunidade Virtual de Aprendizagem em Inglês, *in* 'Anais do XIX Simpósio Brasileiro de Informática na Educação'.

[20] da Silva Jacinto, A. (2006), Uma arquitetura para sistemas tutores inteligentes apoiada por fundamentos da web semântica, Master's thesis, Instituto Tecnológico de Aeronáutica.

[21] da Silva Jacinto, A. & de Oliveira, J. M. P. (2008), 'An ontology-based architecture for intelligent tutoring system', *Scientia - Interdisciplinary Studies in Computer Science* **19**, 25–35.

[22] das Neves, A. F. R. (2009), Resolução e avaliação de problemas em um sistema tutor inteligente, Universidade Federal de Alagoas.

[23] das Neves, A. F. R., Sibaldo, M. A. A., Medeiros, F. M., Bittencourt, I. I. & de Barros Costa, E. (2008), 'Um Sistema Tutor Inteligente Para o Ensino de Fração', *Anais do XXVIII Congresso da SBC - XIV Workshop sobre Informática na Escola (WIE2008)* pp. 322–331.

[24] de Almeida e Silva, A. T. (2009), Interfaces para Entrada e Visualização de Expressões Matemáticas no FraW, Universidade Federal de Alagoas.

[25] de Barros Costa, E. (1997), Um Modelo de Ambiente Interativo de Aprendizagem Baseado Numa Arquitetura Multi-Agentes, PhD thesis, Universidade Federal da Paraíba.

[26] Ford, L. (2008), 'A New Intelligent Tutoring System', *British Journal of Educational Technology* **39**, 311–318.

[27] Frigo, L. B. (2007), Um Modelo de Autoria para Sistemas Tutores Adaptativos, PhD thesis, Universidade Federal de Santa Catarina.

[28] Frigo, L. B., Pozzebon, E. & Bittencourt, G. (2004), O papel dos agentes inteligentes nos sistemas tutores inteligentes, *in* 'Book of Abstracts: Engineering Education in the Changing Society'.

[29] Gamboa, H. & Fred, A. L. N. (2001), Designing Intelligent Tutoring Systems: A Bayesian Approach, *in* 'ICEIS (1)', pp. 452–458.

[30] Greer, J., Mccalla, G., Collins, J., Kumar, V., Meagher, P., Vassileva, J. & Laboratory, A. (1998*a*), 'Supporting Peer Help and Collaboration in

Distributed Workplace Environments', *International Journal of Artificial Intelligence in Education* **9**, 159–177.

[31] Greer, J., Mccalla, G., Cooke, J., Collins, J., Kumar, V., Bishop, A. & Vassileva, J. (1998*b*), The Intelligent Helpdesk: Supporting Peer-Help in a University Course, *in* 'in Proc. of the International Intelligent Tutoring Systems Conference (ITŚ98)', Springer-Verlag, pp. 494–503.

[32] Gruber, T. R. (1993), 'A translation approach to portable ontology specifications', *Knowledge Acquisition* **5**, 199–220.

[33] Gruber, T. R. (1995), 'Toward Principles For The Design Of Ontologies Used For Knowledge Sharing', *International Journal of Human and Computer Studies* **43**, 907–928.

[34] Hockemeyer, C., Conlan, O., Wade, V. P. & Albert, D. (2003), 'Applying Competence Prerequisite Structures for eLearning and Skill Management.', *Journal of Universal Computer Science* **9**(12), 1428–1436. URL http://dblp.uni-trier.de/db/journals/jucs/jucs9.html#COVD03.

[35] Isotani, S., Isotani, N. & Isotani, S. (2008), Ontologias e Web Semântica no Suporte ao Ensino Colaborativo em Salas de Aula Presenciais, *in* 'Anais do XIX Simpósio Brasileiro de Informática na Educação'.

[36] Kerly, A. & Bull, S. (2006), The potential for chatbots in negotiated learner modelling: A wizard-of-oz study, *in* 'Intelligent Tutoring Systems', pp. 443–452.

[37] Kerly, A. & Bull, S. (2007), 'Open Learner Models: Opinions Of School Education Professionals', *Artificial Intelligence in Education*.

[38] Kerly, A., Ahmad, N. & Bull, S. (2008), Investigating learner trust in open learner models using a 'wizard of oz' approach, *in* 'Intelligent Tutoring Systems', pp. 722–724.

[39] Kerly, A., Hall, P. & Bull, S. (2007), 'Bringing chatbots into education: Towards natural language negotiation of open learner models', *Knowl.-Based Syst.* **20**(2), 177–185.

[40] Lab, T. I. (2007), 'Java Agent DEvelopment Framework', Disponível em http://jade.tilab.com/. Acessado em 19, Jul. de 2009.

[41] McCalla, G., Vassileva, J., Greer, J. & Bull, S. (2000), 'Active Learner Modelling', *Proc. Int'l Conf. Intelligent Tutoring Systems (ITS '00)* **1**, 53–62.

[42] Mcilraith, S. A., Son, T. C. & Zeng, H. (2001), 'Semantic web services', *IEEE Intelligent Systems* **16**, 46–53.

