Universidade do Vale do Rio dos Sinos (UNISINOS)
Unidade Acadêmica de Pesquisa e Pós-Graduação
Programa Interdisciplinar de Pós-Graduação em Computação Aplicada (PIPCA)
Nível Mestrado

FÁBIO RAFAEL DAMASCENO

CONCEPÇÃO E DESENVOLVIMENTO DO AGENTE TUTOR E MODELO DE ALUNO NO AMBIENTE INTELIGENTE DE APRENDIZAGEM PAT2MATH

São Leopoldo
2011

Fábio Rafael Damasceno

CONCEPÇÃO E DESENVOLVIMENTO DO AGENTE TUTOR E MODELO DE ALUNO NO AMBIENTE INTELIGENTE DE APRENDIZAGEM PAT2MATH

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós Graduação em Computação Aplicada (PIPCA) da Universidade do Vale do Rio dos Sinos

Orientadora: Dr.ª Patrícia Augustin Jaques Maillard

São Leopoldo

2011

D155c Damasceno, Fábio Rafael

Concepção e desenvolvimento do agente tutor e modelo de aluno no ambiente inteligente de aprendizagem PAT2MATH / por Fábio Rafael Damasceno. – São Leopoldo, 2011.

102 f. : il. color. ; 30 cm.

Dissertação (mestrado) – Universidade do Vale do Rio dos Sinos, Programa Interdisciplinar de Pós-Graduação em Computação Aplicada, São Leopoldo, RS, 2011.

Orientação: Profª. Drª. Patrícia Augustin Jaques Maillard, Ciências Exatas e Tecnológicas.

1. Agentes inteligentes (software). 2. Agentes inteligentes (software) – PAT2MATH. 3.Sistemas tutores inteligentes. 4.Objetos de aprendizagem – Plataformas digitais. 5.Educação – Estudo e ensino – Ensino auxiliado por computador. I.Maillard, Patrícia Augustin Jaques. II.Título.

CDU 004.89
37: 004.89

Catalogação na publicação:
Bibliotecária Carla Maria Goulart de Moraes – CRB 10/1252

Dedico este trabalho aos meus pais e àqueles

que estiveram junto comigo durante

estes anos de pesquisa.

AGRADECIMENTOS

Apoio daqueles que nos são especiais é fundamental no sucesso de qualquer projeto em nossas vidas. Agradeço a:

Meus pais, Milton e Cláudia, pelo sempre presente apoio nas coisas que fiz;

Dinorá Fraga, por me iniciar na vida de pesquisador como bolsista durante a Graduação;

João Ricardo Bittencourt, por me incentivar a seguir a carreira de pesquisador e aspirar à docência;

Minha orientadora, Patrícia, por demonstrar sempre muito interesse nas minhas atividades, bem como pensar nas possibilidades futuras para mim como pesquisador e docente.

RESUMO

No Brasil os alunos enfrentam dificuldades no aprendizado de Álgebra, considerado um expoente entre assuntos relacionados à Matemática. Os Sistemas Tutores Inteligentes (STI) oferecem a alternativa de um ensino individualizado para alunos, cenário em que o PAT2Math contribui, pois trata-se de um STI web para o aprendizado de Álgebra que considera as habilidades de cada aluno individualmente no seu ensino. Utilizar estas informações individuais como variáveis de decisão de diferentes estratégias pedagógicas permitindo a construção do ensino de caráter individual constitui um desafio de um Módulo Tutor em um STI. Na área da Matemática, as variadas maneiras de se resolver problemas dificultam a inferência do quão certo um aluno está em um exercício e do quanto ele domina de um dado conteúdo, informações que devem ser inferidas e guardadas por Modelo do Aluno de um STI. O trabalho proposto neste documento vem a ser a definição do Modelo de Aluno e do Módulo Tutor do sistema, tendo em vista o atual desafio de como é a estruturação computacional do processo de ensino em um ambiente desta categoria. Devido à arquitetura multiagente do STI PAT2Math, esses dois módulos também serão implementados como agentes. O monitoramento das características dos alunos, realizado através do seu respectivo Modelo de Aluno, constitui as variáveis observadas pelo Agente Tutor para ensiná-los de maneira individual. O Agente Tutor, para concretizar este caráter individual, utiliza Planos de Ensino, adapta os mesmos em caso de erro do aluno e aplica uma estratégia pedagógica, resultando em conteúdos e exercícios propostos considerando as habilidades do aluno em questão. Os Módulos Tutor e Modelo de Aluno foram concebidos sob teorias de aprendizado cognitivas, como a Aprendizagem Significativa de Ausubel, Taxonomia de Bloom, além da Teoria de Design Instrucional de Merril. Uma avaliação qualitativa foi realizada com um grupo de quatro pesquisadores, docentes de Matemática, com o objetivo de avaliar a usabilidade e as possibilidades de real aplicação em salas de aula do protótipo construído. Entre os resultados coletados pode-se citar a forte tendência positiva nas respostas do questionário aplicado, que visava descobrir a opinião de tais avaliadores sobre a qualidade do protótipo. Através dos comentários realizados pelos avaliadores pode-se dizer que o tutorial dentro do protótipo está bem organizado e de fácil exploração para o estudante, independente do amadurecimento nos conteúdos de Matemática, havendo a possibilidade de maiores explorações conforme o próprio professor inserir conteúdos e modificar a base de domínio. Foi considerada interessante a utilização da ferramenta, proporcionando uma forma diferente de aprendizado. Isto foi considerado pelos avaliadores algo muito relevante, pois a diversidade de opções em busca de atender todos os modos de aprendizagem mobiliza de forma benéfica o estudante de Matemática. Também foi salientado que complementar a este protótipo seria o professor observar quando possível as ações dos estudantes para obter um pouco mais de informações sobre o seu respectivo desempenho e aprimorar o projeto como um todo.

ABSTRACT

In Brazil the students face difficulties in learning algebra, considered an exponent of issues related to mathematics. Intelligent Tutoring Systems (ITS) offer the alternative of an individualized learning for students, setting in which PAT2Math contributes because it is an ITS on web for learning algebra which considers the abilities of each individual student in their education. The usage of this information about individual issues, as decision variables of different pedagogical strategies for teaching making individual learning, poses a challenge to a Tutor Module in an ITS. In the area of mathematics, the various ways of solving difficult problems make hard the inference of how sure a student is in an exercise and how much it dominates of a given content information to be inferred and stored by the Student Model of an ITS. The work proposed in this document comes to be the definition of the Student Model Agent and Tutor Agent of the system, considering the current challenge of how is the computational structure of the teaching process in an environment of that category. Due to the multi-agent architecture of the STI PAT2Math, these two modules will also be implemented as agents. The monitoring of students' characteristics, performed through their respective Student Model, are the observed variables by the Tutor Agent to teach them individually. The Tutor Agent, to achieve this individual feature, use teaching plans, adapting them in case of student error and applies a pedagogical strategy, resulting in content and exercises proposed considering the abilities of the student in question. Tutor and Student modules were designed in cognitive learning theories such as the Meaningful Learning of Ausubel, Bloom's Taxonomy, and the Theory of Instructional Design Merrill. A qualitative evaluation was performed with a group of four researchers, teachers of mathematics, with the aim of evaluating the usability and potential application in real classrooms about the built prototype. Among the data collected it can be mentioned the strong positive trend in the responses of the questionnaire, which aimed to discover the opinions of those reviewers on the quality of the prototype. Through the comments made by reviewers it can be said that the tutorial inside the prototype is well organized and of easy exploration for the student, regardless of its maturation in the mathematics, with the possibility of larger experiments as the teacher himself inserts and modifies the contents in the domain base. It was considered of interesting use the tool, because it provides a different way of learning. The evaluators considered very relevant the diversity of options in search of ways to suit all types of learning, turning in a beneficial way for the math student. It was also noted that additional to this prototype it would be the teacher observing actions of the students to get a little more information on their respective performance and improve the project as a whole.

A educação faz um povo fácil de ser liderado, mas difícil de ser dirigido; fácil de ser governado, mas impossível de ser escravizado." - Henry Peter

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Esta seção introduz este trabalho abordando a sua respectiva motivação, problema e objetivos com a sua pesquisa e desenvolvimento.

1.1 MOTIVAÇÃO

Alunos do Ensino Fundamental, independente da classe social, possuem grandes dificuldades no aprendizado de Matemática, sendo a Álgebra um expoente entre tais assuntos (STACEY e MACGREGOR, 2000). Estudos mostram, através de (MINISTÉRIO DA EDUCAÇÃO/SECRETARIA DO ENSINO FUNDAMENTAL, 1998), que os alunos brasileiros têm grande dificuldade em assuntos relacionados à Álgebra. Nos resultados do Sistema Nacional de Avaliação da Educação Básica (SAEB), por exemplo, itens referentes à Álgebra raramente atingem um índice de 40% de acerto em muitas regiões do país.

Inicialmente nos estudos matemáticos os alunos lidam com uma série de cálculos aritméticos, tendo como desafio realizar uma cadeia sucessiva de operações, com a devida ordem de resolução. Em um segundo momento, na Álgebra, é preciso fazer relações e simplificar termos, a fim de encontrar valores que solucionem uma condição proposta em uma equação. É principalmente neste último que ocorrem interpretações errôneas sobre como lidar com problemas algébricos (STACEY e MACGREGOR, 2000).

Benjamim Bloom (BLOOM, 1984) mostrou que no ensino particular, em que um professor cuida exclusivamente de um único aluno, os resultados são bem mais significativos do que na abordagem tradicional - um professor para um conjunto de alunos. Porém, tal abordagem é considerada cara, e neste cenário um STI pode oferecer uma alternativa viável de ensino individualizado (MURRAY, 1999).

É neste cenário que a computação oferece uma alternativa, com uso de Sistemas Tutores Inteligentes. Tais sistemas computacionais oferecem um ambiente virtual em que os alunos podem interagir, sendo acompanhados com o uso de agentes computacionais, por exemplo. Estes agentes podem ser capazes de inferir o grau de compreensão dos estudantes em relação aos conteúdos propostos dentro do sistema, bem como que tipo de ajuda individual é necessária para que determinados erros de aprendizado não ocorram novamente.

No entanto a sala de aula convencional possui uma metodologia de ensino que pode diferir do ensino em um ambiente virtual, causando conflitos de aprendizado por parte dos estudantes. Isto exige um desenvolvimento detalhado do modo como é realizado o ensino dentro do sistema, sendo considerada a tarefa mais importante na construção de um Sistema Tutor Inteligente (ANDERSON, CORBETT, *et al.*, 1992).

Os Sistemas Tutores Inteligentes - STI (ou *Intelligent Tutoring Systems – ITS)*, surgiram como uma tentativa de realizar o ensino através de meios computacionais, pois são sistemas computadorizados inteligentes capazes de ensinar determinados conteúdos, possuindo representações do conhecimento do domínio a ser ensinado bem como um modelo dos saberes do estudante e suas respectivas habilidades. Tais sistemas possuem em sua arquitetura diferentes Modelos Instrucionais representando abordagens diferenciadas de ensino (MURRAY, 1999). Tal ensino individualizado fornece benefícios semelhantes aos do um professor particular, considerado o melhor caso para aprendizado do aluno (BLOOM, 1956; BLOOM, 1984).

A arquitetura básica de um STI compreende quatro componentes: modelo de aluno (responsável por armazenar informações do aluno), módulo tutor (encarregado de avaliar as informações do aluno e ensinar de acordo com suas respectivas necessidades), módulo especialista (módulo cujo objetivo é saber resolver todos problemas do domínio de maneira ideal) e modelo de domínio (modelo que contém todas as informações a serem expostas para o aluno, os conteúdos propostos). Esta arquitetura é retratada em trabalhos como os de (BECK, STERN e HAUGSJAA, 1996; ANDERSON, CORBETT, *et al.*, 1992).

Dado este contexto, o papel de um sistema computacional seria ajudar aluno no ensino. Para tanto, deve oferecer um ambiente visual interessante, através de uma linguagem acessível que desperte o seu interesse em aprender e considere seus aspectos de aprendizado, suas características cognitivas, em conjunto com uma interface visualmente simples, operacional e motivadora.

Para tais características se tornarem realidade, o sistema como um todo deve conter os conhecimentos do domínio em questão, além de saber orientar o aluno em eventuais erros, constituindo um caráter pedagógico e didático. Estes conhecimentos comentados anteriormente remetem ao domínio abordado pelo sistema, com suas explanações e exemplos, bem como seus respectivos exercícios, com suas diferentes maneiras de resolução, ideais ou não.

As informações sobre o aluno em questão, que utiliza o sistema, ficam mapeadas no seu respectivo modelo de aluno. Estas são cruciais para que o Módulo Tutor constitua um

ensino individualizado e focado, ao considerar tais variáveis sobre cada aluno que venha a interagir com o sistema. Neste modelo podem ser mantidas informações referentes a exercícios realizados, conteúdos vistos ou exemplos acompanhados, dados que permitem que o Agente Tutor tomar de forma aprimorada decisões sobre que materiais apresentar ao aluno.

Porém, o estudo sobre como acontece o processo de ensino por vezes foi julgado como um fator de pura experiência do professor, sucessivas tentativas e erro, cultura ou mera intuição do docente (GAUTHIER, 1998), o que torna complexa a modelagem computacional dos fatores deste estudo como um todo. No campo da Matemática, existem ainda variadas maneiras de se resolver um problema, umas destas mais simplificadas ou exatas que outras, o que torna difícil saber o quão certo um aluno está e o quanto ele domina um conteúdo.

1.2 PROBLEMA

O ensino de um determinado conteúdo com o auxílio de recursos computacionais impõe um desafio a qualquer desenvolvedor, pois existem variáveis de difícil modelagem computacional, como os dados subjetivos sobre as impressões do aluno acerca de um conteúdo sendo trabalhado por exemplo. Além disso, existem grupos distintos de usuários com suas respectivas características e limitações cognitivas. Generalizar o ensino tendo em vista tais fatores é um erro presunçoso, em que se acredita que as pessoas aprendem da mesma maneira, no mesmo ritmo. Isto pode ser remediado com o acompanhamento individual do aluno, levando em consideração as suas características cognitivas.

No cenário matemático, mais precisamente no ensino-aprendizagem de Álgebra, essa característica inteligente se concretiza com um sistema que possa resolver cada passo de simplificação das equações em um problema algébrico, mostrando ao aluno se estava correto ou não o que foi realizado até o momento. O problema consiste em saber exatamente o quanto o aluno domina de um determinado conteúdo, assim como seus pré-requisitos, e conhecer qual a maneira mais adequada de ensinar um conteúdo novo. No caso de eventuais erros que o aluno cometer, inferir em que parte do aprendizado aconteceu e o que levou este aluno a errar, assim como decidir qual ação seria a mais apropriada para auxiliar o aluno em sua aprendizagem. Esta questão foi trabalhada com as características propostas posteriormente do Modelo de Aluno e Módulo Tutor.

Dessa forma, o desafio deste trabalho encontra-se em como representar os conhecimentos de um determinado aluno dentro do sistema (Modelo de Aluno) e os meta-conhecimentos do Módulo Tutor sobre o domínio abordado, como estratégias pedagógicas e

seu plano de ensino. Estes elementos são necessários para a criação de estratégias pedagógicas, pois a sua aplicação depende do que o aluno sabe e de como pode ocorrer o ensino no sistema. Tais estratégias ficam a disposição do Módulo Tutor, para que sejam utilizadas de maneira a concretizar um aprendizado voltado às características cognitivas do aluno.

1.3 OBJETIVO

O objetivo do presente trabalho consiste em modelar e desenvolver o Módulo Tutor e Módulo de Modelo de Aluno do Sistema Tutor Inteligente PAT2Math, para que o mesmo consiga ensinar de forma eficiente Álgebra Elementar a alunos do Ensino Fundamental. O Módulo Tutor deverá criar planos de ensino específicos para um aluno, levando em consideração as características cognitivas mapeadas no seu respectivo Modelo de Aluno, sendo responsável por aplicar estratégias pedagógicas, seguir um plano de ensino padronizado e customizar este para o aluno em caso de erro, constituindo o processo de ensino do sistema.

O Agente Modelo de Aluno possuirá informações do perfil e desempenho do aluno em questão, registradas no seu respectivo Modelo de Aluno, para servir de apoio ao Agente Tutor na aplicação de uma estratégia pedagógica adequada para o seu ensino individualizado bem como manter registro do seu progresso e suas interações com o sistema.

1.4 ORGANIZAÇÃO DO TRABALHO

O trabalho está organizado como segue:

Na seção dois, Fundamentação Teórica, estão os estudos sobre como propor meios de utilização do computador e sistemas inteligentes no ensino, além de como ocorre o ensino tradicional para então levar isto a tais sistemas, articulando com uma tecnologia de construção. A teoria envolvendo o processo de Avaliação Qualitativa também está presente, em conjunto com ferramentas e metodologias para avaliação sob este caráter de softwares educacionais.

A seção três aborda Teorias de Aprendizagem, como a Aprendizagem Significativa de Ausubel, Taxonomia de Bloom, Teorias de Design Instrucional de Merril e a estratégia pedagógica *Macro-Adaption.* Com este conhecimento consolidado pode-se avaliar

criticamente os trabalhos que estão oferecendo uma alternativa ao ensino de Álgebra, para a avaliação dos seus pontos de destaque e limitações.

A seção quatro mostra alguns ambientes de aprendizagem concebidos com abordagem multiagente, conceituando inicialmente o que são agentes, sistemas multiagentes e uma metodologia para tal.

A seção cinco aborda as questões da Web Semântica e Ontologias, conceituando detalhes de definições desta estrutura, como o RDF e OWL com suas respectivas sub-categorias.

A seção seis contém a análise dos Trabalhos Relacionados, abordando pontos interessantes da construção de sistemas multiagentes para ensino.

A seção sete aborda o Projeto PAT2Math, um projeto de pesquisa que visa auxiliar o ensino de Álgebra através de um ambiente Web. O desenvolvimento deste envolve pesquisadores da computação e matemática e vem a ser o projeto no qual o Trabalho Proposto, na seção oito, se inclui.

A seção nove contém a avaliação qualitativa realizada com docentes e pesquisadores para descobrir a o fator de qualidade do projeto, segundo um questionário adaptado de uma dissertação de Mestrado cujo objetivo era elencar métodos e propor uma alternativa de análise de softwares educacionais.

Por fim, na seção dez, estão as considerações finais deste projeto.

2. SISTEMAS TUTORES INTELIGENTES

Os computadores têm sido usados para alcançar uma série de objetivos educacionais, como testes exaustivos de simulações científicas ou ainda tarefas práticas que eram mecanizadas com tecnologia obsoleta, antes mesmo da década de 60. Essa área era denominada IAC (Instrução Assistida por Computador), e ingressou nos mercados de educação e treinamento oferecendo uma alternativa para ensino em geral, mesmo estando afastada do modelo de ensino convencional Instrucionista e com o uso de quadro-negro (CORBETT, KOEDINGER e ANDERSON, 1997).

No início da década de 1970 pesquisadores definiram um objetivo ambicioso para o já tradicional IAC – adotar um tutor humano como modelo educacional de tais sistemas, aplicando técnicas de Inteligência Artificial (IA) para que o mesmo tenha uma característica inteligente. Assim criou-se o conceito de Sistemas Tutores Inteligentes (STI), cujo objetivo era engajar os estudantes em atividades visando o ensino de determinados conteúdos, considerando o seu comportamento individual. Na época acreditou-se que se tais sistemas conseguissem metade do impacto que tutores reais causam em grupos de alunos, a vantagem para a sociedade já seria interessante, o que alavancou estudos posteriores nesta área (CORBETT, KOEDINGER e ANDERSON, 1997).

Sistemas Tutores Inteligentes (ou *Intelligent Tutoring Systems – ITS)* são sistemas computadorizados capazes de ensinar conteúdos, através de Modelos Instrucionais que estejam presentes em sua arquitetura. Tais modelos representam abordagens diferenciadas de ensino, levando em conta o caráter heterogêneo dos perfis de aluno que possam vir a utilizar o sistema, além das capacidades cognitivas de cada um (MURRAY, 1999).

Um STI possibilita ao aluno resolver problemas propostos, passo a passo, para a validação do aprendizado. Estes sistemas estão preparados para oferecer ajuda em qualquer momento necessário, portanto, a mostrar como solucionar os problemas a partir de qualquer ponto da resolução. É essa característica que os fazem serem considerados ‘Inteligentes’, poder resolver problemas que eles mesmos apresentam aos alunos (CLANCEY, 1986).

Uma característica marcante destes STIs é a capacidade de adaptar dinamicamente o conteúdo a ser exibido ou a abordagem pedagógica de ensino. Estes fatores contribuem para que ocorra um ensino individualizado para cada aluno, considerando suas características cognitivas, assemelhando isto ao ensino de um professor particular para tal (MURRAY, 1999). Estudos feitos por Benjamim Bloom (1984) mostram que no ensino onde um professor

cuida exclusivamente de um único aluno os resultados são bem mais significativos do que na abordagem tradicional - um professor para um conjunto de alunos. No entanto, essa abordagem proposta é cara. Usar um *Sistema Tutor Inteligente* (ou *Intelligent Tutor System – ITS)* é considerada uma alternativa mais viável (ANDERSON, BOYLE, *et al.*, 1990; MURRAY, 1999).

O receio de profissionais do ensino com a utilização destes sistemas se encontra na linguagem utilizada na explanação dos conteúdos, se é condizente com a usada por eles em sala de aula. Tal conceito é de difícil consenso entre diferentes grupos de professores e pesquisadores. O trabalho proposto por (ANDERSON, CORBETT, *et al.*, 1992) sofreu fortes críticas devido à diferença de linguagens utilizadas para o ensino no seu respectivo STI, comparando-se com o ensino tradicional do conteúdo abordado.

2.1 ARQUITETURA PADRÃO DE UM SISTEMA TUTOR

Ao longo da pesquisa na área de STIs, surgiram certas definições de arquitetura para organizar o conhecimento e as ferramentas do sistema. Os modelos mais comuns compreendem um Modelo de Estudante, Módulo Pedagógico, Modelo de Domínio e Modelo Especialista, abordados a seguir (BECK, STERN e HAUGSJAA, 1996; ANDERSON, CORBETT, *et al.*, 1992)

Modelo de Estudante

Sistemas Tutores Inteligentes inferem um modelo da compreensão atual do estudante sobre o conteúdo abordado e usam isto para adaptar o ensino levando em conta as habilidades cognitivas do aluno. Esta estrutura que descreve o estado atual do estudante em relação aos conteúdos abordados em um STI é chamada de Modelo de Estudante ou Modelo de Aluno (POLSON e RICHARDSON, 1988).

O Modelo de Estudante armazena as informações que são específicas de cada estudante, suas habilidades e conhecimentos, bem como qualquer informação que deva alterar o meio como ele deva ser ensinado. Isto é usado na apresentação de conteúdos, que deve ser adaptável ao aluno, assim como a representação do seu processo de aprendizado. Também

visa aumentar a quantidade de informação sobre o estudante, identificando os tipos de conhecimento que devem ser aprendidos por ele (POLSON e RICHARDSON, 1988).

Este modelo monitora o quão bem um estudante está interagindo com os materiais de ensino, bem como as falsas concepções que ele possa ter sobre os mesmos. Representar *misconceptions,* tratados como conhecimentos incorretos que o estudante possui sobre um conteúdo, representa um novo nível de complexidade na modelagem do estudante em um STI. Tipicamente isto é realizado adicionando-se uma *bug library*, uma biblioteca de erros, coletada de forma empírica. Tal trabalho empírico pode ser reduzido fazendo uma compilação destas inconsistências baseada em regras, no conhecimento procedural abordado no sistema (POLSON e RICHARDSON, 1988).

Todas as informações coletadas no sistema sobre o aluno devem estar disponíveis para que o sistema possa fazer a criação de um plano de ensino personalizado e adequado ao aluno em questão, considerando suas respectivas características, maximizando as chances de aprendizado. Considera-se importante que informações mais gerais, de caráter pedagógico, também sejam incluídas neste modelo, como os estilos cognitivos de aprendizagem. Isto inclui informações sobre a preferência do aluno em olhar exemplos antes de responder perguntas, ou se isto ocorre somente em caso de erro, por exemplo. Com isto, o modelo deve avaliar cada uma das ações realizadas, enquadrando estados que representem os estágios do aprendizado do aluno (CLANCEY, 1986; HOLT, DUBS, *et al.*, 1994).

Um dos meios mais comuns de representação do Modelo de Estudante é com o método *overlay Figura 1 à esquerda*, em que o conjunto de saberes do estudante é um subconjunto dos saberes do Modelo Especialista, que é o módulo capaz de resolver problemas da maneira considerada ideal. Com esta representação, o STI apresenta material com o objetivo de que o conhecimento do aluno chegue a ser exatamente igual ao do Especialista. Este modelo *overlay* pode comportar conhecimento incerto do estudante sobre determinados conteúdos, constituindo um layout de overlay estendido, como está representado na Figura 1: Layout Overlay de Modelo de Estudante (à esquerda) e Layout Overlay Estendido (à direita), traduzido de (Clancey, 1986)

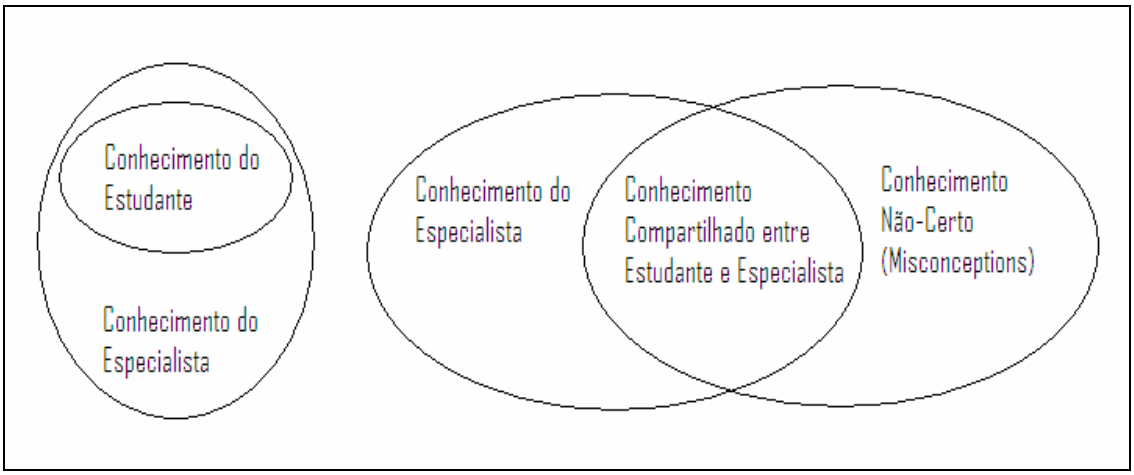

Figura 1: Layout Overlay de Modelo de Estudante (à esquerda) e Layout Overlay Estendido (à direita), traduzido de (Clancey, 1986)

Além do Modelo de *Overlay*, existe uma variedade de definições sobre modelos de estudante, conforme segue:

- *Modelo de Estudante Diferencial*: O conhecimento do aluno é dividido em duas categorias: conhecimento que o estudante deve possuir e conhecimento que não espera-se que o mesmo saiba (HOLT, DUBS, *et al.*, 1994);
- *Modelo de Estudante baseado em Perturbação e Biblioteca de Erros:* Enquanto o Modelo de *Overlay* é representado apenas em termos de conhecimento correto, um Modelo de Perturbação normalmente combina o básico *overlay* com uma representação de conhecimentos incorretos do aluno sobre o domínio abordado (HOLT, DUBS, *et al.*, 1994);
- *Modelo de Estudante PairSM:* trata-se de um abordagem que contém dois modelos de estudante, que são comparados pelo sistema para permitir que o mesmo sugira maneiras para que dois alunos trabalhem juntos de forma eficiente. Encoraja trabalho em dupla e suas interações como algo benéfico no aprendizado (SUSAN e SMITH, 1997).
- *Modelo de Estudante baseado em Restrições:* Modelo que representa o conhecimento do aluno como restrições sobre o conhecimento tido como correto em um dado domínio (HOLT, DUBS, *et al.*, 1994).

- *Modelo de Estudante PeerISM:* Modelo de estudante que visa promover interação em duplas ao permitir que os alunos construam um modelo próprio de domínio para exploração (BULL e BRNA, 1999).

A tendência no desenvolvimento de um Modelo de Estudante dentro de um Sistema Tutor Inteligente é a de incluir mais informação do que o Módulo Tutor poderá usar. O propósito da construção deste modelo é poder construir lições específicas para cada estudante. Sob este ponto de vista, este modelo deve conter somente as informações que o Módulo Tutor irá usar, no entanto, para fins de pesquisa, pode ser interessante incluir elementos adicionais acerca do aluno, com o fim de proporcionar aos projetistas conhecimentos que podem ser úteis na modelagem de futuros modelos de estudante.

Eventos em geral, tais como terminar um exercício ou ir para outra página dentro do sistema, devem ativar mecanismos que atualizam o modelo do estudante. Uma alternativa sobre como o conhecimento deve ser dividido, foi definido em (SIEKMANN e MELIS, 2004) – *conhecimento geral*, *compreensão* e *aplicação.* Dependendo do que o aluno realizar, um determinado parâmetro dos três últimos citados é atualizado. Exemplificando - Se o aluno leu um determinado capítulo, *conhecimento geral* é atualizado; acompanhar exemplos atualiza *compreensão;* fazer alguns exercícios atualiza *aplicação.*

Estas saídas do Modelo de Estudante podem ser utilizadas para vários fins dentro de um STI, como por exemplo: avançar um conteúdo, dar uma sugestão, gerar um problema a ser resolvido ou adaptar os conjuntos de explicações a serem mostrados ao estudante (POLSON e RICHARDSON, 1988).

Estes conhecimentos acerca das *tarefas* realizadas permitem ao sistema mapear os saberes do aluno, além de fazer estatísticas e probabilidades sobre o domínio do aluno nos assuntos abordados. Explicitar tais valores para o estudante e professor é considerado importante, permitindo a análise da sua curva de aprendizado, com o objetivo de avançar nas lições somente quando existir um domínio satisfatório de determinados assuntos abordados. Por exemplo, considerando que um determinado aluno realizou corretamente uma série de exercícios dentro de um STI, cuja resolução exige a utilização das regras A e B. Tais regras remetem ao conteúdo C, implicando que o aluno possui domínio sobre este.

Módulo Tutor

O Módulo Tutor fornece um modelo de processo de ensino, que utiliza informações contidas no Modelo de Estudante, para a customização de planos de ensino, que são estruturas que definem a ordem de conteúdos e de seus respectivos exercícios a serem mostrados a um aluno durante a sua exploração de currículo em um STI. Nesta customização são incluídas estratégias pedagógicas, representando diferentes maneiras de ensino para os conteúdos do sistema. É este módulo que habilita um STI a ensinar um determinado aluno, baseando-se em estratégias adequadas a cada situação. Esta é composta por dois elementos – informações sobre o aluno e o assunto sendo tratado no sistema. Por exemplo, avaliando um cenário hipotético em que o aluno é considerado iniciante em um assunto, o sistema poderia mostrar procedimentos passo a passo e explicações mais detalhadas dos conceitos básicos em um primeiro momento. Posteriormente o sistema permitiria ao estudante explorar simulações e exercícios e só oferecer ajuda quando requisitado.

Um Sistema Tutor Inteligente deve possuir no seu Módulo Tutor três características:

- Controle sobre a representação do conhecimento instrucional do sistema, a ponto de poder selecionar e seqüenciar de maneiras diferentes um conteúdo;
- Capacidade de responder questões de um aluno sobre um conteúdo, ou objetivos deste;
- Estratégias pedagógicas para determinar se um aluno precisa de auxílio e meios para fornecer uma ajuda personalizada, que considere as suas características cognitivas.

As características pedagógicas de um Sistema Tutor Inteligente podem seguir teorias clássicas de ensino, utilizadas em conjunto com recursos sofisticados de análise do Modelo de Estudante, para a seleção de Objetos de Aprendizagem adequados ao ensino idealizado em tais teorias. Avaliação de dados reportados pelos alunos e administradores do sistema sobre estes, no caso Professores, também constituem parte do caráter pedagógico em um STI.

No Sistema Tutor Inteligente ActiveMath (SIEKMANN e MELIS, 2004), estas informações pedagógicas utilizam Taxonomia de Bloom (1956) para criar uma estrutura que representa o quão apto um aluno está em um dado conteúdo, assim podendo criar um cenário para este ensino, com objetos de aprendizagem que respeitam a Teoria de Design Instrucional de Merril (MERRIL, 2001).

Já no Sistema Tutor Inteligente AnimalWatch (COHEN, BEAL e ADAMS, 2008) as informações pedagógicas são construídas a partir de auto-avaliações de alunos, além de

informações inseridas pelos professores no sistema a respeito dos mesmos. Em conjunto com isto, as ações dos estudantes são enquadradas em máquinas de estado representando as interações possíveis dentro do sistema.

Os cenários de ensino em um STI são categorizados pela utilização do conhecimento, seguindo definições como as descritas por (CLANCEY, 1986):

- *Abordagem Socrática:* Nesta abordagem o tutor faz perguntas sucessivas e força o estudante a perceber inconsistências no seu aprendizado. Este método se destaca na resolução de problemas que é fortemente baseada em experiências prévias, tais como Direito e Medicina.
- *Ambiente Reativo:* Com esta abordagem o estudante resolve problemas sem a intervenção direta do sistema, este reage com ações-chave do estudante e fornece *feedback* adequado.
- *Aprendizado por ação:* Neste método o Tutor contribui ativamente no aprendizado do estudante, verificando constantemente o progresso e compreensão. Casos são acompanhados pelo Tutor e estudantes juntamente.

Modelo de Domínio

O Modelo de Domínio contém as informações que o Módulo Tutor é encarregado de ensinar ao aluno, sendo considerado o mais importante dos módulos, pois sem este, não existiria o que ensinar no sistema. Geralmente requer uma boa engenharia de conhecimento para representar os conteúdos e ao mesmo tempo ser totalmente acessível ao Módulo Tutor do sistema. Comumente utilizado como base de conhecimentos teóricos e exercícios a serem propostos, a disposição do sistema para ofertar ao aluno. Diferencia-se do Modelo Especialista por possuir todos os conhecimentos em sua estrutura, caracteristicamente descrito de forma declarativa, contrapondo-se ao caráter procedural de poder resolver problemas do domínio de maneira ideal, correta (MURRAY, 1999).

Modelo Especialista

A maior lição que a comunidade pesquisadora de Inteligência Artificial aprendeu com Sistemas Especialistas é que qualquer Modelo Especialista deve conter muito conhecimento específico e detalhado, proveniente de pessoas que têm anos de experiência no domínio abordado (POLSON e RICHARDSON, 1988).

O Módulo Especialista deve conter todas as informações para resolução de problemas do domínio que está sendo ensinado a um determinado aluno, com o objetivo de ser um modelo que represente um especialista da área de conhecimento abordada, sendo fortemente utilizado na resolução e validação de exercícios propostos de determinado domínio. Diferencia-se do Modelo de Domínio por possuir conhecimentos procedurais, necessários para resolver problemas. Como o Módulo Especialista é capaz de resolver problemas, também pode verificar a resolução de exercícios realizados pelo aluno, analisando se o que o aluno realizou está correto ou não. Com estas informações o Módulo Tutor pode inferir em que pontos o aluno está tendo certa dificuldade, oferecendo ajuda adequada (MURRAY, 1999).

2.2 MODELAGEM DO COMPONENTE TUTOR DE UM SISTEMA TUTOR INTELIGENTE (STI)

Com o objetivo de interpretar o comportamento do estudante dentro de um sistema computacional, é interessante mapear os seus respectivos saberes sobre os assuntos abordados dentro deste mesmo sistema. Uma possível abordagem para esse problema foi proposta pela teoria ACT - (*Atomic Components of Thought*) (ANDERSON, CORBETT, *et al.*, 1992) que afirma que a habilidade cognitiva de um estudante pode ser representada como um conjunto de regras, interpretadas como estruturas de *condição-ação.* Tais regras podem ser utilizadas para interpretar as resoluções de exercícios realizadas pelo estudante, enquadrando as suas ações com estas regras previamente definidas pela teoria ACT. Isto viabiliza o acompanhamento das ações individuais de um aluno através do *Model Tracing* (ANDERSON, CORBETT, *et al.*, 1992) para que se possa inferir os seus respectivos conhecimentos através do *Knowledge Tracing* (ANDERSON, CORBETT, *et al.*, 1992; CORBETT e ANDERSON, 1995). Estas teorias amadureceram no cenário de pesquisa e demais pesquisadores definiram o que é chamado de *Outer Loop* e *Inner Loop*, (VANLEHN, 2006), podendo ser claramente comparados com o *Knowledge Tracing* e *Model Tracing*, respectivamente.

Um estudante que utilize um STI, segundo estas teorias, deve receber do sistema o que são chamados de *tasks (tarefas) e steps (passos)*. *Tasks* compõem uma lista de tarefas para aprender um conteúdo novo, constituindo o *Outer Loop*. Cada uma destas tarefas é selecionada segundo o nível de habilidade do aluno, seus saberes acerca do que está sendo ensinado. Para obter sucesso e concluir cada uma das *tarefas*, seja realizar um exercício ou ler um determinado material, é repassada ao estudante uma série de *passos*, pequenas atividades ou ações na interface do sistema para que isso aconteça, dando forma ao *Inner Loop* (VANLEHN, 2006). No caso do PAT2Math, um aluno que esteja aprendendo o conteúdo de "Equações" teria que ler um texto introdutório sobre equações (tarefa) e realizar a resolução de um exercício completo, aplicando operações (passos) em uma equação até resolvê-la.

Outer Loop

Este é o ponto de partida do ensino em um STI. Aqui existem certas políticas para a criação das *tarefas* necessárias para que um aluno aprenda um dado conteúdo. Entre elas tem-se:

Sequência fixa de tarefas: O padrão menos flexível de todos. Assume-se que concluindo um número X de atividades há certeza de que o aluno dominou um conteúdo. Por exemplo: Ler um texto introdutório, acompanhar dois exercícios resolvidos e resolver três exercícios por conta própria.

Mastery Learning de Bloom (BLOOM, 1984)*:* Nesta política há a questão de *mastery*, em que se trabalham objetos em dadas dificuldades até que não exista erro algum, passando a um próximo item somente após isto.

Macro-Adaption (CORBETT e ANDERSON, 1995; SHUTE, 1993)*:* A política mais adaptável ao perfil do aluno, no entanto a de maior complexidade de implementação. Em cada *tarefa* sabem-se quais conhecimentos ou habilidades são trabalhadas, sendo selecionadas as tarefas que envolvem as menos desenvolvidas. Adicionalmente, este modelo pode ser estendido para selecionar tarefas que trabalhem *misconceptions*, falsas concepções a respeito de um conteúdo. Esta técnica será mais detalhada na seção '3.3 Macro-Adaption'.

Modo Tarefa: Nesta abordagem se mostram os passos de resolução de problemas, explicando-os. Então se deixa o estudante fazer alguns exercícios dando dicas em cada passo. Após isso se exercita a autonomia do aluno, deixando-o sem o auxílio das intervenções automáticas do sistema, mostrando ajuda apenas ao final ou no primeiro erro. Isso constitui uma redução gradual das informações, nomeada "*fading the scaffoling"*.

Inner Loop

Neste segundo momento da interação do aluno com o STI são apresentados os *passos* para se concluir as *tarefas* designadas como necessárias para aprendizado do conteúdo. Aqui o aluno pode ser solicitado a ler um material teórico, observar alguns gráficos ou ainda resolver exercícios, passo-a-passo, de forma autônoma. Na 'Figura 2: Exemplo de Resolução de Equação' tem-se um exemplo de passos fornecidos pelo aluno para realizar uma tarefa, no caso a resolução de uma equação algébrica. Nesse exemplo, um passo seria a aplicação de uma ou mais operações para cada solução parcial da equação, como pode ser observado na Figura 2 (passos 1 a 4).

2x + 3x – 25 = 0; (Equação fornecida pelo Tutor)
2x +3x = 25; (passo 1)
5x = 25; (passo 2)
X = 25/5; (passo 3)
X = 5; (passo 4)

Figura 2: Exemplo de Resolução de Equação

Neste cenário podem ser aplicadas estratégias para a maneira com que se realizam estes passos, sendo as principais a *guia* e a *assistente*. Na primeira o STI mostra passo a passo a resolução de problemas, explicitando as relações teóricas com o material visto anteriormente, não há espaço para o aluno errar. Já na categoria *assistente* o STI só entra em cena no momento em que o estudante comete um erro, ou solicita explicitamente ajuda para realizar um passo, possuindo um caráter bem menos invasivo.

Para a efetivação destas estratégias é necessário informar ao estudante o resultado das suas ações, proporcionar um *feedback* ao realizar essa série de *passos*. E este recurso é dividido em três categorias:

Imediato: Cada ação realizada é avaliada pelo sistema instantaneamente, mostrando o seu sucesso ou falha;

Delayed: Após concluir um determinado número de passos;

Sob demanda: Assim que o estudante solicitar ajuda.

2.3 MODELAGEM DO MODELO DE ALUNO EM UM STI

As idéias envolvidas na criação de um Módulo Tutor são centradas na produção de regras e exposição de exemplos aos estudantes, segundo o *Model Tracing* e *Knowledge Tracing.*

O *Knowledge Tracing* consiste em mapear os conteúdos que o aluno possui domínio através da avaliação das habilidades que o mesmo utilize na resolução de problemas do domínio. Conteúdos possuem uma série de habilidades que são exercitadas através dos seus respectivos exercícios, cenário em que o monitoramento da aplicação destas habilidades pelo aluno pode dizer o quão ele domina do conteúdo em questão. Nesta técnica podem ser aplicadas as políticas comentadas previamente na seção *"Outer Loop"*.

O *Model-Tracing* consiste em tentar encontrar uma seqüência destas regras que reproduza o comportamento do aluno dentro do sistema ao resolver problemas. O objetivo desta técnica dentro de um STI é saber como oferecer ajuda adequada àquela linha de ações que o estudante tomou, pois está se comparando com um modelo ideal de sequência de regras que resolvem um determinado problema proposto. Se alguma etapa prevista pelo sistema não foi identificada nas ações realizadas pelo aluno, isso se torna prioridade no auxílio a ser oferecido ao mesmo. Um exemplo de regra que se enquadre neste modelo seria como a mostrada na Figura 3:

SE: Objetivo é simplificar o uma fração
ENTÃO: Encontrar um divisor comum

Figura 3: Exemplo de Regra de Produção sob aspecto Model-Tracing

Esta técnica pode ser utilizada com o objetivo de proporcionar *feedback* imediato das ações de um aluno ao resolver um dado problema, atuando como interpretadora de pensamentos em cada passo da resolução (POLSON e RICHARDSON, 1988).

Para mapear os conhecimentos do aluno em cada momento da resolução de um problema, o Model Tracing pode criar um espaço de dados consideravelmente complexo computacionalmente. Analisando cada ponto do problema, existe um número de regras que são enquadradas como corretas ou não à continuação da resolução. A combinação destas camadas de regras cria este espaço complexo comentado anteriormente, gerando ambigüidades na interpretação do comportamento do estudante, fato que o Módulo Tutor deve considerar e resolver antes de dar o devido *feedback* (POLSON e RICHARDSON, 1988).

A variável que determina a taxa de aprendizado é o número de problemas que foram solucionados e compreendidos pelo aluno, além dos três critérios a seguir:

- O estudante precisa entender o conteúdo, não simplesmente seguir passos cegamente para resolver um determinado tipo de problema;
- Os problemas devem exercitar os conhecimentos adquiridos, devem envolver o foco do estudo;
- A velocidade de apresentação dos materiais deve ser avaliada; o caso de um ambiente de pura exploração de saberes seria desencorajado, pois há conteúdos de complexa compreensão podendo ser explorados sem a orientação adequada.

Um cenário a ser considerado é o seguinte – o aluno não consegue acertar a questão e recusa ajuda do sistema. Dois motivos podem ser a causa disto, sendo o primeiro o aluno não compreender o que uma determinada regra propõe ser feito e o segundo o aluno realmente acreditar que está certo o que ele realizou, ignorando a ajuda oferecida pelo sistema.

2.3 CATEGORIAS DE SISTEMAS TUTORES

Com uma ampla variedade de sistemas tutores, determinadas categorias foram criadas ao longo da história de tais sistemas. Em geral, há oito gêneros diferentes, que serão caracterizados a seguir: *Curriculum Sequence, Tutoring Systems, Device Simulation, Expert Systems, Cognitve Tutors, Multiple Knowledge Types, Special Purpose Systems* e *Intelligent/Adaptative/Hypermedia* (MURRAY, 1999). O Sistema Tutor Inteligente PAT2Math, projeto no qual será proposta a contribuição deste trabalho, possui um caráter híbrido nesta classificação, situando-se em *Tutoring Systems* e *Expert Systems*. Esta característica se dá pelo fator diferenciado e localizado de ajuda oferecido ao aluno, bem como conhecimento global de resolução dos problemas do domínio, no caso a Álgebra e suas equações.

Curriculum Sequence: É o mais básico, considerado o mais Instrucionista de todos. Neste modo os conhecimentos estão dispostos hierarquicamente, tendo pré-requisitos para serem

alcançados. Trata-se de uma árvore de conteúdos, geralmente tendo textos bem descritivos, muitas imagens e pouca interatividade.

Sistemas Tutores (*Tutoring Systems*): Acrescentam um importante elemento ao modelo anterior, se preocupando com detalhes menores, tais como quando e como mostrar certas informações. Isto acaba definindo certos níveis para a ajuda fornecida, podendo ser bem regulada. É bastante ativo ao oferecer auxílio em diversos pontos, tirando um pouco do foco Instrucionista do modelo anteriormente citado.

Device Simulation: O sistema considera que o aluno tenha um conhecimento prévio de elementos básicos ao assunto, fornecendo ajuda específica e localizada. O aluno avança em passos graduais ao executar uma tarefa, montando um motor, por exemplo, peça a peça. Trata-se de um aprendizado bem exercitado, onde o aluno aprende construindo exemplos sugeridos pelo sistema.

Expert Systems* e *Cognitive Tutors: São sistemas em que um agente tem conhecimento global do assunto em questão e são mantidas informações para executar tarefas por inteiro, caso o aluno necessite de visualização explícita. Requer uma modelagem extensa e trabalhosa, pois o sistema precisa estar apto a ajudar o aluno em qualquer ponto de qualquer tarefa, refletindo um modelo cognitivo bem desenvolvido. Em geral, tarefas específicas e assuntos bastante técnicos são abordados com um sistema deste.

Multiple Knowledge Types: Tais sistemas abordam os seus conteúdos com práticas repetitivas em geral. No caso da explicação de conceitos são usadas analogias, além de exemplos corretos e errados sobre cada ponto importante, denotando explicitamente o que deve ou não ser feito. A característica marcante desse tipo de sistema é que as habilidades complexas são decompostas em pequenos componentes, no caso o conhecimento é desmembrado e colocado de forma linear, auxiliando a compreensão do aluno.

Special Purpose Systems: Esta categoria de sistema possui foco dos assuntos mais específico ainda, constituindo tarefas especificamente técnicas e que envolvem conhecimentos prévios bem desenvolvidos sobre o que é abordado. O problema em potencial é o público bastante restrito deste tipo de sistema.

Intelligent/Adaptive/Hypermedia (Intelligent Learning Enviroments): Estes sistemas são bastante sofisticados, abordando métodos e modelos recentes da pesquisa em Sistemas Tutores Inteligentes. Muitos destes estão na *web*, verificando a integridade dos *links* e conteúdos a serem exibidos. Suas variáveis de decisão são o perfil do usuário, além do esquema de pré-requisitos e árvore de conhecimentos, abordados no *Curriculum Sequence.*

2.4 CRIAÇÃO DE SISTEMAS TUTORES INTELIGENTES

Segundo os estudos propostos por (ANDERSON, CORBETT, *et al.*, 1992), existe uma série de fatores que devem ser considerados no desenvolvimento de um Sistema Tutor Inteligente, conforme segue.

Desenvolver uma Interface Estruturada: A primeira etapa no desenvolvimento da interface do sistema é projetá-la estruturadamente para a resolução de problemas. É necessário construir e testar as capacidades de resolução do sistema antes do Módulo Tutor propriamente dito. O objetivo se concretiza em maximizar a eficiência do ambiente solucionador de problemas.

Especificar Sintaxe de Solução: É preciso que o sistema esteja habilitado a perceber o estado atual da solução e modificá-lo assim como o estudante pode. A manipulação do conhecimento deve requisitar uma sintaxe que descreva como está o ambiente de aprendizado no momento em questão.

Definir Regras de Produção e Sintaxe de Resolução de Problema: Manter uma representação do problema em questão é algo necessário para poder informar ao Modelo Especialista em qual estágio da resolução o aluno está, além de, se necessário, resolver a partir deste ponto o problema vigente. Áreas do conhecimento como Álgebra ou Programação são de complexa representação, geralmente necessitando um grande conjunto de regras para formar esta sintaxe.