[43] Melis, E., Andres, E., Budenbender, J., Frischauf, A., Goguadze, G., Libbrecht, P., Pollet, M. & Ullrich, C. (2001*a*), 'Activemath: A generic and adaptive web-based learning environment', *International Journal of Artificial Intelligence in Education* **12**, 385–407.

[44] Melis, E., Andres, E., Goguadse, G., Libbrecht, P., Pollet, M. & Ullrich, C. (2001*b*), 'Activemath: System description'.

[45] ODonovan, J. & Smyth, B. (2005), Trust In Recommender Systems, *in* 'IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces', ACM, pp. 167–174.

[46] Payne, T. & Lassila, O. (2004), 'Guest editorsíntroduction: Semantic web services', *IEEE Intelligent Systems* **19**(4), 14–15.

[47] Pinto, I. I. B. S. (2009), Modelos e Ferramentas para a Construção de Sistemas Educacionais Adaptativos e Semânticos, PhD thesis, Universidade Federal de Campina Grande.

[48] Pozzebon, E. (2008), Um Modelo Para Suporte ao Aprendizado em Grupo em Sistemas Tutores Inteligentes, PhD thesis, Universidade Federal de Santa Catarina.

[49] Protégé (n.d.), 'The protégé ontology editor and knowledge acquisition system', Disponível em http://protege.stanford.edu/. Acessado em 05, Nov. de 2009.

[50] Rau, M. A. (2008), Flexible knowledge of fractions with multiple graphical representations in intelligent tutoring systems, Master's thesis, Albert-Ludwigs-Universitat Freiburg im Breisgau.

[51] Rezende, S. O., ed. (2003), *Sistemas Inteligentes - Fundamentos e Aplicações*, Manole, Barueri, SP, Brasil.

[52] Self, J. (1990), Theoretical Foundations For Intelligent Tutoring Systems, Technical report, Lancaster University.

[53] Sibaldo, M. A. A., das Neves, A. F. R., Medeiros, F. M., Bittencourt, I. I. & de Barros Costa, E. (2008), FraW - Ambiente Interativo de Aprendizagem para o Domínio de Fração Via Web, *in* 'XIX Simpósio Brasileiro de Informática na Educação'.

[54] Sibaldo, M. A. A., de Sales, T. B. M., Calado, I. A. A. R., Bittencourt, I. I. & de Barros Costa, E. (2007), Mobile GraW: Uma Aplicação para Dispositivos Móveis Baseada em Comunidades Virtuais de Aprendizagem com Suporte a Recomendação, *in* 'Anais do XVIII Simpósio Brasileiro de Informática na Educação'.

[55] Stahl, G., Koschmann, T. & Suthers, D. (1996), 'Computer-Supported Collaborative Learning: An Historical Perspective', *Lawrence Erlbaum Associates, Mahlwah, NJ.*

[56] Studer, R., Benjamins, V. R. & Fensel, D. (1998), 'Knowledge Engineering: Principles and Methods', *Data Knowledge Engineering* **25**(1-2), 161–198.

[57] Su, W., Wang, P., Li, L., Li, G. & Zhao, Y. (2006), 'MathEdit, A Browser-Based Vsual Mathematics Expression Editor', *Proceeding of The 11th Asian Technology Conference in Mathematics* pp. 271–279.

[58] Su, W., Wang, P. S. & Li, L. (2007), An on-line mathml editing tool for web applications, *in* 'IMSCCS', pp. 458–464.

[59] Vassileva, J. & Deters, R. (2001), 'Lessons From Deploying i-Help', *Proceedings of the Workshop Multi-Agent Architectures for Distributed Learning Environments. Procedings International Conference on AI and Education.*

[60] Vassileva, J., Deters, R., Greer, J., Mccalla, G., Kumar, V. & Mudgal, C. (1998), A Multi-Agent Architecture For Peer-Help In A University Course, *in* 'Proceedings of the Workshop on Pedagogical Agents at ITŚ98', pp. 64–68.

[61] Wenger, E. (1987), *Artificial Intelligence and Tutoring Systems: Computacional and cognitive approaches to the Communication of Knowledge*, Morgan Kaufmann.

[62] Winter, M., Brooks, C., Mccalla, G. & Greer, J. (2004), Using semantic web methods for distributed learner modelling, *in* 'Proceedings on the Workshop on Using the Semantic Web in ELearning at the 3rd International Semantic Web Conference'.

[63] Winter, M., Daniel, B. & Brooks, C. (2005), Towards Automatic Discovery of Peer Helpers from a Large Message Board System, *in* 'Workshop on Usage Analysis in Learning Systems, the 12th International Conference on Artificial Intelligence in Education (AIED 2005).'.

[64] Wooldridge, M. (2002), *An Introduction to MultiAgent Systems*, John Wiley & Sons, LTD.

[65] Zambonelli, F., Jennings, N. R. & Wooldridge, M. (2003), 'Developing Multiagent Systems: The Gaia Methodology', *ACM Transactions on Software Engineering and Methodology* **12**, 317–370.