Especificar o Currículo: Criar o currículo de conteúdos a ser abordado em um Sistema Tutor Inteligente pode ser feito com a identificação da seqüência de regras de produção a serem ensinadas ao aluno em questão. Estas regras podem estar agrupadas em pequenas unidades que correspondam às seções vistas em determinadas bibliografias. Sistemas em geral tendem

a controlar os problemas que são apresentados aos alunos, bem como a ordem em que aparecem.

Embarcar Conhecimento Padrão: Existem três classificações de conhecimento que podem ser representadas em um Sistema Tutor Inteligente, segundo (POLSON e RICHARDSON, 1988):

- *Conhecimento Procedural:* É o conhecimento sobre como realizar uma determinada tarefa, geralmente representado de forma satisfatória em um Sistema Tutor Inteligente como um conjunto de regras.
- *Conhecimento Declarativo:* Este contrasta com o Conhecimento Procedural por ser orientado a fatos, não especializado para um uso em particular.
- *Conhecimento Qualitativo:* Trata-se da compreensão causal que permite uma pessoa raciocinar sobre os comportamentos usando modelos mentais.

Exploração/Visualização do Conhecimento: Definir as limitações da interatividade em um STI é um problema considerável. Se por um lado o sistema não permitir que o estudante navegue dentro do sistema para onde ele quiser, acaba limitando o uso da sua curiosidade em explorar novas áreas; de forma alternativa, se não houver iniciativa do sistema em conduzir um estudo inicial, este mesmo estudante pode se encontrar numa situação confusa, sem saber como prosseguir nas lições. O que se busca é um balanço entre estes extremos, possibilitar ao estudante uma exploração livre de conteúdos, porém com uma entidade que o guie neste processo, fornecendo ajuda e idéias que o façam tirar melhor proveito da interação com o sistema.

Usar uma ferramenta de visualização de conhecimentos dentro do sistema é a maneira de fazer com que os alunos lidem com grandes quantidades de conteúdos. No entanto, um bom projeto de interface consome um tempo considerável do cronograma de qualquer projeto, nem sempre sendo desenvolvido em sua totalidade.

2.5 AVALIAÇÃO QUALITATIVA

No contexto de avaliação de materiais didáticos digitais, o professor participa como usuário direto e indireto. Considerado usuário direto na fase de planejamento de suas atividades, quando está selecionando/avaliando o *software* educativo; e usuário indireto na fase de aplicação, quando participa como um facilitador/mediador da interação alunos-*software* educativo. Apesar do professor não ser o público alvo principal dos *softwares* educativos, ele julgará se o *software* é viável para utilização no contexto educacional utilizando instrumentos avaliativos (GODOI e PADOVANI, 2009).

O trabalho de (DIX, 1998) define que a avaliação de um sistema possui três objetivos principais: avaliar a funcionalidade desse sistema; avaliar o efeito de sua interface sobre o usuário; e identificar algum problema específico com o sistema. Desta forma, pode-se inferir que a avaliação de material didático digital é uma tarefa complexa e deve ser efetuada tanto na fase de desenvolvimento quanto na fase de utilização do *software* educativo.

À medida que ganha espaço o interesse por políticas de teor qualitativo – questão cultural, identidade comunitária, participação e espaço político – torna-se interessante buscar caminhos de avaliação que lidem com estes fatores abstratos. A realidade social possui dimensões qualitativas. Não se nega a vigência da qualidade na realidade histórica e social, não se tratando de estabelecer entre qualidade e quantidade uma polarização radical, como se uma fosse o oposto da outra. Avançando um pouco sobre esta avaliação, encontra-se a chamada qualidade formal e qualidade política. Dentro da primeira temos instrumentos e métodos. Na segunda temos finalidades e conteúdos (DEMO, 2005).

Dizem que a qualidade é melhor percebida em esferas alternativas do saber, que não foram tão devassadas pela teimosia tecnológica e científica. Uma avaliação qualitativa dedica-se em perceber tal problemática para além dos levantamentos quantitativos usuais, que nem por isso deixam de ter sua importância. Não há razão para se polemizar contra apresentações quantitativas, de estilo empírico e estatístico, a não que ser que a análise se torne puramente empirista. Há uma diferença entre aproveitamento empírico da realidade e redução empirista, não havendo mal em qualquer avaliação qualitativa vir a ser secundada por dados quantitativos, até porque são considerados inevitáveis. Não significa recair em um cenário que nega qualquer importância a análises quantitativas ou que se esconde atrás de uma linguagem confusa e dispersa.

Para avaliar os processos participativos, embasadores da captação de qualidade, é necessário participar, não bastando mera observação participante. Esta não é combatida, em um sentido mais denso exige tempo de convivência e compromisso comprovado. A avaliação qualitativa de processos coincide logicamente com a auto-avaliação, o que contraria a atitude

de mero observador. Não é uma iniciativa externa, sendo somente factível em profundidade, como forma de auto-expressão. Pode-se analisar a participação dos outros, mas, se fizermos somente isto, perdemos o cerne do fenômeno participativo, que á a autopromoção. A qualidade não se capta observando-a, mas sim vivenciando-a.

O que interessa, na verdade, é o conteúdo, não a forma. Seria um erro tentar formalizar de partida a avaliação qualitativa, porque nisto já perderia em qualidade. Uma forma adequada de expressão é o depoimento, o testemunho. Não um relatório, no sentido formal. Mas a transcrição vivencial de um conteúdo participativo. Certamente pode predominar o subjetivo, porque é depoimento, não uma análise obtida pela observação.

Um diagnóstico qualitativo é aquele autodiagnóstico, ou seja, ao mesmo tempo um fenômeno político em sua essência. Assim, onde não há se delineia o fenômeno participativo não há, em princípio, o que avaliar qualitativamente. Esta colocação leva a entender que uma avaliação qualitativa não é tema para mestrado, a não ser em casos de relativa maturidade teórica e prática. É mais fácil fazer uma dissertação de Mestrado em campo quantitativo, sobretudo em casos que não vão além de um exercício acadêmico. Todavia, uma dissertação pode admitir ‘cuidados qualitativos’ com maior facilidade quando o pesquisador se esforça por tratar a comunidade com respeito, devolver a ela dados colhidos além de conviver certo tempo com ela.

Toda avaliação qualitativa supõe no avaliador qualidade metodológica. Isto significa dizer que não faz sentido desprezar o lado da quantidade, desde que bem feito. Só tem a ganhar a avaliação qualitativa que souber se cercar inteligentemente de base empírica, mesmo porque a qualidade não é contradição lógica da quantidade, mas faces contrárias da mesma moeda, não se excluem.

2.5.1 Ferramentas para Avaliação Qualitativa de Softwares Educacionais

Para embasar a avaliação realizada com o protótipo, foram avaliadas as seguintes alternativas: CSEI (*Children Software Evaluation Instrument*) (CHILDREN TECHNOLOGY REVIEW, 1998; BUCKLEITNER, 1999), PROINFO (Programa Nacional de Informática na Educação) (PROGRAMA NACIONAL DE INFORMÁTICA, 1998), MAQSEI (Metodologia de Avaliação de Qualidade de Software Educacional Infantil) (ATAYDE, 2003) e o trabalho de Mestrado intitulado “Um Instrumento de Avaliação da Qualidade para Software Educacional de Matemática” (GLADCHEFF, 2001), trabalhos que trazem alternativas a

avaliação de qualidade de software no cenário educacional, visando auxiliar o processo de um avaliador que deseje aplicar um sistema destes em sala de aula.

CSEI – *Children software Evaluation Instrument*

Trata-se de uma ferramenta de avaliação usada para fazer inspeção de sistemas, baseada em seis categorias que definem o que é qualidade de *software* infantil. Este instrumento permite identificar a facilidade de uso do *software*, se é possível ser controlado pelas crianças, solidez do conteúdo educacional, fatores lúdicos e se o *design* corresponde às expectativas do usuário. É considerado um método de avaliação somático, pode ser utilizado antes, durante e depois da utilização do *software* devendo esta avaliação ser aplicada por professores. A lista abaixo contém uma lista de questões a serem respondidas pelo professor a respeito do software que será utilizado. Cada um dos tópicos abaixo é composto por uma série de características que devem ser mensuradas seguindo o critério:

1 – Sempre;

0.5 - Parcialmente ;

0 – Nunca.

N.A – Não mensurável.

As categorias de características são: Facilidade de Uso, Faixa-Etária, Valor Educacional, Valor de Entretenimento e Características de Design. Após atribuir valores as características, se divide o somatório obtido pelo número de características, resultando em um valor entre zero e cinco, sendo o primeiro a pior avaliação e o último a melhor possível.

PROGRAMA NACIONAL DE INFORMÁTICA NA EDUCAÇÃO – PROINFO

Trata-se de um relatório apresentado no III Encontro Nacional do PROINFO proposto pelo MEC, contendo 20 perguntas que orientam o professor numa avaliação sobre *software* educativo. É um método somático e prognóstico, devendo ser utilizado antes da utilização do *software* educativo em questão em sala de aula. De acordo com o relatório, todo recurso utilizado em sala de aula (incluindo o *software* educativo) deve passar por análise prévia do professor. A metodologia mais comumente utilizada para se fazer avaliação de *software* educacional tem sido por meio de *checklist*, isto é, um conjunto de questões específicas e preestabelecidas, que visam conduzir o processo de avaliação, que deve ser realizado por

professores que queiram utilizar uma ferramenta educacional em sala de aula. A lista abaixo compreende o *checklist* proposto para avaliação de um software educacional:

1. Qual a proposta pedagógica que permeia o *software*?
2. Proporciona um ambiente interativo entre aluno e o *software*? Como?
3. Permite uma fácil exploração?(seqüencial, não linear)
4. Apresenta conceitos de forma clara e correta?
5. Desperta o interesse do aluno, sem perder de vista os objetivos do *software* e do usuário?
6. Oferece alternativas diversificadas para a construção das ações do aluno?
7. Permite que o aluno construa seu conhecimento a partir da ação-reflexão-ação?
7.1. Tem recursos de programação?
7.2. Permite o registro e a consulta das ações desenvolvidas?
8. Os recursos de multimídia usados têm relevância para os objetivos do *software*?
9. Proporciona condições de abordagem sócio-cultural que contemple aspectos regionais? As especificações no *software* são compatíveis com a configuração dos equipamentos existentes na sala de aula?
10. E os aspectos técnicos da escola?
11. É de fácil instalação e desinstalação?
12. Permite a utilização em rede?
13. Apresenta uma visão interdisciplinar?
14. Apresenta encarte com explicações sobre objetivos, conteúdos, equipe de desenvolvimento do *software* e sugestões metodológicas para a sua utilização?
15. Em que idioma o *software* é apresentado? Existe uma versão em português?
16. Em relação aos demais recursos didáticos, qual o diferencial que o *software* apresenta?
17. Proporciona um ambiente de aprendizagem por descoberta?
18. Permite a integração com outros *software*?
19. Apresenta um ambiente lúdico e criativo?
20. Qual o tipo de *software* (jogo, tutorial, exercício- prática, autoria, outros)?

MAQSEI – Metodologia de Avaliação de Qualidade de Software Educacional Infantil

A metodologia MAQSEI é composta por quatro fases (Reconhecimento e proposta de avaliação do *software*, Planejamento dos testes, Realização dos testes e Análise dos dados e

produção de relatório final de avaliação). A fase do Reconhecimento e proposta de avaliação do *software* consiste no primeiro contato para conhecimento do programa a ser avaliado, em que o avaliador deve usar todas as funções do *software*. A etapa de planejamento dos testes é tão importante quanto a própria realização dos testes, em que se define o que será necessário para a realização da avaliação torna-se mais simples ao se especificar exatamente o que e quando ocorrerá. Tendo sido completada a fase de planejamento dos testes de avaliação, pode-se começar os testes propriamente ditos. A avaliação de um *software* consiste na aplicação do teste preparado com vários participantes, que, individualmente ou em duplas, serão observados e questionados pelo condutor durante o teste. A fase de análise dos dados: Essa fase consiste na transformação de todos os dados coletados em resultados e recomendações sobre o *software*. Abaixo seguem os critérios que compõem a tabela que orienta a realização do relatório final de avaliação:

Pertinência em relação ao programa curricular: O *software* educacional infantil deve ser adequado e pertinente em relação a uma disciplina específica ou a um contexto educacional e permitir a identificação do modelo de aprendizagem que ele privilegia, deixando explícitos os seus objetivos pedagógicos.

Utilização de recursos computacionais: O *software* educacional infantil deve aproveitar as qualidades únicas do computador como meio. A simples transferência de conteúdos para um programa não traz ganhos para a educação.

Avaliação da aprendizagem: É importante que o *software* permita que o(a) docente ou os pais verifiquem se e quais os conceitos estão sendo aprendidos pelos alunos ou seja, a efetividade da aprendizagem.

Interação: O *software* educacional infantil deve proporcionar uma boa condução da criança durante a interação com suas interfaces, que facilite o aprendizado e utilização do programa, e, consequentemente, melhore o desempenho e diminua o número de erros. Relacionam-se com a condução as heurísticas de reconhecimento no lugar de memorização, qualidade das opções de ajuda, legibilidade, *feedback* e agrupamento e distinção de itens.

Adaptabilidade: Um *software* educacional infantil não consegue abranger a todo momento todo o seu público alvo, mas pode ser capaz de adaptar-se às necessidades e preferências de

diferentes perfis de criança. As heurísticas de flexibilidade, nível de experiência com o *software* ou com computadores e ambiente cooperativo relacionam-se com a adaptabilidade.

Controle e autonomia do usuário: O controle e liberdade do usuário refere-se tanto ao controle que as crianças devem ter sobre o processamento de ações solicitadas quanto à liberdade que devem ter sobre a
utilização destas ações. O acesso, saída, adiamento e anulação são heurísticas relacionadas.

Recursos motivacionais: O uso de recursos motivacionais (figuras, sons, animações) tem grande importância em *software* do tipo educacional infantil. Estes recursos podem estimular as crianças e
proporcionar o aumento da vida útil do programa ao fazer com que ela deseje usá-lo por mais vezes.

Gestão de erros: O *software* educacional infantil deve tratar os erros que ocorrerem de forma diferenciada, dependendo da sua classificação: erros de utilização ou erros conceituais. Duas heurísticas estão relacionadas à gestão de erros: prevenção de erros e auxílio no reconhecimento, no diagnóstico e na recuperação de erros. A primeira diz respeito a erros de utilização do *software* e a segunda, aos erros de utilização ou conceituais.

Carga de trabalho: A carga de trabalho refere-se a toda informação contida nas interfaces do software educacional infantil que serão utilizadas para a realização de tarefas. As heurísticas de carga informacional e brevidade referem-se à carga de trabalho.

Conteúdo: O *software* educacional infantil deve apresentar seu conteúdo, ou seja, toda a informação contida no *software* que se intenciona transmitir às crianças, de forma objetiva e adequada a uma proposta pedagógica.

Significado de códigos e denominações: O significado de código e denominações refere-se à correspondência entre o objeto apresentado e a respectiva informação apresentada.

Consistência e Padrões: Consistência e padrões referem-se à uniformidade na apresentação de elementos e informações de um *software*, que ajuda a evitar que a criança tenha dúvidas se palavras, situações ou ações diferentes no *software* significam ou não a mesma coisa.

*Correspondência entre o software e o mundo real***:** A correspondência com o mundo real refere-se à escolha e uso de padrões, convenções ou associações familiares à criança no *software* educacional infantil. A utilização destas convenções fazem a informação aparecer numa ordem lógica e natural para a criança, facilitando a compreensão e utilização do programa. A correspondência com outros programas similares está relacionada com esta heurística.

*Documentação***:** O *software* educacional infantil pode fornecer documentação direcionada tanto para pais/docentes quanto para crianças. No caso de adultos, a documentação refere-se à descrição do *software* (identificação, recursos necessários, objetivos, entre outros) e ao uso (instalação e instrução), enquanto para as crianças, a documentação refere-se somente ao uso do programa (instrução).

Um Instrumento de Avaliação da Qualidade para Software Educacional de Matemática

Segundo o trabalho de (GLADCHEFF, 2001) antes de avaliar-se um software educacional se deve enquadrar ele segundo uma das seguintes categorias: Instrução Programada, Simulação e Jogos Pedagógicos ou Sistemas Hipermídia.

Instrução Programada

A instrução programada consiste em dividir o material a ser ensinado em módulos, em que cada fato ou conceito é apresentado sequencialmente. Assim que o estudante interage com o mesmo é questionado logo em seguida. Em caso de sucesso na resposta, o próximo módulo é passado ao aluno. Em caso negativo, a resposta certa pode ser fornecida ou o aluno é sugerido a rever módulos prévios. Em geral softwares educacionais deste tipo solicitam o nome do aluno e o nível de dificuldade que o mesmo deseja enfrentar, não havendo uma heurística que enquadre ou infira o seu estado atual.

Exercício e prática: Estes são considerados uma maneira comum de utilização do computador na educação, sua utilização está na revisão de materiais vistos nas aulas, em especial

conteúdos que trabalham memorização e repetição. Uma das vantagens do uso destes programas é o fato do professor dispor de uma infinidade de exercícios que o aluno pode resolver de acordo com o seu grau de conhecimento e interesse.

Tutoriais: Estes constituem a versão computadorizada da instrução programada por completo, ou seja, antes de o aluno ser questionado informações são apresentadas. Uma das vantagens dos tutoriais é a possibilidade de apresentação de materiais a serem trabalhados com algumas características que não são permitidas no papel, tais como, animação, som, vídeo e a possibilidade do professor ou até mesmo o próprio aluno manter um controle sobre seu desempenho, no caso do sistema utilizar um Modelo de Aluno, por exemplo. Sistemas desta categoria podem utilizar técnicas de Inteligência Artificial (pesquisa em Sistemas Tutores Inteligentes) para analisar padrões de erro, avaliar estilo e a capacidade de aprendizagem do aluno, assim como oferecer instrução especial sobre o conceito que o aluno está apresentando dificuldade. Os exercícios presentes nestes ambientes costumam requisitar respostas imediatas do aluno, propiciando *feedback* adequado, fazendo com que a progressão dos alunos aconteça no seu próprio ritmo. O seu objetivo principal é levar o computador a instruir o aluno, em uma determinada área do conhecimento, através de um contato individualizado, simulando um tutor particular.

Simulação e Jogos Pedagógicos

Softwares de simulação: Sistemas recriam modelos simplificados do mundo real, modelos de sistemas complexos dinâmicos. Estes permitem a exploração de situações fictícias, de situações de risco, experimentos caros ou que consumiriam tempo para serem executados. Simulações pedagogicamente relevantes podem ser realizadas com grande complexidade e realismo através de ferramentas computacionais. As simulações oferecem ao aluno a possibilidade de desenvolver hipóteses, testá-las, analisar resultados e refinar os conceitos. Estes ambientes também costumam oferecer um ambiente para o trabalho em grupo, em que vários grupos podem testar hipóteses diferentes. Embora válidas, estas simulações devem ser vistas como um complemento às apresentações formais, leituras e discussões em salas de aula, pois sem estas não existe a garantia do aprendizado.

Jogos pedagógicos: A abordagem pedagógica nesta modalidade é a exploração ao invés da instrução explícita e direta. Com os jogos, o aprendizado acontece a partir da vivência lúdica e

da sua respectiva reflexão. Do ponto de vista da criança, estas constituem a maneira mais divertida de aprender. Os jogos pedagógicos podem ser utilizados para a aprendizagem de conceitos de difícil assimilação, tendo como problema o fato de que a competição pode desviar a atenção em relação ao conceito envolvido. Com isso, o objetivo pode passar a ser unicamente vencer no jogo e a questão pedagógica ficar em segundo plano.

Sistemas Hipermídia

O termo hipertexto é atribuído a Ted Nelson que o criou em 1960, referindo-se aos primeiros sistemas construídos com a filosofia de ligações embutidas. A abordagem de descrição mais simples é como um texto estruturado em rede. O termo hipermídia é considerado como uma extensão do termo hipertexto, implicando em ligações e navegações através de materiais armazenados em diferentes mídias. As informações são armazenadas em uma coleção de nós, existindo ligações entre estes, habitualmente conectados por links. É através desta estrutura que os usuários podem acessar a informação, navegando na mesma.

Esses sistemas podem possuir uma elevada interatividade com o usuário, adotando formas abertas de navegação, ou seja, permitem a descoberta imprevista e a descoberta de exploração livre. Aqueles menos interativos acabam restringindo a liberdade de navegação, fornecendo uma aprendizagem de maior recepção direcionada. Uma característica destes sistemas é que o foco está no controle das lições pelo próprio aprendiz. Mais do que apresentar a informação conectada em nós (ligações), os ambientes de aprendizagem de hipermídia permitem uma reflexão sobre o conteúdo que está sendo utilizado.

Questionário

Após o avaliador enquadrar o software educacional em uma das categorias acima, a autora propõe uma série de questionários a serem respondidos para avaliar a qualidade do sistema em questão. Estes questionamentos são baseados em cada uma das categorias anteriormente comentadas. Dentre as categorias, o trabalho proposto dentro do sistema PAT2Math faz com que o mesmo se enquadre como um *Tutorial,* tendo em vista a apresentação de conteúdos que acontece antes da exposição de exercícios, além do aluno ter as suas interações e informações registradas em um Modelo de Aluno, que serve como fonte de variáveis para uma técnica de Inteligência Artificial, no caso uma estratégia pedagógica,

selecionar conteúdos apropriados para este aluno em questão. Por estes motivos, será abordado o questionário do tipo *Tutorial* dentre as opções que este trabalho aborda.

A lista a seguir descreve o questionário proposto no trabalho original proposto em (GLADCHEFF, 2001).

Vocabulário

TUT.1 – A linguagem utilizada está no nível de compreensão do aluno? (vocabulário, metáforas, etc.) ( ) SIM ( ) NÃO

TUT.2 – O vocabulário é adequado, sem deixar de ser científico quando necessário?
( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

TUT.3 – O tutorial utiliza termos iguais com significados diferentes? ( ) NÃO ( ) SIM

TUT.4 – O tutorial utiliza termos diferentes com o mesmo significado? ( ) NÃO ( ) SIM

Conceitos Matemáticos

TUT.5 – Os conceitos matemáticos definidos pelo tutorial estão corretos?
( ) SIM ( ) NÃO

TUT.6 – Os conceitos matemáticos definidos pelo tutorial são precisos, ou seja, os conceitos são definidos de forma clara, sem utilização de palavras ambíguas?
( ) SIM ( ) NÃO

TUT.7 – Os conceitos são colocados de forma adequada, utilizando, sempre que possível, estratégias de simulação ou mesmo de jogos, histórias motivadoras, a fim de ganhar a atenção do aluno? ( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

TUT.8 – Os exemplos são representativos dos conceitos a que se referem?
( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

TUT.9 – As notações são adequadas e coerentes com as utilizadas pelo professor em sala de aula? ( ) SIM ( ) NÃO

Conteúdo

TUT.10 – O tutorial é abrangente no sentido de abordar o máximo possível dentro do assunto a ser trabalhado, e coincide com a organização do currículo adotado na escola?
( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

TUT.11 – Se o tutorial abordar o assunto a ser trabalhado de forma mais abrangente do que o adotado na escola, você se considera no momento preparado para utilizá-lo?
( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO ( ) NN
TUT.12 – O tutorial caminha do básico ao profundo de forma suave? ( ) SIM ( ) NÃO

TUT.13 – O tutorial permite a revisão do conteúdo que já foi trabalhado?
( ) SIM ( ) NÃO

TUT.14 – O tutorial permite modificações do conteúdo por parte do professor?
( ) SIM ( ) NÃO ( ) NN

TUT.15 – O tutorial possui uma base de dados com conhecimentos enciclopédicos sobre o domínio (assunto a ser trabalhado)? ( ) SIM ( ) NÃO

TUT.16 – O tutorial possui exemplos, questões de revisão e definições necessárias para sanar dúvidas com relação aos pré-requisitos exigidos para sua utilização?
( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO
TUT.17 – O tutorial permite que o conteúdo seja particularizado para cada aluno, ou seja, permite que o conteúdo a ser abordado seja limitado de acordo com o que se deseja trabalhar com cada aluno individualmente? ( ) SIM ( ) NÃO ( ) NN

Exercícios

TUT.18 – Os exercícios propostos pelo tutorial são representativos da realidade do aluno, sempre que possível? ( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

TUT.19 – Os enunciados dos exercícios propostos permitem que o aluno entenda o que está sendo pedido? ( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

TUT.20 – Os exercícios propostos são variados e apresentados de forma interessante?
( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

TUT.21 – Na apresentação dos exercícios o software utiliza ao máximo os recursos da máquina? (som, imagem, animação, ...)
( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

TUT.22 – Há um equilíbrio entre o conteúdo exposto e os exercícios propostos?
( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

TUT.23 – O tutorial pode gerar problemas dinamicamente levando em conta as necessidades do aprendiz? (O software gera um modelo do aprendiz e a partir deste modelo pode criar problemas dinamicamente) ( ) SIM ( ) NÃO

Apresentação de Problemas

Caso o tutorial aborde o conhecimento matemático com o objetivo de ser aplicado na resolução de problemas rotineiros e não rotineiros, as questões de TUT.24 a TUT.24.5 devem ser respondidas.

TUT.24 – O tutorial propõe problemas?
( ) SIM ( ) NÃO
Se Sim, responda:

TUT.24.1 – O tutorial propõe problemas envolventes e desafiadores, de acordo com a faixa etária a que se destina?
( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

TUT.24.2 – O tutorial propõe problemas significativos, que dizem respeito a realidade do aluno?
( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

TUT.24.3 – O tutorial possibilita que o aluno formule hipóteses?
( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

TUT.24.4 – O tutorial possibilita vários caminhos para a solução dos problemas?
( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

TUT.24.5 – O esquema utilizado pelo tutorial para direcionar a criança à resolução do problema é adequado?
( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

Usabilidade

TUT.25 – As orientações dadas pelo tutorial sobre sua forma de utilização são claras e fáceis de serem entendidas? ( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

TUT.26 – O tutorial é objetivo, ou seja, possui um caminho objetivo que direciona o aluno para onde deve ir após a etapa em que se encontra, fazendo com que ele não se perca? ( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

TUT.27 – O tutorial permite que "sessões" interrompidas sejam reiniciadas a partir do "ponto de parada"? ( ) SIM ( ) NÃO

Feedback

TUT.28 – Qual é a forma de *feedback* emitida pelo tutorial quando o aluno erra a resposta do exercício proposto?
( ) repetição – O tutorial simplesmente reapresenta a pergunta anteriormente feita ao aluno.
( ) pista – O tutorial fornece uma mensagem na intenção de chamar a atenção do aluno sobre o fundamento do erro cometido, com o objetivo de fazer com que o aluno descubra o que "implicitamente" já sabe.
( ) explicação através de mensagem padrão – O tutorial fornece uma única mensagem de explicação para todo e qualquer erro, no sentido de simplesmente "não ser a resposta correta".
( ) explicação em função da resposta do aluno – A resposta do aluno é analisada na sua originalidade e uma explicação é colocada de acordo com esta resposta.

TUT.29 – O *feedback* realizado pelo tutorial permite que o aluno reflita sobre seu erro e tente corrigi-lo, sem intervenção ostensiva do professor?
( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

TUT.30 – Você considera a forma de *feedback* emitida pelo tutorial adequada?
( ) SIM ( ) NÃO

TUT.31 – As respostas do aluno são verificadas corretamente?
( ) SIM ( ) NÃO

TUT.32 – O tutorial justifica suas ações ou raciocínios? (Quando, por exemplo, mostrar ao aluno a forma de resolução de algum exercício) ( ) SIM ( ) NÃO

TUT.32.1 – Se SIM, as justificativas ou raciocínios estão corretamente empregados?
( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

TUT.33 – O tutorial consegue acessar e analisar as razões das respostas do aprendiz?
( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

Desempenho dos Alunos

TUT.34 – O tutorial oferece um relatório sobre o desempenho do aluno para que seja possível verificar se os objetivos da lição foram alcançados? (número de respostas certas, número de respostas erradas para cada sessão, etc.)
( ) SIM ( ) SIM, MAS COM POUCAS INFORMAÇÕES ( ) NÃO

TUT.35 – O tutorial mantém um histórico de utilização por parte do aluno? (número de sessões que o aluno realizou, tempo gasto em cada módulo, etc.)
( ) SIM ( ) SIM, MAS COM POUCAS INFORMAÇÕES ( ) NÃO

Avaliação das Metodologias Propostas

A metodologia CSEI apresenta um questionário bastante flexível, podendo ser aplicado tanto na fase de design, desenvolvimento ou ainda aplicação do sistema educacional, que é visto de maneira genérica, sem especificações de categorias, fato provado pelas questões lúdicas e de entretenimento em sua estrutura, não presentes em todos os sistemas educacionais.

A abordagem proposta pela PROINFO se concentra em uma avaliação de caráter prognóstico, a fim de verificar se um sistema está apto a fornecer uma boa experiência em sala de aula. É realizada por um professor, que responde um questionário sobre um sistema e pode então julgar se a sua respectiva aplicação é uma boa alternativa. No entanto, este

questionário não está dividido em categorias de características de um sistema, está disposto em um caráter mais superficial e abrangente, que pode comprometer a avaliação de sistemas mais complexos e ricos em interação e/ou conteúdos.

A MAQSEI parece ser uma metodologia mais completa, explicitando etapas de avaliação e oferecendo critérios bem explanados para avaliação e elaboração de um relatório final do processo. Porém, não há distinção entre os tipos de software educacional, o que leva a acreditar numa generalização da metodologia para tais sistemas, característica diferencial do último trabalho abordado, por (GLADCHEFF, 2001).

O trabalho de Mestrado de (GLADCHEFF, 2001) foi selecionado para realizar a avaliação do trabalho proposto, conforme a seção “Avaliação”. Esta faz uma abordagem mais completa, explicitando diferenças entre sistemas educacionais para que o avaliador faça um enquadramento inicial do que está avaliando, para em um segundo momento seguir na metodologia com questões específicas sobre o tipo de sistema em questão. A metodologia como um todo prevê ainda um questionário posterior a ser aplicado conforme o ciclo letivo da turma que for utilizar. Como não houve aplicação em sala de aula do protótipo construído neste trabalho, esta parte da metodologia foi omitida na sua descrição anteriormente.

3. TEORIAS DE APRENDIZAGEM

O ensino através de um ambiente computacional exige um estudo de como um indivíduo constrói conhecimento, para que tais características estejam previstas na maneira de ensinar deste sistema. Trata-se de um processo complexo cuja teorização exige um estudo da mente humana, de como acontece internamente o nosso processo de aprendizagem.

Avaliando os tipos gerais de aprendizagem, como está retratado em (MOREIRA, 1999), pode-se afirmar que a aprendizagem resultante do armazenamento organizado de informações na mente do aluno é considerada *cognitiva*; a resultante de sinais internos ao indivíduo e pode ser identificada com experiências tais como prazer *versus* dor ou ainda alegria *versus* ansiedade é considerada *afetiva*; e por fim, a que envolve respostas musculares adquiridas por meio de treino e prática, é nomeada *psicomotora.* Tais conceitos embasam os três tipos de ensino clássicos, em que os grandes teorizadores de ensino se enquadram:

- Ensino Behaviorista: Volta-se para eventos observáveis no mundo exterior ao indivíduo; provê uma base para o estudo de manipulações que produzem mudanças comportamentais;
- Ensino Cognitivista: Enfatiza o processo da cognição, por meio do qual o mundo de significados tem origem. À medida que o aluno aprende, estabelece relações de significação, atribuindo significados à realidade que se encontra;
- Ensino Humanístico: considera primordialmente o aluno como pessoa. Livre para fazer escolhas em cada situação. O importante é a auto-realização.

A premissa da estrutura cognitiva do aluno em relação a um conteúdo enquadra o problema dentro do *ensino cognitivista,* e tendo este como ponto de partida, pode-se avaliar o trabalho de autores desta categoria para complementar o embasamento do projeto.

3.1 APRENDIZAGEM SIGNIFICATIVA

Ausubel (AUSUBEL, 1982) se enquadra como um autor que propõe teorias cognitivas, diferenciando-se dos demais behavioristas como Skinner e humanistas como Rogers, abordados em (MOREIRA, 1999). Ele propõe o conceito de aprendizagem

significativa, em que uma nova informação relaciona-se com um aspecto relevante da estrutura de conhecimento existente do aluno, envolvendo um processo de interação desta nova informação e da estrutura cognitiva já existente no indivíduo. Esta estrutura cognitiva representa a idéia de uma hierarquia de conceitos e representações de experiências sensoriais do indivíduo. Ausubel considera este armazenamento de informações no cérebro humano como uma tarefa organizada, gerando uma hierarquia conceitual, em que elementos mais específicos são ligados a conceitos mais gerais. Tais conceitos presentes nesta estrutura são chamados *subsunçores*.

Como exemplo, se o aluno aprende significativamente que o cão é um mamífero, ele deverá ser capaz de expressar isso de diversas formas, como: “o filhote de cachorro mama de sua mãe” ou “o cachorro é um animal que, como nós, mama quando é filhote”. Este aprendizado significa que o aprendiz apreendeu o sentido, o significado daquilo que se ensinou, de modo que pode expressar este significado com as mais diversas palavras. Para Ausubel, o objetivo maior do ensino acadêmico é que todas as idéias sejam aprendidas de forma significativa. Isso porque é somente desta maneira que estas novas idéias serão retidas na memória de maneira estável. Além disso, a aprendizagem significativa permite ao aprendiz o uso do novo conceito de forma inédita, independentemente do contexto em que este conteúdo foi primeiramente aprendido. O extremo oposto da aprendizagem significativa é a mecânica. Neste caso, as novas idéias não se relacionam de forma lógica e clara com nenhuma idéia já existente na estrutura cognitiva do sujeito, são simplesmente memorizadas. Desta maneira arbitrária, não há garantia de flexibilidade no seu uso, nem tampouco longevidade da informação na mente do indivíduo. Como conseqüência dessa falta de flexibilidade, o aluno não é capaz de expressar o novo conteúdo com linguagem diferente àquela que o material foi aprendido. Tal aluno não aprende o significado, o sentido do novo material. Por conta disso, ele seria incapaz de utilizar este conhecimento em um contexto diferente daquele no qual fora primeiramente apresentado a estes conceitos ou idéias. No exemplo dado acima - do cachorro ser um mamífero - o indivíduo seria incapaz de fazer a relação entre o cachorro e o ser humano, ou mesmo com o fato de que quaisquer mamíferos mamam. É importante ressaltar que, apesar de Ausubel ter enfatizado a aprendizagem significativa, ele compreendia que no processo de ensino-aprendizagem existem circunstâncias em que a aprendizagem mecânica é inevitável. No ensino de História, por exemplo, conhecer e entender os eventos que se sucederam no surgimento e desenvolvimento do Império Romano requer, muitas vezes, que se saibam os nomes de diversas de suas instituições e personagens principais, o que é tipicamente um aprendizado mecânico.

A aprendizagem significativa é caracterizada pela interação entre os novos conhecimentos e os já existentes subsunçores, na qual ambos se modificam de forma subordinada ou superordenada. No caso da maneira subordinada, a relação adquire significado através da estrutura preexistente no aluno. Quando isto ocorre, de maneira *bottom-up*, acontece a *diferenciação progressiva.* Trata-se do princípio que progressivamente conceitos vão sendo diferenciados e expandidos em termos de detalhe e especificidade. Na superordenada, o conceito a ser aprendido é considerado mais geral e inclusivo do que as idéias contidas nesta estrutura prévia, servindo de base, se enquadrando em um nível superior dos subsunçores já presentes no indivíduo, concretizando a *reconciliação integrativa*. A 'Figura 4: Diagrama Aprendizagem Significativa' demonstra este princípio. No caso do Conceito Subsunçor B interferir na estrutura de forma a ser considerado mais geral que o Conceito Subsunçor A, ocorre a Reconciliação Integrativa. Já no cenário em que o Conceito Subsunçor B for classificado como uma especificação mais detalhada do Conceito Subsunçor A, acontece então a Diferenciação Progressiva.

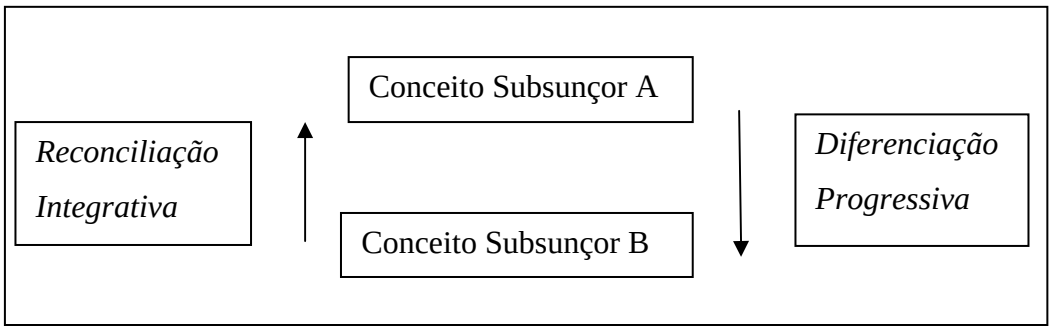

Figura 4: Diagrama Aprendizagem Significativa

Segundo Ausubel, existem três tipos de aprendizagem significativa: a aprendizagem representacional, a aprendizagem de conceitos e a aprendizagem proposicional, descritas a seguir:

- Aprendizagem Representacional: Categoria mais básica de aprendizagem significativa, do qual as demais dependem. Envolve atribuição de significados a símbolos (tais como palavras ou objetos);
- Aprendizagem de Conceitos: É considerada uma aprendizagem representacional também, no entanto ocorre entre conceitos abstratos, já formados a partir do passo anterior. Relaciona de maneira mais genérica representando abstrações de eventos ou objetos.

- Aprendizagem Proposicional: Constitui a tarefa de aprender o significado do que está além da soma de significados de palavras ou conceitos em conjunto. Representa uma abstração em cima de conceitos previamente absorvidos e assimilados pela aprendizagem de conceitos.

Existe também o conceito ausebeliano de *organização seqüencial* que versa sobre a disponibilidade de idéias-âncora relevantes a serem consideradas no ensino, provenientes da natural hierarquia entre conteúdos de uma disciplina, por exemplo. Este processo de aprendizagem significativa deve ser explorado com a maximização da identificação de conceitos subsunçores a um dado conteúdo, através das dependências naturais dos conteúdos a serem ensinados.

Além da diferenciação progressiva e reconciliação integrativa, existe outro caso de enquadramento nos subsunçores. Este tipo de aprendizagem acontece quando a nova idéia não está hierarquicamente acima nem abaixo da idéia já existente na estrutura cognitiva à qual ela se relacionou. Esta nova idéia não é exemplo nem generalização daquilo que se usou como âncora para ela na estrutura cognitiva do indivíduo.

Um exemplo deste tipo de aprendizagem é o caso da metáfora que se faz de um sistema elétrico com um hidráulico. Neste caso, usam-se conceitos já dominados pelo indivíduo com relação aos sistemas de águas, para ensinar conceitos novos e que guardam alguma relação com os antigos que serviram como âncora. Mas os sistemas elétricos não são uma generalização nem um exemplo de sistemas hidráulicos, e vice-versa.

No entanto, para a maioria dos alunos é mais simples começar a lidar com os novos conceitos da eletricidade, a partir de conceitos com os quais já estão acostumados, relativos à hidráulica. É imprescindível que, nestes casos, as semelhanças e diferenças entre a idéia nova e a antiga que lhe serviu como âncora sejam progressivamente explicitadas, a fim de que o aluno não misture, confunda ou reduza os conceitos relativos de uma idéia aos da outra. Antes de terminar esta seção, faz-se necessário atentarmos para algumas considerações importantes.

Primeiramente, deve-se ter em mente que a proposta de Ausubel afirma que uma condição necessária para que se possa ter aprendizagem significativa é a nova idéia se relacionar de maneira não-arbitrária e substantiva com idéia(s) já existente(s) na estrutura cognitiva do indivíduo. No entanto, a cadeia de relações que existem – ou que pode ser construída – não é necessariamente “plana”, no sentido que uma idéia pode estar associada, por subordinação, superordenação ou de forma combinatória com uma ou com várias outras. E essa é exatamente uma das preocupações de Ausubel na sua proposta pedagógica: que se

estabeleçam (de forma lógica e não-arbitrária) as mais variadas conexões possíveis entre as novas idéias que estão sendo apresentadas, e entre elas e as idéias que o indivíduo já domina. Além disso, estas novas idéias, mesmo que não se tenham construídas todas as pontes possíveis com as idéias já existentes na estrutura cognitiva do indivíduo, podem progressivamente ir se interconectando umas com as outras, através do trabalho intelectual consciente do indivíduo, que busca e estabelece estas relações (denominado reconciliação integrativa, comentado anteriormente).

3.2 ESTÁGIOS DE APRENDIZAGEM (TAXONOMIA DE BLOOM E TEORIAS DE DESIGN INSTRUCIONAL)

Segundo a teoria da Taxonomia de Bloom (1956), percebe-se que existem diferentes maneiras de classificar o aprendizado, sintetizadas na seguinte maneira:

- *Conhecimento:* Relembrar conhecimento prévio, como uma citação, uma determinada lei.
- *Compreensão:* Entender o conteúdo e saber identificar ele em um dado contexto, como explicar com palavras próprias um determinado problema.
- *Aplicação:* Aplicar um conhecimento ou conteúdo em um cenário novo, abstraído a partir de um novo contexto, como aplicar leis estatísticas em um teste escrito para verificar a sua eficácia.

Isto se revela particularmente útil no momento de classificar os conteúdos de objetos de aprendizado e materiais de ensino dentro de um Sistema Tutor Inteligente.

Existem também as Teorias de Design Instrucional, que descrevem como fazer o projeto e organização destes materiais e conteúdos para estudantes de maneira que sejam eficientes e condizentes. Merril (MERRIL, 2001; MERRIL, 2002; MERRIL, 2007) realizou uma análise de certas abordagens para estas teorias, sinalizando princípios em comum a todas, percebendo cinco estágios que são necessários para ocorrer o aprendizado:

Problema: “O aprendizado é promovido quando aprendizes estão engajados em resolução de problemas do mundo real”. O aprendizado é mais simples de acontecer quando o estudante

está resolvendo problemas do mundo real, seu cotidiano, sua realidade. No início do ensino de um determinado conteúdo ficam consideravelmente mais motivados quando percebem o que estarão aptos a resolver depois de aprendê-lo. O problema não deve ser demasiadamente simples, nem tão complexo. Se um problema adequado não for encontrado, deve-se mostrar um grande problema e este aprendizado como parte para resolvê-lo. Para um material se enquadrar nesta categoria deve ser a seguinte pergunta: "O material envolve problemas ou tarefas do mundo real?"

Ativação: "O aprendizado é promovido quando o conhecimento existente é ativado como base para novo conhecimento". Quando existem experiências relevantes anteriores, a ativação no aprendizado é realizada. É importante que o estudante relembre, descreva e aplique conhecimento prévio dos conteúdos, durante o aprendizado de outros. Para se enquadrar como ativador de conhecimentos, o material instrucional deve orientar os alunos a relembrar, relacionar, descrever ou ainda aplicar conhecimento anterior como base deste.

Demonstração: "O aprendizado é promovido quando o novo conhecimento é demonstrado ao aprendiz". No momento em que algo real e concreto é exposto ao estudante sobre o que ele está aprendendo, ao invés de simplesmente citar algo abstrato, o aprendizado ocorre de maneira mais significativa. Se o material instrucional demonstrar, através de exemplos, o que será aprendido, ao invés de simplesmente dizer será considerado deste tipo.

Aplicação: "O aprendizado é promovido quando o novo conhecimento é aplicado pelo aprendiz". Os exercícios propostos aos estudantes devem ser selecionados tendo em vista os objetivos de aprendizado, por exemplo, os citados por (BLOOM, 1956), além dos materiais permitirem que o aluno aplique conhecimentos recém-adquiridos.

Integração: "O aprendizado é promovido quando o novo conhecimento é integrado ao mundo do aprendiz". No momento em que os estudantes transferem o que aprenderam no seu cotidiano em outros problemas da sua realidade, o aprendizado se torna um processo marcante. Segundo Merril (2001), se o estudante puder mostrar para o seu grupo de convívio o que aprendeu, estará mais motivado no que lhe é proposto ensinar.

3.3 MACRO-ADAPTION

A Macro-Adaption (CORBETT e ANDERSON, 1995; SHUTE, 1993) é uma estratégia pedagógica definida dentro das teorias que embasam a construção de Sistemas Tutores Inteligentes. É considerada a mais adaptável ao perfil do aluno, no entanto a de maior complexidade de implementação. Em um currículo de um STI que utilize esta estratégia, cada uma das *tarefas* possui quais habilidades que são trabalhadas com o seu respectivo aprendizado. O que esta estratégia realiza é uma avaliação de quais habilidades o aluno ainda não possui pleno domínio, sendo então selecionadas as tarefas que envolvem justamente estas habilidades menos desenvolvidas.

O STI pode designar ao aluno um conjunto de tarefas, sendo que o sistema sabe que componentes de conhecimento são exercitados por elas. Para cada um destes, o STI mantém uma estimativa do grau de competência do aluno em utilizá-los. No momento em que o estudante completou uma das tarefas o sistema precisa selecionar um próximo material, e este é selecionado baseado na sobreposição entre as tarefas que exercitem estes componentes, preferencialmente utilizando aqueles que o aluno ainda não tenha desenvolvido em sua totalidade. Adicionalmente, este modelo pode ser estendido para considerar a seleção de tarefas que trabalhem *misconceptions* (representação de conhecimentos incorretos) do aluno em questão, no caso as concepções errôneas a respeito de um conteúdo que o mesmo tenha. No caso do Módulo Tutor do sistema inferir que o estudante está realizando uma *misconception,* ele pode selecionar uma tarefa que venha a remediar este aprendizado incorreto.

3.4 APRENDIZADO DE ÁLGEBRA

Alunos de Ensino Fundamental têm grandes dificuldades durante o aprendizado de Matemática, sendo a Álgebra considerada um expoente entre tais assuntos. Mesmo entendendo o que um determinado problema propõe, eles não conseguem formular a equação correta para solucioná-lo. Os estudantes tentam usar uma série de cálculos seqüenciais aritméticos para solucionar a questão, testando-os com dados informados no problema.

Isso é reflexo do forte conceito aritmético presente em tais alunos, a transição para o estudo da Álgebra não é simples. Inicialmente eles lidam com uma série de cálculos, tendo

como desafio realizar uma cadeia sucessiva de operações, com a devida ordem de ser resolvida. Em um segundo momento, na Álgebra, é preciso fazer relações e simplificar termos, a fim de encontrar valores que solucionem uma condição proposta em uma equação. Trata-se de um paradigma bem diferente para o aluno se adaptar e é importante que seja bem exercitado.

Este processo aritmético feito para solucionar problemas algébricos é considerado procedural. Muitos alunos desejam encontrar um caminho simples para resolver determinada questão, não refletindo em cima do problema proposto. O objetivo do aluno é sentir que sabe o caminho para a resposta e o mais rápido possível começar a trabalhar na resolução do problema. Quando o estudante não entende corretamente a lógica por trás da Álgebra, acaba interpretando cálculos como prioridade e durante o processo de solução pode confundir os conceitos dos valores de incógnita a serem resolvidos.

O uso de representações abstratas, como símbolos algébricos, são considerados concisos e de fácil manipulação, mas distantes de quaisquer objetos reais presentes no mundo do aluno. Representações mais concretas, como descrições verbais de situações, são mais familiares e mais semelhantes aos objetos presentes no dia a dia do aluno, devendo ser explorados no ensino aprendizagem (KOEDINGER, ALIBALI e NATHAN, 2008). É neste cenário que percebe-se que *story problems* trazem melhores índices de acerto por parte dos estudantes, ao comparar-se com equações encadeadas simplesmente, estudo realizado e descrito em (KOENDINGER e NATHAN, 2004).

Em casos específicos, os alunos percebem que com Álgebra podem obter um caminho bem mais curto para solucionar um problema, passando a se interessar mais pelo assunto e buscando aplicar tal conhecimento em outras questões. Estudos foram feitos em escolas no EUA (STACEY e MACGREGOR, 2000), e algo importante foi notado – alunos de ensino particular tendem a usar mais Álgebra para solucionar questões do que os seus colegas de ensino público. Isso reflete as dinâmicas e atividades diferenciadas que algumas escolas particulares utilizam para exercitar os seus alunos. O aluno precisa de estímulo por parte do professor, precisa notar que a Álgebra facilita muitos cálculos e é importante de ser aprendida, pois é usada em outras atividades de estudo que o aluno irá ter, como Física.

Além disso, este estudo (STACEY e MACGREGOR, 2000) apontou que alunos se sentem mais confiantes na resolução de problemas algébricos com o uso de fórmulas ao invés da Álgebra convencional. Em geral tais alunos definem uma variável ‘*x*’ para uma fórmula, que por sua vez, requer uma série de cálculos aritméticos a serem feitos para se resolver corretamente. O fato de não precisar rearranjar certos elementos e simplesmente efetuar

cálculos é o argumento de alguns alunos que tentam expressar a solução dos problemas com fórmulas ao invés da Álgebra convencional. Um fator problemático notado é que os alunos em geral não sabem os valores que devem ser simbolizados como variável desconhecida, ou ainda as suas devidas proporções. Se, por exemplo, temos a seguinte expressão “5x = 10”, deve-se interpretar como “o valor que eu quero descobrir multiplicado por cinco é dez” e isso não é corretamente interpretado por todos os alunos. Outro elemento percebido é que há estudantes que acreditam que o valor de *x* pode ser dois números diferentes ao mesmo tempo. Digamos que na expressão “x + x + x = 14”, eles acham que o primeiro x pode ser um valor, o segundo outro e assim por diante. Este é outro conceito problemático que deve ser bem tratado pelos professores.

Esta utilização da variável ‘desconhecida’, o comum ‘x’ numa equação, também é fruto de outras matérias que o aluno cursa em sua escola. Física, por exemplo, utiliza fórmulas e a partir de um problema proposto o aluno deve ‘montar’ uma, preenchendo os valores de ‘x’ e ’y’ e assim por diante. Passar aos alunos a correta definição é algo a ser trabalhado pelo corpo docente de tais matérias.

Um fator importante na construção da equação é como o aluno interpreta o problema, é determinante no momento, define se tal aluno cria a fórmula ou equação devidamente. Nas explicações é importante que o professor mostre em seus exemplos a construção da equação que vai resolver, passo a passo. Como cada professor tem a sua própria maneira de explicar determinados conteúdos, é importante que ele passe aos seus alunos a forma mais ‘algebricamente’ correta, embora possa ser de mais difícil compreensão no momento. Uma vez absorvido isto, todos estarão aptos a usar métodos mais eficientes dentro da matemática, pois se eleva o nível intelectual da turma em questão. O ponto-chave é, durante as aulas, o professor mostrar que Álgebra é uma ferramenta útil e pode ser compreendida. Para tanto, é necessário que ele passe problemas que não podem ser facilmente resolvidos com outros métodos mais simples, desviando o foco de aprendizado do aluno (CHIAPPINI e LEMUT, 1991).

Em outro experimento (STACEY e MACGREGOR, 2000), um total de doze escolas australianas foi selecionado aleatoriamente para que alguns de seus alunos resolvessem um conjunto de problemas propostos. Em torno de novecentas soluções escritas foram coletadas, referente a alunos de idade entre treze e dezesseis anos destas instituições. Para aprofundar tal estudo, um grupo de alunos foi escolhido para entrevistas individuais, onde explicaram os passos da suas respectivas lógicas. Trata-se de problemas representativos usados nos primeiros anos de estudo da Álgebra, tendo as seguintes características:

- Os problemas são aplicações de equações lineares;
- São situações comuns e de simples compreensão;
- Linguagem usada evita termos mais complexos;
- Operações lidam com números positivos somente;
- Não há valores relativos (como taxas e velocidade);

Os três primeiros problemas abordados levam a uma situação onde a variável a ser descoberta está em um dos lados apenas, considerada pelos autores algo simples. Apenas a última questão envolvia valores de ‘x’ em ambos os lados da igualdade.

Basicamente um terço dos alunos conseguiu responder corretamente as quatro questões, mas muitos deles formularam a equação apenas após responder a questão aritmeticamente ou sequer usaram qualquer princípio algébrico. Em torno de dois terços dos alunos conseguiram respostas numéricas corretas para os três primeiros exercícios, embora não usaram métodos algébricos nem formularam as respectivas equações. A grande dificuldade foi no último, onde como foi citado, havia valores desconhecidos em ambos os lados da igualdade.

A observação dos dados mostrou que há uma grande variedade de métodos e tentativas que foram utilizadas pelos alunos para se chegar a uma solução. Percebeu-se que muitos alunos utilizavam meios puramente aritméticos para tentar resolver os problemas, enquanto outros tentavam na modalidade ‘tentativa e erro’ encontrar valores que satisfizessem os problemas. Por fim, outros utilizaram fórmulas ao invés de equações para chegar a um resultado.

4. AMBIENTES DE APRENDIZAGEM CONCEBIDOS DE ACORDO COM ABORDAGEM MULTIAGENTE

Esta seção contém a análise de dois Ambientes Inteligentes de Aprendizagem, que utilizam agentes em sua arquitetura para alcançar o seu propósito, no caso ensino. Isto vem a ser importante pois a análise de sistemas que utilizam agentes para um determinado fim traz informações a se considerar no trabalho proposto, que se encontra em um STI com características de Sistemas Multiagentes.

A Engenharia de Software evoluiu e passou a definir metodologias diferentes para a construção de sistemas computacionais, com o uso de Agentes Computacionais por exemplo (WOOLDRIDGE, 2002). Com este novo paradigma, no caso a construção de STIs concebidos segundo uma abordagem multiagente, surgiram possibilidades de desenvolvimento de STIs sob aspecto social, mais próximo do ensino real com tutores humanos, gerando esta nova categoria de sistemas considerados atualmente como Ambientes Inteligentes de Aprendizagem (AIA).

4.1 AGENTE

A definição formal de agente ainda não chegou a um consenso na comunidade de pesquisa, no entanto existem descrições bastante abrangentes. Um agente pode ser visto como qualquer entidade que esteja percebendo o ambiente a sua volta através de sensores e agindo através de atuadores (RUSSEL e NORVIG, 1996). É considerado como uma entidade real ou virtual, capaz de interagir em um ambiente e de se comunicar com outros agentes, movida por um conjunto de objetivos individuais ou funções de satisfação a serem otimizadas (REZENDE, 2005). Este agente também é capaz de perceber seu ambiente, mesmo que de forma limitada, além de poder dispor de uma representação parcial do mesmo (REZENDE, 2005). Agir de forma autônoma, decidindo por si só o que fazer para realizar um objetivo ou tarefa é uma característica presentes em agentes computacionais (WOOLDRIDGE, 2002). Estes fatores constituem a sua característica inteligente, visto que um agente opera de forma flexível e racional em diversas circunstâncias percebidas no ambiente (WEISS, 1999).

Estes agentes devem possuir duas características importantes – autonomia e interação (WOOLDRIDGE, 2002). A primeira possibilita ao agente executar ações autonomamente,

com o intuito de alcançar um objetivo, possuindo controle sobre o seu comportamento e podendo agir sem a intervenção de humanos e/ou outros sistemas, de maneira flexível e racional. Isso os dá uma característica inteligente, pois perseguem seus objetivos e executam suas tarefas de maneiras a otimizar algumas características de performance. A segunda característica, a interação, permite a cooperação entre estes agentes para concluir um ou mais objetivos, podendo ser afetados por outros agentes ou até mesmo humanos perseguindo objetivos e realizando tarefas. Poder criar uma lista de objetivos, analisando situações, gerando alternativas de ação e escolhendo as situações que melhor atendem seus objetivos, mesmo que sem a intervenção de demais entidades, são características de agentes computacionais (REZENDE, 2005).

4.2 SISTEMAS MULTIAGENTES

Sistemas Multiagentes são sistemas que são compostos por agentes. Este é um campo da Inteligência Artificial – seu estudo começou na década de 1980 e obteve reconhecimento global apenas nos anos 90, adquirindo interesse ascendente desde então. Isto foi induzido pela crença de que Agente é um paradigma de desenvolvimento de software que fará expandir as possibilidades presentes em grandes sistemas distribuídos, de caráter aberto, como a Internet. Além desta motivação, percebe-se que Sistemas Multiagentes são uma metáfora natural para descrever e construir uma ampla gama de sistemas, conhecidos atualmente como Sistemas Sociais Artificiais, o que justifica o interesse de estudos e desenvolvimentos nesta área (WOOLDRIDGE, 2002).

4.3 METODOLOGIA PROMETHEUS

A construção de sistemas computacionais evoluiu e criou na Engenharia de Software uma linha de pesquisa que trata exclusivamente da utilização de Agentes Computacionais para tal, nomeada Engenharia de Software Orientada à Agentes. Neste trabalho foram estudadas metodologias para desenvolver sistemas sob a perspectiva multiagente, como a Metodologia Prometheus (WINIKOFF, 2004).

A metodologia *Prometheus* é uma abordagem sólida para construção de sistemas multiagentes, sendo consistida de três fases distintas – Especificação do Sistema, Design

Arquitetural, Design Detalhado. Na primeira destas, Especificação do Sistema, os objetivos do sistema em geral são definidos, bem como suas funcionalidades. No Design Arquitetural são definidos os agentes que atenderão a estas especificações iniciais, seguindo para o Design Detalhado, momento em que se arquiteta internamente as funcionalidades de tais agentes. Estão sumarizadas a seguir as atividades que compreendem cada uma destas fases:

Especificação do Sistema:

- Identificar objetivos e sub-objetivos do sistema;
- Criar cenários de caso de uso;
- Identificar a interface do agente com o ambiente em termos de ações, percepções e dados externos;
- Identificar funcionalidades;
- Identificar funcionalidades de dados a serem lidos e escritos;
- Preparar esquemas das funcionalidades (nome, descrição, ações, percepções, dados criados/usados, interações e objetivos)

Design Arquitetural:

- Agrupar funcionalidades para determinar os tipos de agentes;
- Definir tipos de agentes e desenvolver seus descritores;
- Produzir um diagrama de *overview* do sistema descrevendo a estrutura geral do sistema;
- Desenvolver protocolos de interação baseando-se nos diagramas de caso de uso;

Design Detalhado:

- Desenvolver diagramas de processos;
- Produzir diagramas gerais de agentes mostrando processos internos (capacidades, eventos, dados e planos);
- Refinar capacidades internas;
- Introduzir planos para lidar com eventos;
- Definir detalhes dos eventos (externos, entre agentes, entre capacidades, dentro de agentes);
- Definir detalhes dos planos (relevâncias, contextos, sub-objetivos);

O desenvolvedor ao utilizar esta metodologia deve considerar alguns fatores importantes na modelagem do seu respectivo sistema multiagente:

- Informações provenientes do ambiente do sistema são consideradas percepções do agente;
- Mecanismos que podem afetar este ambiente são considerados ações que o agente pode tomar;
- Percepções do ambiente necessitam de processamento por parte do agente para se tornarem um evento;
- Um evento não é considerado igual a uma percepção; trata-se de um conceito de maior importância para um agente. Nem tudo que ocorre no ambiente faz o agente alterar seus planos para alcançar um objetivo;
- Tal processamento deve ocorrer no agente para ser considerada uma metodologia *Prometheus*.

A Figura 5 sumariza os estágios da metodologia, com os tópicos mencionados, retirado de *http://www.cs.rmit.edu.au/agents/SAC2/methodology.html*:

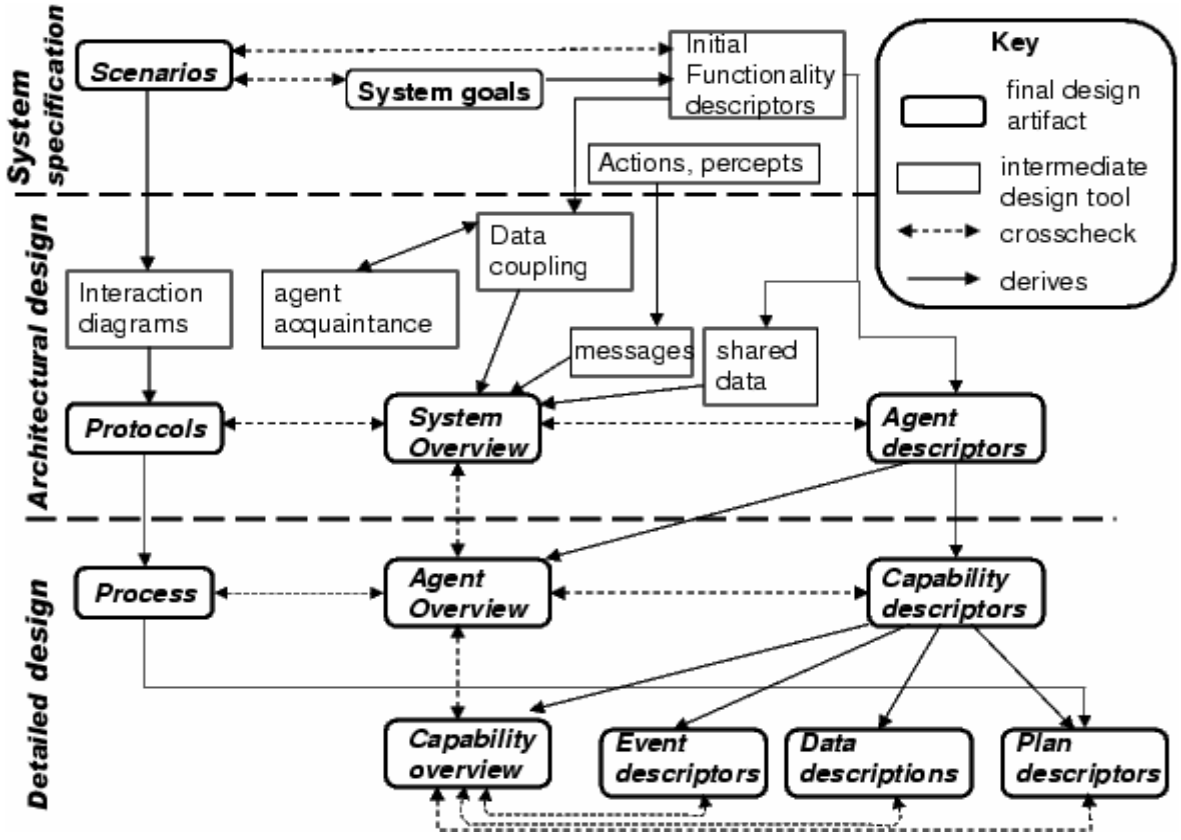

Figura 5: Overview dos Estágios da Metodologia Prometheus

Percebe-se na Especificação do Sistema que:

- Cenários e Objetivos do Sistema embasam a definição inicial das funcionalidades e ações e percepções.

No Design Arquitetural percebe-se que:

- Os cenários da etapa anterior embasam a criação dos diagramas de interação, que por sua vez são base para os protocolos do sistema;
- A definição inicial das funcionalidades servem de base para os acoplamentos de dados e características sociais do agente, contribuindo para a visão geral do sistema, em conjunto com as mensagens e dados compartilhados (fruto da definição de ações e percepções da etapa anterior) e descritores dos agentes;

No Design Detalhado, é possível ver que:

- A definição dos processos é alimentada pelos protocolos definidos na etapa anterior, servindo de base para os descritores de plano e complementando a visão geral dos agentes;
- Esta visão geral dos agentes tem como base os descritores de agente da etapa anterior, contribuindo para visão geral das capacidades e descritores das capacidades dos agentes;
- Estes descritores de capacidades são a base para as definições da visão geral das capacidades, descritores dos eventos, descritores de dados e descritores de planos, que estão inter-relacionados.

Nas seções a seguir são abordados exemplos de Ambientes Inteligentes de Aprendizagem, implementados com arquitetura multiagente para exercer a sua função, que vem a ser o ensino de conteúdos, publicados em eventos internacionais e reconhecidos, por autores que possuem ligação no desenvolvimento de outros STIs. Este estudo foi realizado, pois o Projeto PAT2Math possui esta característica multiagente, presente em AIAs, além de lidar com ensino. Embora PAT2Math possa também ser considerado um AIA, o grupo optou por utilizar a denominação tradicional de Sistema Tutor Inteligente.

4.4 BAGHERA

Baghera (WEBBER e PESTY, 2004) é um sistema multiagente com o objetivo de ilustrar e auxiliar um aluno na resolução de problemas de geometria. Nos estudos de Inteligência Artificial aplicada à educação um padrão de arquitetura foi estabelecido – módulo de conhecimento, módulo de estratégias pedagógicas e modelo de aluno. No entanto, segundo (WEBBER e PESTY, 2004), sistemas que utilizam tais arquiteturas se mostram muito rígidos em certos aspectos. Avaliando tal cenário, na construção do projeto levaram-se em consideração alguns pontos, descritos a seguir.

No desenvolvimento de ambientes educacionais deve-se ter a colaboração entre agentes humanos e artificiais como um princípio fundamental, além da aprendizagem ser o resultado de um processo complexo; não pode ser considerada como o resultado de uma única ação ou estratégia de um agente ou indivíduo. Com base nisso, pode então emergir conhecimento através da interação entre humanos e agentes artificiais.

O ambiente Baghera é uma aplicação *web*, cuja interface do estudante possui uma tela descrevendo um problema que deve ser resolvido. Tal problema é composto por um enunciado e por uma figura manipulável, característica possível graças à tecnologia *Cabri-Java (http://www.cabri-java.net/cabrijava/).* O aluno, através da interface, pode escrever um texto livre que define a sua solução, podendo conter material da biblioteca de teoremas e funções da geometria, presente no sistema. No lado do professor é possível enviar novos problemas aos alunos, além de visualizar e acompanhar as atividades dos mesmos.

O sistema possui uma arquitetura de Sistema Multiagente de dois níveis – alto e baixo. Usam sistema BDI (*Belief – Desire – Intention)* como abordagem de concepção de agentes, o que possibilita a comunicação e tomada de decisões pela observação parcial do estado do ambiente como um todo. O nível alto tem foco nas tomadas de decisão e no comportamento pedagógico do sistema, enquanto que o nível baixo se centraliza no aluno e nas interações feitas pelo mesmo dentro do sistema.

Descrevendo a parte nível alto do aluno, pode-se dizer que três agentes estão presentes. O *agente companheiro* monitora ações do aluno e traz informações do ambiente de aprendizagem para ele. O *agente tutor* é encarregado de propor um conjunto de situações problema adequadas ao aluno em questão, pois se devem considerar as metas educacionais do estudante no atual momento. Por fim, o *agente mediador* atua como um solucionador de problemas, corrigindo exercícios feitos pelo aluno, bem como podendo dar contra-exemplos caso necessário.

Na parte de alto nível referente ao professor, tem-se dois agentes – o *agente pessoal de interface*, que controla o acesso à pasta virtual do professor e traz informações do ambiente; e

o *agente assistente*, que atua como um assistente *pessoal*, porém com o objetivo de ajudar o professor na criação e distribuição das atividades.

Já na arquitetura de baixo nível, o objetivo é diagnosticar as concepções do aluno, os motivos pelos quais ele acertou ou errou determinada questão, por exemplo. De acordo com (BALACHEFF, 2000), se um aluno não acerta uma questão, foi resultado de uma lógica, mesmo que esta seja errada. Há uma razão por trás do caminho que ele fez, deve-se ver o sentido contido nisto. Essa arquitetura é composta por cento e cinqüenta agentes diferentes, cada qual lidando com aspectos diferentes de um problema em potencial. No caso os agentes se dividem entre *Agentes do tipo Problema* (que representam classes de problemas), *Agentes do tipo Operador* (que representam operadores disponíveis ao aluno) e *Agente do tipo Controle* (que representam estruturas meta-cognitivas e estratégias de resolução).

Nos testes realizados com o sistema, onde grupos de alunos entre onze e quinze anos resolveram uma série de problemas geométricos, compararam-se tais resultados com os de grupos de agentes virtuais, que resolveram também tais problemas. Percebeu-se uma grande convergência dos resultados propostos com os construídos por equipes humanas, provando o valor do modelo como um todo. De vinte e oito casos, dezessete a convergência foi absoluta.

4.5 MCOE (MULTI-AGENT CO-OPERATIVE ENVIRONMENT)

MCOE (GIRAFFA, VICARI e SELF, 1998) é um jogo que foi baseado em uma arquitetura de STI, composta por uma sociedade híbrida de agentes que trabalham para alcançar um objetivo em comum: combater a poluição de um lago, resultante de elementos externos (poluidores) e manter o equilíbrio. Os agentes se dividem em duas categorias: reativos (fundo do lago, micro-organismos, água, plantas e três tipos de peixe) e cognitivos (agente tutor, ecologista e estudantes, interpretando personagens).

A mecânica de jogo funciona com dois estudantes jogando em máquinas diferentes. Cada um deles pode selecionar um personagem, entre *Mãe Natureza, Prefeito, Cidadão e Turista.* O jogo então randomiza os elementos que causarão a poluição inicial e será tarefa dos jogadores utilizarem as habilidades do seu personagem para reverter o quadro e estabilizar o ambiente, sendo o objetivo controlar isto por dez minutos.

Entre os sinais que o jogo utiliza para indicar a situação do lago estão: número de peixes, plantas e micro-organismos, transparência e pH da água e aparência do leito. Existe também um visor, chamado de *ecômetro*, que auxilia o jogador a observar as condições do lago. Em caso de visor totalmente verde está tudo balanceado, mas com a perda de equilíbrio

gradualmente a cor é alterada para amarelo e por fim vermelho, disparando um alarme sonoro.

O equilíbrio é baseado no nível de energia, condições predatórias do ambiente e número de elementos restantes no sistema depois do impacto dos poluentes. Os agentes do tipo peixe, plâncton e plantas se reproduzem baseados em função da energia, passando a sua para os descendentes. Se um agente chegar a nível zero de energia, é removido do sistema.

As ferramentas que os personagens têm para agir no ambiente são únicas, atuando de forma diferenciada. Os efeitos podem ocorrer diretamente na poluição ou ter um caráter positivo, mas não direto para o seu combate. A Figura 6 a seguir demonstra a interface do jogo.

Figura 6: Interface MCOE

A cada vez que o jogador realizar uma ação ou se passarem trinta segundos de jogo a simulação e processo de avaliação das condições do equilíbrio são iniciadas, atualizando o número de agentes reativos dentro do sistema.

O ecologista, atuando como tutor, e os dois personagens escolhidos pelos alunos aparecem na parte inferior da interface, com suas respectivas ferramentas para combater a poluição e alterar o ambiente logo abaixo. Este ecologista auxilia os estudantes em combinar estratégias para manter o equilíbrio ambiental utilizando as ferramentas do personagem escolhido.

4.6 AMPLIA – AMBIENTE MULTIAGENTE PROBABILÍSTICO INTELIGENTE DE APRENDIZAGEM

O projeto AMPLIA (FLORES, VICARI, *et al.*, 2003) é um ambiente computacional multiagente, que tem como objetivo apoiar o desenvolvimento do raciocínio diagnóstico e a modelagem de hipóteses diagnósticas, para a formação dos estudantes de Medicina. Tal sistema utiliza modelo de redes bayesianas (JENSEN, OLSEN e ANDERSEN, 1990) para a representação do conhecimento, além de estratégias pedagógicas apoiadas na teoria da construção do conhecimento baseada em argumentações. Elas acontecem durante o processo de negociação pedagógica para a resolução de conflitos entre os agentes do sistema.

O ambiente permite ao aluno construir modelos diagnósticos através do desenho de redes probabilísticas. A abordagem de Redes Bayesianas trata o conhecimento incerto por sua fundamentação em princípios matemáticos. A incerteza é representada por probabilidades e a inferência básica é o raciocínio probabilístico dadas as evidências disponíveis. O enfoque qualitativo é representado pelo conjunto de variáveis e seu relacionamento causal, enquanto o enfoque quantitativo expressa a intensidade desse relacionamento causal.

Revisões de estudos de casos publicados no domínio médico apóiam a hipótese de que um médico, quando realiza um diagnóstico, implicitamente realiza raciocínio probabilístico. Existem evidências empíricas de que o raciocínio probabilístico, como realizado pelas redes bayesianas, é similar aos padrões do raciocínio humano. Estes sistemas podem ser usados nos processos de ensino/aprendizagem e prática diária nos consultórios.

O AMPLIA compara seu modelo de domínio com o modelo construído pelo aluno. Se estes modelos forem diferentes a ponto de se gerar um conflito entre o aluno e o ambiente, é iniciado um *processo de negociação* baseado em estratégias pedagógicas, de forma a induzir o aluno a revisar seu modelo. Essa revisão pode levar à modificação, ou não, de seu modelo, dependendo da avaliação dos argumentos repassados ao aluno.

No ambiente AMPLIA, além dos agentes humanos professor e aluno, existem os seguintes agentes artificiais: *Agente Aprendiz*, *Agente Mediador* e *Agente de Domínio*.

- *Agente Aprendiz* – representa o aluno no ambiente. Este agente possui informações tais como nome do aluno, *login* e senha de acesso, modelo de rede construído, e sua auto-avaliação. Também possui informações extraídas da grade curricular (posicionamento e desempenho no curso) e registro do *log* das ações do aluno durante a modelagem de sua rede hipotética.

- *Agente de Domínio* – compara a rede modelada pelo especialista com o modelo construído pelo aluno, identificando os pontos de conflito. O resultado desta análise é encaminhado para o *Agente Mediador*.
- *Agente Mediador* toma decisões sobre como e quando intervir no processo de construção do modelo de rede do aluno, após ter identificado um conflito ou impasse, mas também pode agir mediante uma solicitação do aluno. A função principal do *Agente Mediador*, entretanto, é selecionar a estratégia pedagógica mais apropriada, possibilitando a resolução de conflitos que podem ocorre entre os *Agentes de Domínio e Aprendiz*.

4.7 MATHEMA

O ambiente MATHEMA (COSTA, LOPES e FERNEDA, 1995; COSTA, PERKUSHI e FIGUEIREDO, 1996) é uma proposta de ambiente para resolução de problemas através da cooperação entre humanos e uma comunidade de agentes virtuais inteligentes. A arquitetura do ambiente é composta por cinco elementos: Um designer, não podendo ser o aluno, uma Sociedade de Agentes Inteligentes - SAI (que podem cooperar entre si para realizar tarefas de resolução de problemas), uma Sociedade de *Experts* Humanos – SEH (que atua como fonte de conhecimento para a Sociedade de Agentes Inteligentes), um Supervisor (entidade que é encarregada de selecionar agentes para tarefas e um Agente de Comunicação, responsável por fazer a intermediação entre a SAI e a SEH. A ‘Figura 7: Arquitetura do Ambiente MATHEMA’ mostra esta arquitetura.

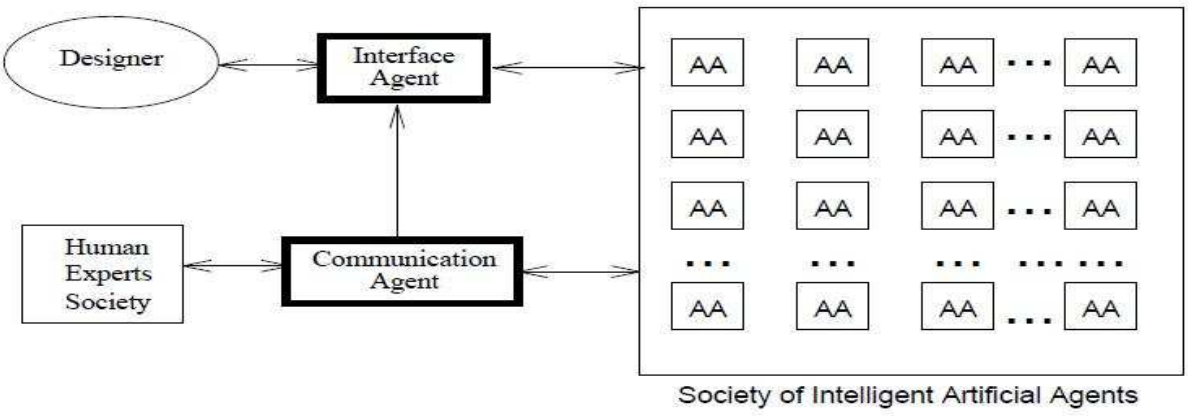

Figura 7: Arquitetura do Ambiente MATHEMA

Para que exista uma cooperação efetiva, o Agente de Interface precisa reconhecer os objetivos do Designer em primeiro lugar, para promover um trabalho cooperativo baseado no que o Designer quer fazer. Assim o Agente de Interface seleciona um agente para ser Agente Supervisor. Este será o responsável pela comunicação entre a SAI e o Designer. Conforme o problema for mais complexo, o Agente Supervisor aciona mais e mais agentes da SAI para trabalhar no problema proposto pelo Designer, sendo o pior caso possível todos agentes ocupados com esta tarefa. Neste cenário, o Agente Supervisor informa ao Designer que o problema não pode ser resolvido e repassa-o para a SEH.

Sociedade de Agentes Inteligentes

A Sociedade de Agentes Inteligentes é composta por um conjunto de agentes inteligentes que podem cooperar entre si nas atividades de solução de problemas. O propósito principal desta sociedade é auxiliar o designer (usuário) a resolver o problema que está trabalhando. Cada um dos agentes é uma entidade inteligente pertencente a um determinado domínio ou em parte dele, podendo contribuir em um comportamento social através de suas habilidades e *expertise.*

As características destes agentes compreendem:

- Solução de Problemas: Resolver ou auxiliar na resolução de problemas propostos pelo Designer.
- Aprendizado: O agente aprende sobre os demais agentes na sua sociedade (um aprendizado social), além de aprender com o designer ou ainda com o SEH.
- Resolução de Conflitos: Um agente pode não saber resolver um problema, ou saber apenas parte da resolução. Neste caso dois cenários podem acontecer:
    - (i) O agente conhece vários agentes que podem resolver o problema em questão; Então este precisa decidir quem fará isto.
    - (ii) O agente não conhece nenhum agente que pode resolver o problema; Nesse caso, ele transmite para todos a sua situação e avaliando quem pode lhe ajudar. Em caso de nenhum agente estar apto a resolver, o problema é repassado a SEH.
- Resolução de Discordâncias: Situação em que o conhecimento dos agentes difere do designer ou da SEH.

- Cooperação e Comunicação: Comunicação com outros agentes, com o designer e com a SEH.

Modelo de Agente

O modelo de agentes do MATHEMA se baseia em um modelo de agente cognitivo descrito em (COSTA, LOPES e FERNEDA, 1995; COSTA, PERKUSHI e FIGUEIREDO, 1996), pois é inspirado em uma organização social baseada em suas interações. Existem duas camadas neste modelo: A camada de cooperação e a cada de solução de problemas.

A camada de cooperação representa o comportamento social do agente. É responsável pela coordenação das atividades cooperativas e internas do agente. Para concretizar isto, cada agente possui módulos de comunicação, coordenação, planejamento, controle e conhecimento pessoal e social.

A camada de solução de problemas é responsável pela atividade cooperativa de solução de problemas. Sua idéia básica é estabelecer uma cooperação entre o designer e o agente, guiando o mesmo durante o processo de resolução de um problema, avaliando eventuais erros, misconceptions ou quando o designer solicitar auxílio.

5. WEB SEMÂNTICA

Atualmente, existem variados cursos no Brasil oferecidos na modalidade “Educação à Distância” em que a Internet serve como a plataforma base de comunicação e interação entre alunos e professores (ISOTANI e BITTENCOURT, 2009). Com a necessidade de ofertar maneiras mais eficientes de ensino na Web, duas linhas de pesquisa estão em crescente expansão: a primeira delas é a Web Semântica, que desenvolve tecnologias que permitem ao computador compartilhar e manipular as informações contidas na Web de forma adequada e inteligente. Com a utilização desta, tanto ambientes de aprendizagem como agentes computacionais podem interagir entre si, trocando informações e auxiliando professores na seleção, combinação e classificação do conteúdo disponível na Web. A segunda área em expansão é a Web 2.0, ou Web Social, em que usuários podem compartilhar e construir conhecimento de forma simples, interativa e colaborativa (ISOTANI e BITTENCOURT, 2009).

As interligações no ciberespaço, na Internet, parecem ser limitadas, não tirando proveito do potencial contido nos significados disponíveis em suas páginas, documentos e mídias. O que existe atualmente são ligações relacionando os atributos físicos destes artefatos presentes na *Web*, não seu real significado. Isto deveria constituir uma das variáveis de tal interligação, e não o todo como acontece atualmente. Entre as alternativas para extrair relações e conceitos de toda essa massa de informações existem ferramentas como as Ontologias (JACOB, 2003). A informação na Web é representada em linguagem natural, sendo compreensível por pessoas. No entanto, para prover informação de maneira que computadores ou agentes computacionais possam a compreender, extraindo seu significado, é necessário representar ela formalmente e de maneira sistemática (ISOTANI e BITTENCOURT, 2009).

A *Web Semântica* (LEE, HENDLER e LASSILA, 2001) constitui um paradigma de uma Web interligada através dos significados e sentidos destas informações e arquivos. Promissora e foco de muitas pesquisas, alavancou o interesse em projetos que utilizassem o significado das informações na sua estrutura e/ou objetivo – seja recomendar um produto para um cliente em uma loja virtual ou ensinar um dado conteúdo para um aluno. Esta introduz a nova geração de tecnologias cujo objetivo é representação da informação de maneira que computadores sejam capazes de interpretá-la. Além disso, através desta representação, pesquisas nesta área propõem tecnologias para automação, integração e reuso da informação, considerando diferentes plataformas de desenvolvimento, sistemas operacionais, protocolos de rede, e outras variações de tecnologia (ISOTANI e BITTENCOURT, 2009).

Para se tornar realidade, esta nova Web exigiria uma estrutura condizente, cuja semântica pudesse ser interpretada por computadores e humanos ao mesmo tempo – neste cenário as Ontologias ganharam destaque (GRUBER, 1995; GUARINO, 1995). Conhecidas pela possibilidade de modelar diferentes domínios e suas relações, ao mesmo tempo possuindo uma estrutura visual simples como a de um Mapa Conceitual, as Ontologias fizeram seu espaço e se tornaram a força motriz da nova *Web* que está se formando, oferecendo uma linguagem expressiva e formal para gerar informação que possa ser interpretada por computadores (MIZOGUCHI, HAYASHI e BOURDEAU, 2007), e podendo serem combinadas, compartilhadas, modificadas e utilizadas para definir semanticamente diferentes categorias de recursos (JOVANOVIC, TORNIAI, *et al.*, 2008). Porém, criar ontologias ou taxonomias, anotando o conteúdo de forma estruturada é um processo complexo. Devido a isso, a Web Semântica ainda encontra dificuldades para ser adotada em cenários menos "acadêmicos" (ISOTANI e BITTENCOURT, 2009). Segundo (DEVEDZIC, 2006), o uso de ontologias e o desenvolvimento de serviços Web para processar a informação disponível na Internet está transformando a Web da informação (Web tradicional) na Web do conhecimento.

No contexto educacional, a Web Semântica e Ontologias têm sido utilizadas para resolver diversos problemas encontrados nos atuais ambientes educacionais baseados na Web (ISOTANI e BITTENCOURT, 2009). Problemas como dificuldade no compartilhamento e reutilização de material didático, busca por conteúdos e gerenciamento de repositórios de objetos de aprendizagem, podem ser resolvidos melhor utilizando e adaptando os conceitos da Web Semântica (ISOTANI e BITTENCOURT, 2009). As aplicações de ontologias no cenário atual da informática no ensino estão crescendo. A seguir é citada a questão de contextos em AVAs.

A personalização da apresentação e relação de conteúdos possui uma outra vertente que utiliza largamente ontologias – nomeada contextos (EYHARABIDE, 2009). Contexto pode ser definido como a descrição de aspectos de uma situação, sendo vital para aprimorar acesso personalizado e apresentação de conteúdos em ambientes E-Learning. Enquanto uma entidade é uma coisa que existe por si só, o contexto é o que pode ser dito sobre esta entidade. Pesquisas em hipermídia educacional adaptativa provam que considerar tal contexto leva a uma melhor compreensão e personalização para aluno. Para ser efetivo, um processo de aprendizado deve ser adaptável para o contexto do estudando, cenário em que as ontologias são muito úteis, pois removem ambigüidades e também ajudam a identificar categorias semânticas de um domínio particular. Podem ser usadas não apenas para modelar informações

do domínio, mas personalizar também serviços aos usuários em sistemas adaptativos. O cenário em que um estudante de EaD (Educação à Distância) deseje acessar uma aula disponível no ambiente virtual da sua universidade retrata uma possibilidade de aplicação deste conceito. Se o aluno está na biblioteca da Universidade, os materiais propostos já podem sugerir leituras que estejam próximas ao aluno em questão. Além disso, se o aluno está acessando do seu celular o ambiente, as imagens utilizadas podem ser menores, que se adaptem melhor a tela que o estudante tem em seu aparelho. Se a conexão com o ambiente estiver lenta, este pode oferecer somente arquivos de áudio ao invés de vídeos em uma lição, por exemplo.

5.1 ONTOLOGIAS

Ontologia (com 'O' maiúsculo) é a área da Filosofia que estuda a natureza da existência e a estrutura da realidade, atuando como uma especificação explícita de uma conceitualização (GUANGZUO, 2004). Já o termo ontologia (com 'o' minúsculo) se resume a categorização de tipos de elementos, relacionando estes com domínio estudado. Trata-se de um termo frequentemente utilizado para se referir a compreensão semântica – a estrutura conceitual do conhecimento – compartilhada por pessoas que participam em dado domínio. Em sistemas de Inteligência Artificial o que existe de fato é o que pode ser representado (GUANGZUO, 2004), já em um ambiente da *Web*, no entanto, uma ontologia não é simplesmente um *framework* conceitual, mas sim uma estrutura concreta e sintática que modela a semântica de um domínio, em linguagem de máquina, compreensível por ela (JACOB, 2003).

A *web* está evoluindo de um grande espaço de informação e comunicação para um repositório massivo de conhecimento e serviços. A ferramenta que permite esta mudança são as ontologias, descritas como conceituação sobre um domínio (ZHOU, 2007). Atualmente a *Web* é estruturada para lidar com pessoas, termos de domínio e ainda *tags* HTML que são claramente compreensíveis por desenvolvedores e usuários, no entanto não possuem sentido algum para sistemas computadorizados, aplicações e/ou agentes. A estrutura XML vem mudando este conceito, porém seu *layout* lida primariamente com estrutura física dos documentos, além de sofrer com a falta de semântica presente em suas *tags,* que poderiam ser o meio de compreensão computacional (JACOB, 2003).

Ontologias fornecem um embasamento semântico sobre descrições compreensíveis por máquina de conteúdo digital, sendo considerada ubíqua em sistemas de informação

através de documentos com meta-dados, aprimorando a performance de extração de informação e inferência, tornando dados entre diferentes aplicações inter-operáveis. No campo de SMAs, pode-se dizer que se, comportamentos e serviços forem descritos semanticamente por ontologias, há como estes agentes computacionais cooperarem melhor para alcançar seus objetivos (ZHOU, 2007).

Uma ontologia é construída para facilitar o compartilhamento de conhecimento e reutilização do mesmo entre pessoas e agentes computacionais. Abstraindo, linguagens de representação de ontologias formam um espectro que varia entre a linguagem natural em um extremo até a linguagem formal no outro. Estes extremos se aplicam melhor a pessoas e agentes virtuais, respectivamente. Para diminuir o *gap* entre eles, pode-se permitir que pessoas acessem e modifiquem ontologias utilizando linguagem natural, mapeando esta em uma série de conceitos de baixo nível para que o processo fique automatizado. Isto envolve e aprimora o *aprendizado em ontologias*, processo em que técnicas de aprendizado de máquina são utilizadas para extração de conhecimento (ZHOU, 2007). Conforme estas alterações fazem com que o conteúdo dentro da ontologia evolua, estas se tornam tarefas centrais na sua manutenção. Neste cenário, a reutilização de ontologias e seu princípio de utilizar novamente conhecimentos já mapeados, tem o seu maior valor.

5.2 RESOURCE DESCRIPTION FRAMEWORK (RDF) E RDF VOCABULARY DESCRIPTION LANGUAGE SCHEMA (RDFS)

Embora a hierarquia não seja uma característica determinante e sempre presente em ontologias, se trata de um importante componente no seu modelo de representação, formalmente definido pelo *Resource Description Framework* (RDF) e RDF *Vocabulary Description Language Schema* (RDFS). Tais modelos suportam reuso de elementos de qualquer ontologia ou esquema *metadata* que pode ser identificado por um URI (*Uniform Resource Identifier)* (JACOB, 2003).

RDF define um modelo e conjunto de elementos para descrever recursos baseados em propriedades nominais e valores. Adicionalmente ele provê uma sintaxe que permite a qualquer comunidade de descrição de recursos criar uma representação específica do domínio com as suas respectivas semânticas, além da incorporação de elementos de distintos esquemas *metadata*. Tanto o modelo como a sintaxe permite a sua utilização para codificar informações em linguagem de máquina, para a posterior troca de dados entre aplicações e processamento

da semântica. O RDFS adicionalmente complementa e estende o RDF definindo uma linguagem de máquina declarativa e processável, que pode ser usado para formalmente definir uma ontologia ou conjunto de classes e suas respectivas propriedades. De forma cooperativa, fornecem um modelo sintático e estrutura semântica para definir ontologias *machine-processable* e esquemas *metadata*, além de permitir interoperabilidade de estruturas representativas entre conjuntos heterogêneos de recursos (JACOB, 2003).

Uma ontologia RDFS é diferente das taxonomias e estruturas de classificação clássicas, no entanto no topo dos níveis da hierarquia – classe *recurso* e suas filhas *classe* e *propriedade* – não são determinadas pelo domínio da ontologia, e sim pelo esquema RDFS (JACOB, 2003). Cada elemento presente nesta ontologia é um tipo de *classe* ou *propriedade*. Relações entre estas são potencialmente multi-hierárquicas, ou seja, uma classe podendo ser filha de uma, ou mais classes superiores.

Comparando tais estruturas com as ontologias RDFS percebe-se que a diferença mais significativa consiste em que estas últimas definem conjuntos de elementos que podem receber valores, para representar características físicas e conceituais de um determinado recurso, ao contrário das estruturas classificativas simples, que lidam com entidades que são atribuídas a classes hierárquicas do sistema, não podendo também lidar com regras de inferência sobre o conteúdo.

Uma ontologia não pode ser considerada como uma taxonomia, um esquema classificativo ou ainda um dicionário. Trata-se de um sistema representativo único, que integra em uma única estrutura as características de vários meios de representação, como os três últimos citados. A ontologia fornece a semântica básica para esquemas *metadata* e facilita a comunicação entre sistemas e agentes através de um modelo conceitual de comunidade de usuários padronizado. Tais fatores descritos fornecem o fundamento conceitual que torna o objetivo da Web Semântica possível (JACOB, 2003).

5.3 OWL (ONTOLOGY WEB LANGUAGE)

A linguagem OWL é o padrão para descrição de ontologias, utilizado pela comunidade que trabalha com a Web Semântica. Sua característica principal é expressividade de alto nível e inferência. Este padrão possui três subcategorias, conforme descrito a seguir (RICHARDSON, 2006):

OWL-Lite: Possui pouca expressividade, mais utilizada em taxonomias e classificações hierárquicas, definindo restrições e cardinalidade simples entre classes.

OWL-DL: Esta subcategoria garante um poder maior de expressividade que a OWL-Lite, possibilitando criações mais complexas com a utilização de lógica descritiva. Nela estão contidas características para relação de instâncias de classes como propriedades funcionais, inversas, transitivas, entre outras relações importantes em inferências.

OWL-Full: A mais poderosa em questão de expressividade, trabalhando com as ferramentas de expressividade da OWL-DL e construtores RDF juntamente. No entanto, tal subcategoria não garante a característica de decidibilidade que a OWL-DL traz ao desenvolvedor.

A OWL visa tornar os recursos na *Web* mais simples de serem interpretados por processos automatizados, através da adição de informação sobre os recursos que descrevem ou provém o próprio conteúdo. Como a Web Semântica tem caráter distribuído, a OWL deve permitir que a informação seja reunida de fontes separadas, fato que acontece com os relacionamentos entre ontologias, que trabalham com a importação de informações de outras ontologias presentes na Web.A estrutura da OWL permite também que fatos e descrições sejam estendidos, mas não apagando e invalidando informação prévia. Uma classe A definida na ontologia A1 pode ser estendida por outras ontologias, mas estas novas informações não podem invalidar o que foi proposto anteriormente.

As categorias OWL anteriores foram avaliadas para orientar a construção da Ontologia utilizada para definir os Planos de Ensino do Agente Tutor, sendo selecionada a categoria OWL-DL que possibilita a expressividade de relações entre conteúdos do sistema, habilidades e misconceptions do aluno, permitindo tomar decisões através disto.

6. TRABALHOS RELACIONADOS

Esta seção tem como objetivo mostrar trabalhos com objetivo semelhante ao trabalho proposto, que é ensino de Álgebra, dentro do projeto PAT2Math. Aqui serão abordados quatro Sistemas Tutores Inteligentes cujo conteúdo abordado para ensino reside também no tema da Álgebra. São sistemas que se destacam pelas suas arquiteturas e estratégias pedagógicas para ensino, publicadas em diferentes conferências de grande relevância internacional.

6.1 ACTIVEMATH

O *ActiveMath* (SIEKMANN e MELIS, 2004) é um Sistema Tutor Inteligente (STI) que se baseia em conteúdos adaptados ao aluno e suas características cognitivas. Existem STIs que não consideram isto devido a restrições da aplicabilidade de conteúdo, em que todos os alunos devem aprender um conteúdo para se usar em um determinado contexto somente. Em contrapartida, na Matemática por exemplo, um mesmo conteúdo pode ser aplicado de formas distintas, para soluções em diferentes áreas do conhecimento. A abordagem pedagógica do sistema consiste na livre interação pelas lições propostas, para a construção do aprendizado.

A arquitetura do *ActiveMath* tem como princípio básico a separação de conteúdos e funcionalidades, bem como a devida diferenciação entre tipos de conhecimento. As regras pedagógicas de ensino são armazenadas em uma *Base Pedagógica de Regras*, e as informações dos alunos armazenadas em um *Modelo de Estudante*. Tal princípio facilita questões de reutilização e manutenção do modelo, além de estruturá-lo como um todo para a *web*. Os conhecimentos presentes no sistema são tratados como objetos de aprendizado, o que traz alguns benefícios, tais como serem mais compreensivos e poderem ser interligados com outros saberes presentes no sistema.

O sistema é baseado em uma interface cliente-servidor, em que o estudante (cliente) somente precisa utilizar um navegador para interagir com o sistema, tornando-o independente de plataforma. Uma vez dentro do sistema o aluno pode selecionar os conceitos que deseja aprender, constituindo o chamado *cenário de aprendizado*. Tal informação é enviada para o

gerador de cursos, encarregado de contatar a base de conhecimentos matemáticos. Esta, por sua vez, identifica quais os pré-requisitos são necessários para que o aluno possa aprender o que deseja, afim de que o módulo de *regras pedagógicas* estruture a ordem das lições a serem exibidas, bem como os respectivos exemplos e exercícios, se baseando nas habilidades do aluno, armazenadas no Modelo de Estudante. A sua interface é representada através da Figura 8.

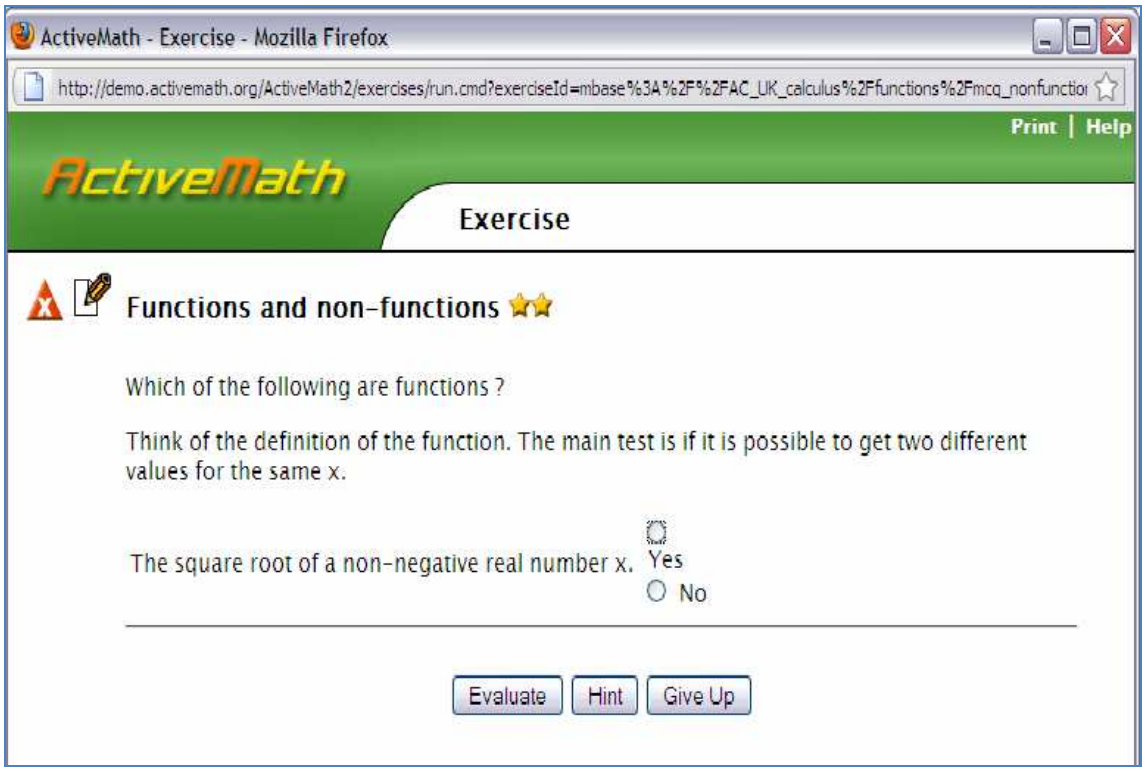

Figura 8: Interface ActiveMath

O objetivo do sistema não é necessariamente guiar o estudante ao longo de uma resposta pré-definida, a que seja mais correta e eficiente existente. Na verdade é avaliar os passos do aluno e sugerir uma linha de ações, caso estes passos não estejam dentro de um escopo que o levará a solucionar o problema. O sistema se adapta ao usuário em certos aspectos, como idioma e demais preferências de aprendizado do aluno, ou seja, seu perfil. No seu *histórico* residem as ações do aluno dentro do sistema como um todo, possibilitando análises de atenção e tempo de leitura por exemplo, úteis na construção de um plano de ensino condizente com as capacidades do estudante. Existem mecanismos internos que são atualizados conforme o aluno interage dentro do sistema, como informar um resultado correto

em um exercício por exemplo. Isso é considerado importante, pois os alunos tendem a verificar o seu progresso e histórico.

O modelo de estudante é composto por informações estáticas, compreendendo preferências de ensino por exemplo, e dinâmicas, que abrangem os dados de comportamento e proficiência do aluno nos conteúdos propostos. O conhecimento do aluno como um todo foi dividido em três partes – *conhecimento geral, compreensão* e *aplicação.* Dependendo do que o aluno realiza, um determinado parâmetro dos três últimos citados é atualizado. Se um determinado aluno leu um capítulo, *conhecimento geral* é atualizado; acompanhar exemplos atualiza *compreensão;* fazer alguns exercícios atualiza *aplicação.*

A representação do conhecimento do sistema é definida em linguagem semântica XML OMDoc, que é uma extensão do OpenMath. Esta estrutura proporciona uma coleção de objetos juntamente com uma gramática para representação de objetos matemáticos e conjuntos de símbolos padronizados, lidando com objetos ao invés de uma pura sintaxe. As relações entre tais objetos se enquadram sempre como *definição, teorema e prova* (LIBBRECHET e ULLRICH, 2003).

As regras pedagógicas do sistema (MELLIS, ANDRÈS, *et al.*, 2001) contém *know-how* de qual conteúdo apresentar, bem como de que maneira fazer isto e em que condições habilitar determinados serviços no ambiente. Quando um conteúdo é apresentado, o sistema verifica alguns pontos:

- Quais informações adicionais devem ser apresentadas com os conceitos;
- Quais exercícios e exemplos mostrar ao aluno;
- Qual ordem as informações devem ser apresentadas.

Eventos que o aluno pode disparar no sistema, tais como terminar um exercício ou ir para outra página de lições, ativam mecanismos que atualizam o seu modelo do estudante. O sistema utiliza estratégias pedagógicas para maximizar o aprendizado por parte do aluno, além de empregar técnicas de Inteligência Artificial sobre como e o que apresentar, utilizando Taxonomia de Bloom (1956) para criar uma estrutura que representa o quão apto um aluno está em um dado conteúdo, assim podendo criar um cenário para este ensino, com objetos de aprendizagem que respeitam a Teoria de Design Instrucional de Merril (2001).

O sistema não permite uma interação muito complexa com o usuário, sempre oferecendo opções pré-definidas de respostas para o aluno selecionar e validar se está correto. Isso parece ser reflexo da ausência de um modelo cognitivo detalhado, com regras para

resolução de exercícios que tratem os *inputs* do usuário de maneira flexível. O sistema está disponível para acesso na Internet no seguinte endereço: http://www.activemath.org.

6.2 ANIMALWATCH

AnimalWatch (COHEN, BEAL e ADAMS, 2008) é um Sistema Tutor Inteligente que integra o aprendizado de Matemática, envolvendo problemas algébricos, com o contexto de Ciências. Possui uma abordagem diferenciada, pois os problemas propostos ao estudante lidam com espécies de animais em extinção, fornecendo uma visão de como a Matemática é utilizada em problemas da vida real, fato questionado por muitos estudantes durante o seu aprendizado.

A solução de problemas matemáticos, a partir de formulações textuais, é considerada complexa, pois envolve a habilidade de compreender o que o problema propõe, construir equações a partir disto, calcular um resultado e avaliá-lo sob aspectos de precisão e plausibilidade no contexto. Para esta questão, o sistema dispõe de mais de mil problemas, organizados sob a forma de narrativas sobre espécies em extinção, incluindo introdução e gráficos explicativos, propondo um exercício ao final do texto.

Um professor de Matemática geralmente lida com várias turmas de alunos, tornando difícil dar um acompanhamento individual ao seu aprendizado, caso considerado melhor para o desenvolvimento cognitivo do aluno em relação a um conteúdo. No sistema este acompanhamento individualizado do aluno se concretiza com a apresentação de diferentes problemas ao aluno baseando-se nas suas habilidades, o que foi estudado e resolvido dentro do sistema, sendo dinamicamente alterados entre as sessões de ensino realizadas.

O AnimalWatch se concentra em w*ord problems,* ou seja, problemas textuais, que exigem boa leitura e interpretação para que o aluno possa saber como organizar sua resolução. O sistema está apto a oferecer ajuda em qualquer momento da resolução de um exercício se solicitado, assim como quando um passo errado é realizado no desenvolvimento deste. Clicando sobre o botão de ‘dicas’, um menu aparece mostrando tipos de ajuda para consulta, como textos explicativos, vídeo-aula sobre um assunto ou ainda um exercício semelhante resolvido. Dúvidas e erros por parte de um aluno afetam o seu respectivo Modelo de Estudante, fazendo o sistema testar periodicamente os conteúdos que foram exercitados em lições anteriores.

Além de testar os conhecimentos do aluno com os problemas propostos comentados anteriormente, o sistema possui um módulo de testes para habilidades matemáticas básicas. Este módulo lida com expressões matemáticas simples que exigem uma confirmação, questionando o aluno sobre sua veracidade. Como são relativamente simples, fazem o aluno alcançar pontuações altas e se sentir motivado a resolver os demais problemas dentro do sistema, além de permitir que o seu processo cognitivo se concentre em atividades mais importantes, como interpretação do problema e possíveis respostas para ele, em um dado contexto. A sua interface pode ser visualizada na Figura 9.

Figura 9: Interface AnimalWatch

A modelagem de estudante segue um modelo estatístico, através de Modelos Ocultos de Markov de engajamento. O sistema foi desenvolvido para suportar modelagem estatística do comportamento dos estudantes enquanto lidam com o ambiente, reunindo informações a cada momento sobre aspectos observáveis dos alunos, como sequência de problemas vistos no sistema, cliques em respostas e requisições de ajuda e seus respectivos intervalos de tempo, além do tempo entre uma solicitação de novo problema e outro. Um sistema de máquina de estados é enquadrado às ações do aluno, permitindo identificar abuso das solicitações de ajuda, ou seleção aleatória de respostas (BEAL, QU e HYOKYEONG, 2006). Isto permite que o sistema organize um objeto com tais informações a cada instante de tempo relevante para avaliação posterior.

O AnimalWatch é projetado para ajudar os alunos a desenvolver proficiência em temas que ainda não tenham pleno domínio, com base em um currículo de competências com trinta habilidades matemáticas. A sequência de problemas específicos apresentadas a um aluno é personalizada para o seu nível de proficiência, que naturalmente sofre mudanças durante as sessões com o tutor. Os tópicos de Matemática e a dificuldade dos respectivos problemas são aumentados conforme o aluno demonstra que pode resolver problemas que envolvem uma habilidade em particular. As competências que foram dominadas são periodicamente revisadas, mais especificamente, se um aluno comete erros em um problema que envolve certas habilidades matemáticas, a probabilidade de selecionar um problema que envolve as habilidades anteriores a esta aumentará. Assim, o sistema se concentra nas áreas que cada aluno precisa de mais prática.

Como alguns estudantes preferem ajuda de meios tecnológicos ao invés dos tradicionais, como um professor ou colega, os recursos do sistema, em especiais os multimídia, oferecem uma alternativa aqueles que relutam em pedir auxílio, por exemplo. No entanto, a resolução se limita a entrar com a resposta de problemas pontuais ao fim de cada contexto, de cada texto que o sistema apresenta. A resolução exige um raciocínio que se concretiza no papel, gerando uma resposta final que deve ser inserida no sistema, para avaliar e deduzir o que o estudante fez, a fim de oferecer ajuda adequada.

6.3 APLUSIX

O Aplusix (NICAUD, BITTAR, *et al.*, 2006; NICAUD, BOUHINEAU e CHAACHOUA, 2004) é um sistema que auxilia alunos no aprendizado de Álgebra, mais precisamente no campo de cálculos numéricos, expansões, fatorações, equações e sistemas compostos por elas. Faz parte dos trabalhos desenvolvidos pelo grupo MeTAH (Disponível em: http://www.noe-kaleidoscope.org/group/metah/), em que pesquisadores e docentes estudam a concepção e desenvolvimento de ambientes informatizados focados em aprendizagem. Está vinculado ao LIG (*Laboratoire d'Informatique de Grenoble*), cujo site encontra-se em: http://www.liglab.fr/.

O Aplusix geralmente é instalado no servidor principal de escolas e acessado pelos estudantes remotamente nos seus respectivos laboratórios de informática. Permite que o professor crie e gerencie classes no sistema, bem como as contas de estudante, através da sua interface de administrador. As ações dos estudantes dentro do sistema são gravadas para que o professor possa acompanhar os passos realizados, de forma global com estatísticas ou

mediante um *replay* de cada passo realizado. O sistema possui uma série de quatrocentos exercícios, agrupados em categorias estruturadas como *mapas*, que possuem caráter aleatório em variáveis-chave, individualizando a instanciação de cada um deles. Tais mapas são grupos de exercícios, cadastrados no sistema por professores para que o mesmo exija a sua conclusão por parte do aluno antes de prosseguir em um determinado conteúdo. Na resolução de tais exercícios se encontra uma característica de destaque do Aplusix, que possui um editor, capaz de modificar e visualizar expressões algébricas complexas, oferecendo funções de edição (copiar, colar, recortar) para a realização dos passos de resolução dos exercícios.

O conhecimento no sistema se baseia em regras para resolução de problemas algébricos, garantindo uma interação livre do usuário para editar cada passo da resolução. Isso é alcançado com um sistema de *drag and drop*, ao invés de comandos formais que disparam regras simples. Assim, o estudante pode aprender com erros gerados pelas suas ações, não simplesmente por selecionar uma regra que não se aplicasse, a ajuda do sistema é mais pontual desta forma.

Ao fim do exercício realizado, o sistema verifica se uma solução aceitável foi alcançada. Em caso negativo, traça os passos realizados anteriormente informando até que ponto estava correto o andamento do exercício. A sua interface está representada na Figura 10.

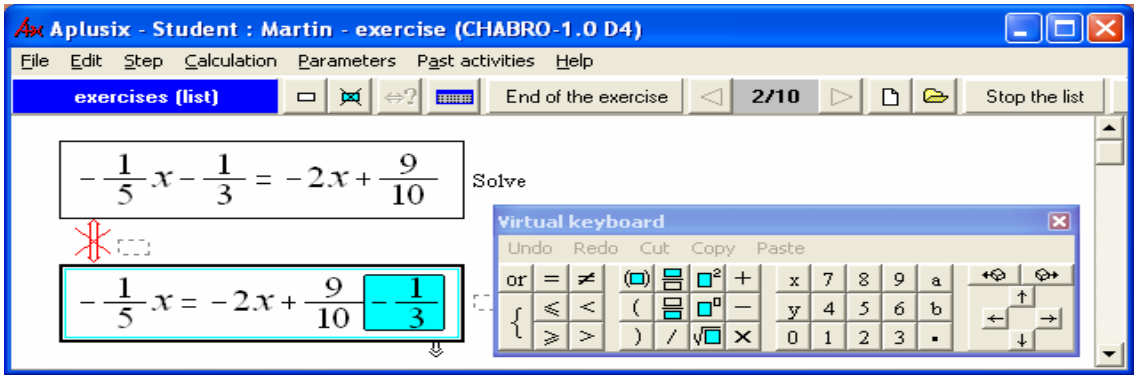

Figura 10: Interface Aplusix

Para a interação com os estudantes são oferecidos certos modos, conforme segue abaixo:

Modo Treinamento: É mostrado o resultado de cada passo realizado da resolução do exercício, se está correto ou não, até o resultado final.

Modo Teste: Não existe *feedback* das ações, somente ao final do exercício é informado ao estudante uma pontuação;

Modo Observador: Acessível ao estudante e professor, este modo permite visualização

completa da resolução e dos estados finais possíveis para um determinado exercício.

O sistema possui uma característica intuitiva para a resolução de problemas algébricos, o que motiva os estudantes a tentarem estratégias diferentes para tal de forma simples. Os fatores aleatórios, provenientes dos *mapas* de exercícios comentados anteriormente, reduzem o trabalho de professores na criação de exercícios de fixação, tornando este sistema uma alternativa interessante e rápida para mudar o estilo de ensino em sala de aula sobre Álgebra. Testes do Aplusix foram realizados em países com características distintas, reportados em (NICAUD, BITTAR, *et al.*, 2006), trazendo resultados satisfatórios.

6.4 THE ALGEBRA TUTOR

O *Algebra Tutor* (ANDERSON, BOYLE, *et al.*, 1990) é um sistema que oferece uma alternativa ao ensino de Álgebra com a utilização de recursos computacionais. Sua construção foi motivada para testes com metodologia ACT (ANDERSON, CORBETT, *et al.*, 1992) aplicadas em um domínio diferenciado.

O ambiente do sistema é dividido em duas partes, sendo a esquerda uma janela em que o estudante pode trabalhar equações e a direita interage com o tutor, lendo mensagens e solicitando ajuda, além de manipular uma calculadora, conforme representado na Figura 11.

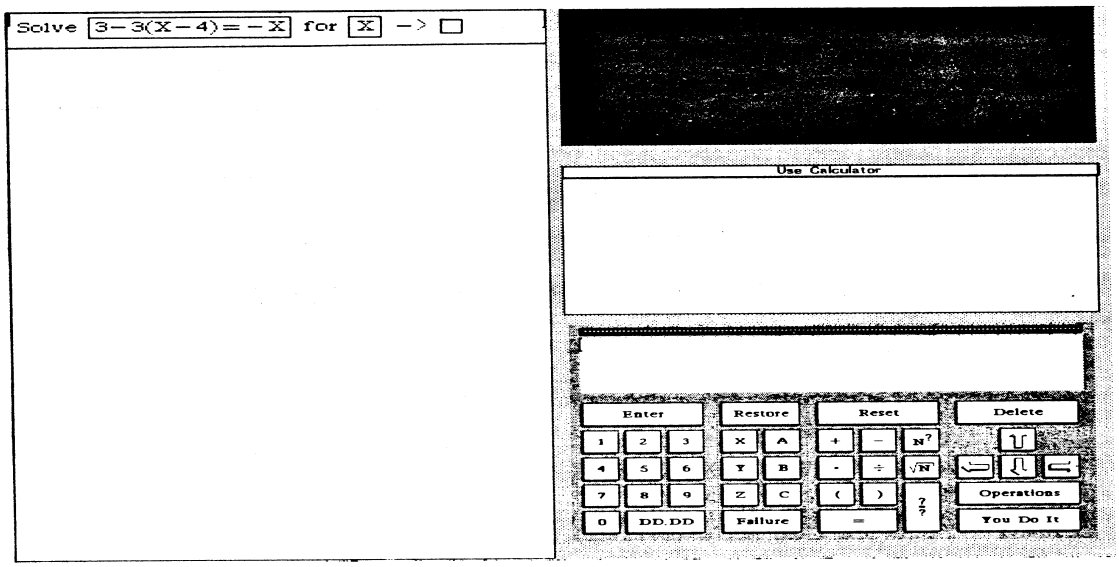

Figura 11: Interface do Algebra Tutor

O conhecimento do sistema é composto por regras ACT, que quando invocadas representam partes da expressão algébrica sendo resolvida pelo aluno. A modelagem do estudante se baseia nestas regras, conforme exercícios são resolvidos corretamente incrementa-se o valor de habilidade do aluno com as regras utilizadas.

As características pedagógicas deste sistema constituem uma revisão de pré-Álgebra e equações lineares e quadráticas. Na parte direita, comentada anteriormente, o aluno pode interagir com o Tutor, que pode oferecer fragmentos de informação para se agregar à solução corrente ou decompor um passo realizado em passos menores para simplificação e melhor compreensão do desenvolvimento do exercício. Para fins didáticos o aluno pode utilizar a opção *"você faz"*, em que o sistema mostra todos os passos para se alcançar um determinado objetivo durante a resolução de um determinado problema.

O estudante possui na tela a equação problema a ser resolvida e pode solucioná-la com passos pré-definidos, utilizando regras ACT individualmente. Isto forma a característica que diferencia este sistema dos demais, em que acontece a decomposição recursiva de objetivos até que se alcancem passos primitivos para a sua solução. Outra opção é fornecer um resultado mais simplificado, omitindo a utilização de tais regras, havendo mais liberdade na construção da solução final do problema proposto pelo sistema.

6.5 COGNITIVE TUTOR AUTHORING TOOLS (CTAT)

O CTAT (VICENT, MCLAREN, *et al.*, 2006) é uma ferramenta que se destina a facilitar a criação de cursos online através do aprendizado por ação, pelas interações do próprio aluno, mantendo registro de como os estudantes interagem com os materiais. Esta ferramenta auxilia na criação de dois tipos de tutores - Tutores Cognitivos e Tutores *Example-Tracing.* Os Tutores Cognitivos se baseiam em um modelo cognitivo baseado em regras, sendo bem sucedido em aprimorar a proficiência de estudantes na Matemática em escolas Americanas (KOEDINGER, ANDERSON, *et al.*, 1997) e permitindo a autonomia em decidibilidade, pois seu conhecimento está baseado em regras de produção. Já os Tutores *Example-Tracing* são uma categoria de tutor que fornecem estas mesmas funcionalidades tutoras sem requisitar programação alguma (KOEDINGER, ALEVEN, *et al.*, 2004), se baseando puramente em exemplos de ações e comportamentos do aluno ao resolver problemas do domínio. Estes exemplos podem ser utilizados como base para migrar um tutor do tipo *Example-Tracing* para um *Cognitive Tutor*.

O CTAT tem como objetivo aumentar a eficiência de autoria de STIs através da abordagem baseada em exemplos, em que o autor do sistema vai demonstrando comportamentos corretos e incorretos de um aluno no sistema, sendo armazenados pelo mesmo. Destes exemplos gravados, o autor pode generalizar e fazer anotações, para que possam servir de base para tutores baseados em exemplos ou ainda para guiar o desenvolvimento e testes de um modelo cognitivo utilizado por um Tutor Cognitivo.

A sua interface compreende um editor de interface (geralmente NetBeans ou Flash), um conjunto de ferramentas para tarefas baseadas em demonstração e exemplos, análise e testes de modelos cognitivos, juntamente a ferramenta externa de edição destes, comumente sendo a plataforma Eclipse.

- **Construtor de GUI** – utilizado para criar a interface para o estudante, um ambiente de solução de problemas em que o aluno interage com o tutor. Suporta esta criação sem programação, com o uso de *drag and drop*.
- **Gravador de Comportamento** – Ferramenta central com três funções: Grava exemplos de comportamento correto/incorreto na resolução de problemas; Implementa a função de *Example-Tracing*; Fornece suporte para planejar e testar modelos cognitivos.
- **Editor Externo:** Editor que possibilita a criação e inspeção de regras do tipo Jess para o modelo cognitivo. O plug-in para a plataforma Eclipse oferece funções de correção de sintaxe e auto-completar.

Até o momento, o CTAT foi utilizado por mais de 220 usuários em um grande número de workshops, cursos de graduação, intensivos de verão e tutoriais. A estimativa é de que 30% a 40% destes usuários não eram alunos ou colaboradores na universidade em que foi desenvolvido, a CMU (*Carnegie Mellon University*), sendo utilizado para tutoria de conteúdos de variadas áreas do conhecimento.

6.6 COMPARATIVO ENTRE OS TRABALHOS RELACIONADOS

Este tópico possui uma análise geral dos sistemas que foram estudados nesta seção do documento, conforme é mostrado na Tabela 1, assim como uma breve visão de como o trabalho proposto poderá preencher certas lacunas apontadas com esta análise. Nas colunas da

tabela estão cada um dos sistemas analisados, e nas linhas os critérios avaliados de forma geral.

Foram selecionados cinco aspectos para comparação, a interação do aluno com o sistema, representação do conhecimento, a abordagem pedagógica, planos de ensino e estratégias pedagógicas. A interação com o usuário compreende aspectos de como o aluno pode inserir dados e interagir com o sistema com exercícios, buscando ajuda, etc. A representação do conhecimento envolve como é mapeado o domínio que o sistema busca ensinar, em que tipo de repositório o mesmo faz consultas para mostrar ao aluno conhecimentos novos e exercícios. A abordagem pedagógica representa como o sistema auxilia o aluno a aprender, de que maneiras uma teoria pedagógica afeta como são mostrados conhecimentos. Os planos de ensino se situam no que influencia a seleção de conteúdo a ser mostrado ao aluno, que variáveis compõem a construção dessa exploração de conteúdo. Por fim, as estratégias pedagógicas compreendem as características utilizadas para adaptar o ensino ao aluno, tais como avaliar quais habilidades o aluno não possui domínio pleno, para seleção de um conteúdo que as desenvolva, como é o caso da *Macro Adaption*.

Tabela 1: Comparativo entre os Trabalhos Relacionados

	ActiveMath	**AnimalWatch**	**Aplusix**	**The Algebra Tutor**	**CTAT**
Interação com Aluno	Seleção simples de respostas pré-definidas	*Word-Problems* que exigem desenvolvimento no papel e informação de resposta única no sistema	Exercícios categorizados com caráter aleatório em variáveis-chave; Utilização livre de regras com sistema *drag and drop*	Janela com duas áreas - problema proposto e espaço para resolução e outra com Tutor e calculadora	Interface em NetBeans ou Flash para construção do material
Representação do Conhecimento	Semantic XML OMDoc (Teorema, prova, definição)	Problemas de diferentes dificuldades relacionados com 30 habilidades matemáticas	Regras para resolução de equações referenciadas a conteúdos de apoio;	Regras ACT representando partes da expressão algébrica sendo manipulada	No caso de um Tutor Example-tracing são exemplos de ações do estudante; No Tutor Cognitivo são regras de produção.
Abordagem Pedagógica	Critérios para mostrar informações adicionais, ordem destas e exercícios, abordagem construtivista com liberdade para exploração de conteúdos	Problemas relacionados com trinta habilidades matemáticas, revisão periódica destas, abordagem construtivista pelo raciocínio exigido em papel e liberdade para respostas	Verificação da solução final, traçando passos anteriores e oferecendo auxílio nas regras com utilização incorreta, ambiente permite livre exploração de regras e operações para exercícios, caráter construtivista	Janela com Tutor oferecendo fragmentos de informação corretos sobre o problema, permitindo ao aluno construir seu conhecimento gradualmente, no seu ritmo, caráter construtivista	A abordagem consiste no aluno interagir com problemas e resolvê-los, sendo auxiliado através do monitoramento de suas ações em um modelo de aluno
Planos de Ensino	Lista pré-definida de conteúdos e exercícios	Série de *Word-Problems* gradualmente mais	Mapas de Exercícios pré-cadastrados pelo professor no	Abre mais espaço para a resolução de problemas	Problemas definidos com uma interface drag and

	baseados no cenário que o estudante selecionar, *exploração inicial* ou *exame*	complexos, que exercitam habilidades mapeadas no modelo de estudante	sistema, sobre conteúdos;	que o usuário deseje resolver, atua mais como ferramenta para resolução de problemas algébricos	drop e baseados em comportamento correto/incorreto informado pelo designer do sistema
Estratégias Pedagógicas	Avalia que informações são necessárias para mostrar ao aluno, avaliando a ordem que devem ser expostas posteriormente e decidindo o número de exemplos/exercícios	Oferece problemas de dificuldade cada vez maior desde que o estudante mostre que domina certas habilidades prévias	Para passar para o próximo conteúdo é necessário resolver com sucesso todos os pertencentes ao conjunto atual	Oferece equações que gradualmente exercitem habilidades contidas no Modelo de Estudante	Como o sistema mantém registro das ações do aluno, pode oferecer auxílio através da análise dos exemplos em sua base ou das regras de produção que estão sendo utilizadas pelo aluno.

Quanto à interação, os sistemas oferecem seleção ou inserção simples de respostas finais, mas com contexto e conteúdo bem trabalhado, ou oferecem um ambiente mais livre para elaboração da resolução sem este background. A exploração de conteúdos no PAT2Math é realizada de maneira diferente, através do seu Agente Tutor e Agente Modelo de Aluno, que recomendam conteúdos segundo a estratégia nomeada *Macro-Adaption* (CORBETT e ANDERSON, 1995; SHUTE, 1993) em conjunto com a avaliação de informações cognitivas do aluno. A livre exploração não procede no sistema, o caráter de interação com os materiais é sempre assistido por esta característica.

Comparando as representações de conhecimento, a estrutura predominante é regras, tendo em vista o caráter procedural que a resolução de problemas algébricos exige, ou relação de conteúdos com estas regras pré-definidas. Dentro do sistema PAT2Math existe um agente de domínio que contém os materiais instrucionais a serem mostrados para o aluno, sendo desenvolvido com a tecnologia de representação OMDoc.

A abordagem pedagógica dos sistemas é predominantemente construtivista, oferecendo liberdade de resolução de problemas e inclusão de respostas finais aos problemas propostos, havendo em alguns casos uma parte tutora a disposição do estudante, que assim faz o seu ritmo de aprendizado. A seleção de cenário de aprendizado, no caso do ActiveMath, concretiza a liberdade do aluno em alterar o ritmo e dificuldade dos materiais abordados no sistema. A verificação de respostas, no entanto, ocorre apenas na resposta final informada ao aluno ou nas regras que estão sendo utilizadas em dado momento, oferecendo feedback em mais de um momento ao aluno. O restante do caráter pedagógico está na amostragem de conteúdos com base nas regras que o aluno utiliza com sucesso ou não. No PAT2Math, a abordagem pedagógica se situa nas teorias construtivistas de Ausubel e sua aprendizagem significativa, utilizando uma ontologia para definição de conteúdos e representação dos conceitos subsunçores que o aluno está desenvolvendo, bem como relações entre eles através da mesma estrutura. A liberdade para realizar exercícios utilizando o conjunto de operações que o aluno julgar pertinente consolida este caráter construtivista da mesma forma, além do auxílio oferecido em caso de erro, com a apresentação de novos conteúdos e novos exercícios propostos para a construção de conhecimento do aluno.

Os planos de ensino dos sistemas avaliados consistem em listas de problemas e conteúdos, que consideram o cenário de aprendizado que o estudante seleciona, no caso do ActiveMath, ou habilidades que estão sendo exercitadas, no caso do AnimalWatch. O Aplusix trabalha com mapas de exercícios manualmente cadastrados em sua maioria, enquanto o Algebra Tutor oferece opções de exercícios e materiais conforme o sistema percebe o domínio

do aluno ao utilizar certas habilidades. Em geral os sistemas se preocupam com a quantidade e dificuldade de exercícios a serem expostos, aumentando gradualmente em relação ao domínio que o aluno demonstra, utilizando o modelo de estudante como variável de decisão para tal, ou a confirmação da realização de um conjunto de exercícios para avançar e ofertar um novo grupo ao aluno, como acontece no Aplusix. O PAT2Math, por sua vez, trabalha com Planos de Ensino através do seu Agente Tutor, Agente Modelo de Aluno e da Ontologia de Conteúdos. Um plano de ensino padrão, descrito formalmente nesta ontologia, contém os conteúdos a serem ensinados segundo os Parâmetros Curriculares Nacionais (PCNs).

As estratégias pedagógicas presentes nos sistemas utilizam primariamente o modelo de estudante como variável na tomada de decisão sobre de que maneira os conteúdos devem ser apresentados, tais como qual a dificuldade do material ou a ordem dos mesmos. Neste cenário o Agente Tutor do PAT2Math, em conjunto com as informações provenientes do Agente de Modelo de Aluno, aplica sua estratégia pedagógica para saber qual conteúdo trabalhará as habilidades ainda não perfeitamente desenvolvidas pelo aluno. No caso é aplicada a teoria de *Macro-Adaption* (sua estratégia pedagógica) e são verificadas quais habilidades o aluno não possui domínio pleno, selecionando um conteúdo que as exercite. Assim ele segue a partir daquele ponto a ordem contida na estrutura da ontologia. Isso consequentemente mostrará ao tutor em que ponto da ontologia de conteúdos deve iniciar o processo de ensino, para aquele aluno em questão. Em caso de erro na resolução de exercícios correlatos ao conteúdo, a ontologia é consultada para adaptar este plano de ensino para que sejam mostrados materiais instrucionais de acordo com a habilidade que o aluno utilizou de maneira indevida.

7. SISTEMA TUTOR INTELIGENTE PAT2MATH

O trabalho proposto neste documento consiste na definição e implementação dos agentes Tutor e Modelo de Aluno do Sistema Tutor Inteligente PAT2Math. Esta seção tem como objetivo abordar as funcionalidades do projeto no qual a contribuição deste Trabalho de Mestrado está inserido, bem como a sua contextualização no cenário de ensino brasileiro.

O Projeto PAT2Math (JAQUES, 2008; JAQUES, LEHMANN e SYLVIE, 2009; DAMASCENO, CRUZ e JAQUES, 2010; MELLO, RUBI, *et al.*, 2009; SEFFRIN, RUBI, *et al.*, 2009; OLIVEIRA, VICARI, *et al.*, 2008; DAMASCENO e JAQUES, 2010) é um Sistema Tutor Inteligente que oferece uma alternativa ao ensino de Álgebra Elementar, entrando no cenário descrito na Introdução deste trabalho, em que os alunos da realidade brasileira de ensino possuem sérias dificuldades no aprendizado de Matemática. Isto constitui um diferencial interessante comparando-se com demais trabalhos desta área, como aqueles citados na seção de “Trabalhos Relacionados”, pois está sendo desenvolvido dentro do ambiente nacional, acompanhando a sua realidade docente.

A construção do sistema se baseia no conceito de um ambiente inteligente de aprendizagem capaz de considerar as emoções do aluno, fazendo uso de um Agente Pedagógico Animado que aplicará táticas pedagógicas para motivação e engajamento em sua interação com o ambiente, através de emoções inferidas e no seu desempenho. Esta característica emocional inferida pelo sistema traz um conjunto de variáveis importantes na tomada de decisão sobre como proporcionar um ensino individualizado para o aluno.

7.1 AGENTES DO PAT2MATH

Cada um dos Módulos do PAT2Math, que seguem a estrutura descrita na seção ‘2.1 Arquitetura Padrão de um Sistema Tutor’ correspondem a um agente computacional. Existe um total de cinco agentes que compõem atualmente a arquitetura multiagente deste projeto, correspondendo aos módulos da arquitetura tradicional de um sistema tutor inteligente, sendo eles: Agente Interface, Agente Domínio, Agente Tutor, Agente Modelo de Aluno e Agente Modelo Cognitivo.

Agente Tutor

Este agente analisa as informações do estudante e seleciona uma estratégia pedagógica adequada, para criação de um plano de ensino apropriado para o mesmo. Está sendo implementado em linguagem de programação Java e integrado com as demais implementações realizadas dos outros agentes. Como sua definição e funcionalidades fazem parte da proposta deste trabalho, maiores detalhes deste se encontram na sessão 'Trabalho Proposto' deste documento.

Agente de Domínio

Este agente varre a sua base, a Base de Domínio, em que todo o conhecimento declarativo, enviando estes dados para o Agente Tutor quando requisitado. Está sendo desenvolvido com linguagem Java e lida com banco de dados em OMDoc, um formato baseado em XML para representar o conteúdo de documentos matemáticos. Diferentemente de formatos como OpenMath e MathML, que representam apenas fórmulas matemáticas, OMDoc é um formato capaz de representar toda gama de conteúdos matemáticos escritos, tais como explanações, exemplos, teorias e exercícios. Além disso, pode-se inserir a semântica do conteúdo matemático e meta-informações sobre ele.

O conhecimento matemático declarativo contido no PAT2Math é formado por tópicos, uma pequena unidade de conhecimento a ser apresentado ao aluno. Cada um destes é formado por diversos elementos OMDoc, nos quais podem ser manipulados e combinados pelo Agente Tutor constituindo a criação de diferentes planos de ensino.

Para a estruturação sobre os tópicos matemáticos, será utilizada uma base de dados entidade-relacionamento, conforme a 'Figura 12: Diagrama Entidade Relacionamento dos Componentes do Domínio'. A organização de cada tópico utilizará as seguintes informações:

- Tipo do Conteúdo (Exercício, Explanação, Exemplo)
- Assunto do Conteúdo
- Tópico do Assunto
- Grau de equação do conteúdo (0 grau, 1º grau, 2º grau)
- Nível de dificuldade do conteúdo
- Habilidade
- Teoria do Design Instrucional (Problema, Ativação, Demonstração, Aplicação, Integração)

Como exemplo pode-se ter o seguinte componente na Base de Domínio, segundo os tópicos descritos anteriormente:

- Explanação
- Equações
- Equações Primeiro Grau
- Primeiro Grau
- Dificuldade 1
- Soma, Operação Inversa
- Demonstração

No caso do exemplo acima, trata-se de um conteúdo do tipo explanação, sobre o assunto de equações que aborda equações do primeiro grau se enquadrado como dificuldade um e trabalha com as habilidades de soma e operação inversa. Por fim, deve ser apresentado, segundo a TDI, como demonstração do conteúdo.

Um exemplo de material desta categoria é o seguinte: *"Equações Algébricas são igualdades que contém números desconhecidos, geralmente representados por letras. Para compreendermos as manipulações necessárias para a resolução de uma equação devemos pensar em uma balança de dois pratos. Assim, para termos o seu equilíbrio, ou seja, para que os dois pratos fiquem na mesma altura (iguais) precisamos ter o mesmo peso dos dois lados. Então, na nossa equação algébrica temos o "x" que representa este valor que irá equilibrar. Neste sentido, a letra "x" é a Variável e/ou Incógnita da equação que pode ser representada por qualquer outra letra ou símbolo que represente o número desconhecido da equação. Normalmente são utilizadas as letras x, y, z, a, b, entre outras. Mas todas representam o número a ser descoberto"*.

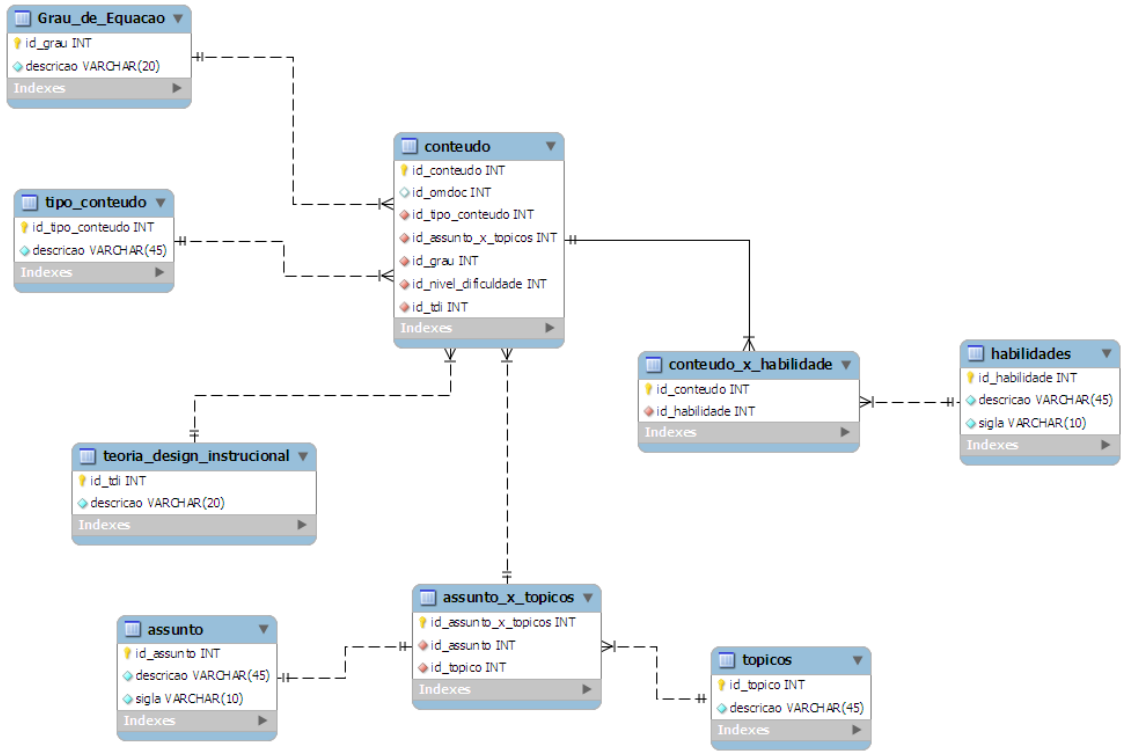

Figura 12: Diagrama Entidade Relacionamento dos Componentes do Domínio

Neste diagrama, criado através da ferramenta MySQL Workbench podemos destacar as seguintes relações:

CONTEÚDO X TIPO_CONTEÚDO

Cada conteúdo terá no mínimo 1 e no máximo 1 tipo de conteúdo (Exercício, Explanação, Exemplo) e um tipo de conteúdo pode estar em vários conteúdos.

ASSUNTO X TÓPICO

Cada assunto terá no mínimo 1 e no máximo n, um ou mais tópicos estarão presentes em no mínimo 1 e no máximo 1 assunto

CONTEÚDO X ASSUNTO X TÓPICO

Cada conteúdo terá no mínimo 1 e no máximo n tópicos, um ou mais tópicos estarão presentes em no mínimo 1 e no máximo 1 assunto.

CONTEÚDO X GRAU_EQUAÇÃO

Cada conteúdo terá no mínimo 1 e no máximo 1 grau (0, 1º grau, 2º grau)

CONTEÚDO X NÍVEL_DIFICULDADE
Cada conteúdo terá no mínimo 1 e no máximo 1 nível de dificuldade.

CONTEÚDO X HABILIDADES
Cada conteúdo terá no mínimo 1 e no máximo n habilidades.

CONTEÚDO X TDI (Teoria de Design Instrucional)
Cada conteúdo terá no mínimo 1 e no máximo 1 TD1, e um TDI pode estar presente em vários conteúdos

A partir da necessidade da Base de Domínio armazenar um grande número de fragmentos OMDoc, por consequência XML, foi adotado o uso de um banco de dados XML nativo. A escolha de um banco de dados XML nativo foi motivada pelo desempenho alcançado ao armazenar os documentos em seus formatos nativos e pela flexibilidade oferecida para organização, consulta e manipulação de fragmentos dos documentos através da linguagem de consulta do formato XML XQuery, permitindo a composição de novos documentos dinamicamente (MELLO, RUBI, *et al.*, 2009).

Agente Modelo de Aluno

Monitora as informações sobre o aluno conforme este interage com o sistema, servindo como referência para o Agente Tutor no momento da criação de um plano de ensino específico para um aluno. Seu desenvolvimento está sob linguagem Java, fortemente integrado com as implementações do Agente Tutor e demais agentes do sistema. Como sua definição e funcionalidades fazem parte da proposta deste trabalho, maiores detalhes deste se encontram na sessão de ‘Trabalho Proposto’ deste documento.

Agente do Modelo Cognitivo

Neste módulo, é fornecida uma equação a ser resolvida pelo aluno, escolhida pelo agente tutor segundo informações do Modelo de Aluno, para o aluno resolver passo a passo. Cada passo, ou seja, cada solução intermediária fornecida pelo aluno, é corrigida pelo Agente Modelo Cognitivo. As regras possíveis para resolução de equações algébricas estão presentes neste agente, através delas sendo possível verificar se um determinado aluno conseguiu

realizar uma etapa de resolução de um problema corretamente. Estas regras referenciam as operações algébricas que podem ser aplicadas na resolução de problemas desta categoria. Isto constitui o *Model Tracing*, abordado anteriormente na seção
2.2 Modelagem do Componente Tutor de um Sistema Tutor Inteligente (STI), fazendo-o atuar como um sistema especialista, estando habilitado a resolver e verificar a integridade de equações algébricas. O reconhecimento e resolução da equação ocorre com a aplicação de regras sucessivamente, até que não se possa mais aplicar regra alguma, alcançando um resultado final. Não havendo mais soluções possíveis e alcançado um resultado não permitido pelas regras, a equação informada pelo aluno é considerada incorreta.

Este agente atua com habilidades referentes à resolução de equações, representadas como um conjunto de regras que são utilizadas nos passos de resolução de exercícios desta categoria, para mostrar o método ideal de se resolver um dado problema proposto. Tais habilidades estão presentes no Modelo de Aluno de cada estudante, e este agente pode informar ao Agente Tutor se o aluno está aplicando-as corretamente. Em caso negativo, o AMC auxilia o AT com a informação de qual habilidade seria melhor aplicada naquele momento, para que ocorra a demonstração de material de ensino condizente e a posterior criação do plano de ensino seja adequada. Estas habilidades são:

- Adição;
- Subtração;
- Divisão;
- Multiplicação;
- Mínimo Múltiplo Comum (MMC);
- Distributiva em relação à multiplicação;
- Fator Comum (Colocar Termo em Evidência);
- Produto Notável – Quadrado da Soma;
- Produto Notável – Quadrado da Diferença;
- Produto Notável – Produto de uma soma indicada por uma diferença;
- Fórmula de Bháskara;
- Operação Inversa / Oposta (adição, subtração, divisão e multiplicação);
- Raiz Quadrada;
- Simplificação.

O papel deste agente consiste na verificação da consistência dos elementos de uma equação, trabalhada pelo aluno durante a resolução de exercícios. Cada passo realizado deve se enquadrar em uma destas regras, caso contrário este agente informa ao Agente Tutor para que entre em cena para oferecer o auxílio adequado, com a customização do plano de ensino padrão, para apresentação do conteúdo referente ao assunto vigente. O Agente de Modelo Cognitivo foi desenvolvido em linguagem Java, utilizando uma extensa base de definições de regras sobre Equações Algébricas, que foram mapeadas com a ferramenta de criação de regras de produção *Drools* (SEFFRIN, RUBI, *et al.*, 2009).

Agente de Interface

É responsável por unir os dados sobre ações do usuário dentro do sistema e, posteriormente, enviá-los para tomada de decisão do Agente Tutor. Este agente também é encarregado de mostrar tópicos e informações na tela do sistema. A interface está sendo desenvolvida como uma aplicação *Web*, um tipo de aplicativo desenvolvido na linguagem Java que executa dentro de navegadores para Internet. Para a resolução de equações, foi desenvolvido o módulo PATEquation, que é composto por uma versão estendida do *DragMath* que é um *Applet* de edição de equações.

A interface do módulo de resolução de equações disponibiliza ao usuário ferramentas para auxiliá-lo na resolução de equações matemáticas passo a passo. Cada passo consiste de uma solução intermediária para uma operação algébrica escolhida pelo usuário para ser realizada sobre um ou mais operandos. É exibido ao usuário informações sobre a correção dos passos, sendo este retorno de dois tipos: indicando se a solução intermediária é correta ou não para a operação escolhida ou indicando que a operação escolhida não corresponde a solução fornecida (SEFFRIN, RUBI, *et al.*, 2009), podendo ser visualizada na ‘Figura 13: Interface do PatSolver’.

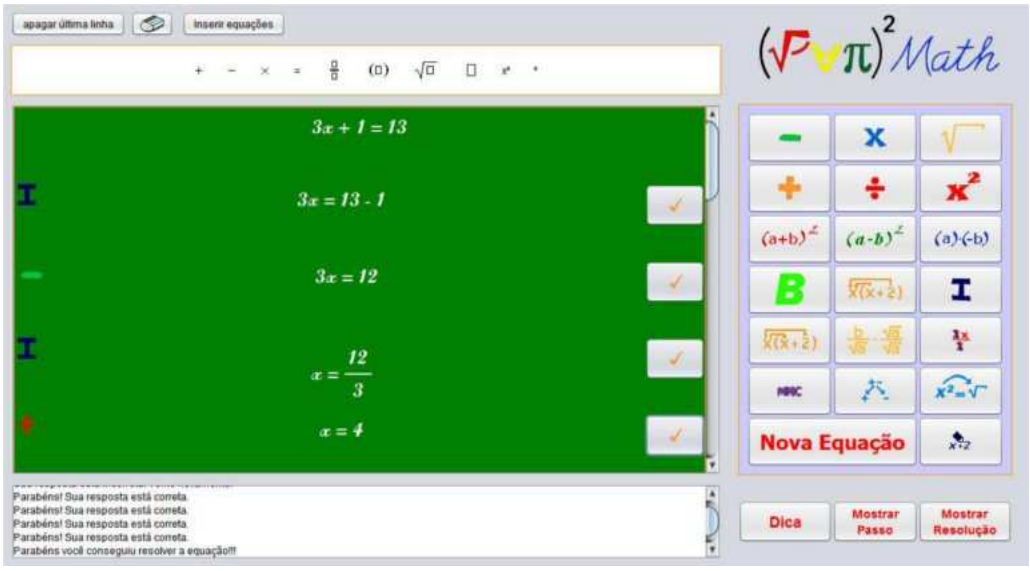

Figura 13: Interface do PatSolver

7.2 SISTEMA MULTIAGENTE DO SISTEMA TUTOR INTELIGENTE PAT2MATH

O diagrama, que pode ser visualizado na ‘Figura 14: Diagrama Geral dos Agentes’, explica a relação entre os agentes da arquitetura proposta. Trata-se da remodelagem da arquitetura inicial (MELLO, RUBI, *et al.*, 2009) já existente do projeto, mas realizada segundo uma metodologia de sistemas multiagentes, no caso a Prometheus, trabalho descrito em (DAMASCENO, CRUZ e JAQUES, 2010). Nela estão presentes o Agente Tutor (AT), Agente de Domínio (AD), Agente de Modelo Cognitivo (AMC), Agente de Modelo de Aluno (AMA) e Agente de Interface (AI). Percebe-se com esta figura, a importância do Agente Tutor, pois este atua como centro dos demais presentes na arquitetura, tomando decisões, repassando e processando dados.

A metodologia *Prometheus* foi selecionada para este projeto por fornecer uma modelagem concreta de fatores relevantes a um sistema multiagente, além de possuir um forte valor educacional agregado, sendo utilizada no ensino de estudantes que estão explorando possibilidades de construção de um sistema com múltiplos agentes. Isso é de extrema importância dadas as circunstâncias do projeto, em que existe um número considerável de pesquisadores conduzindo trabalhos relacionados ao sistema e implementando funcionalidades. Adicionalmente, *Prometheus* fornece uma interface para a fase de Design da

metodologia, tornando claro para uma equipe de desenvolvimento os avanços realizados em cada versão do sistema, com o uso de uma ferramenta como a *Prometheus Design Tool* (PDT), por exemplo. Outro aspecto interessante para ser adotada esta metodologia é avaliar o diagrama do sistema idealizado, analisando possíveis melhorias e embasando melhor a continuidade das atividades de implementação em cima deste.

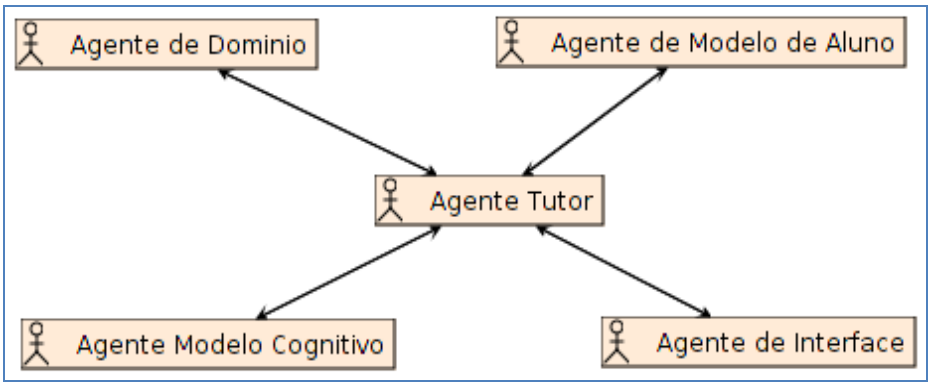

Figura 14: Diagrama Geral dos Agentes

O AD consulta na sua base de dados Objetos de Aprendizagem conforme o AT solicita. O Agente de Modelo de Aluno pesquisa, na sua respectiva base de informações, dados sobre o aluno que atualmente utiliza o sistema, conforme o AT requisita informações, para criar um plano de ensino por exemplo. Já o AI recebe informações do Tutor para mostrar na tela, bem como informa a ele dados de uso do sistema, para que sejam encaminhados ao AMA, que atualiza a base de informações dos estudantes. Por fim o AMC recebe os passos de resolução de equações algébricas do aluno, e os analisa a fim de determinar se estão corretos ou não.

7.3 PAPÉIS DOS AGENTES

Esta seção apresenta a relação de papéis no sistema que cada um dos agentes presentes na arquitetura desempenha. De um total de cinco agentes, existem nove papéis que são trabalhados, sendo o AMA o que mais possui tais funções. O diagrama que pode ser visualizado na ‘Figura 15: Papéis dos Agentes do Sistema’, mostra a relação de papéis no sistema que cada um dos agentes presentes na arquitetura desempenha. De um total de cinco

agentes, existem nove papéis que são trabalhados, sendo o AMA o que mais possui tais funções.

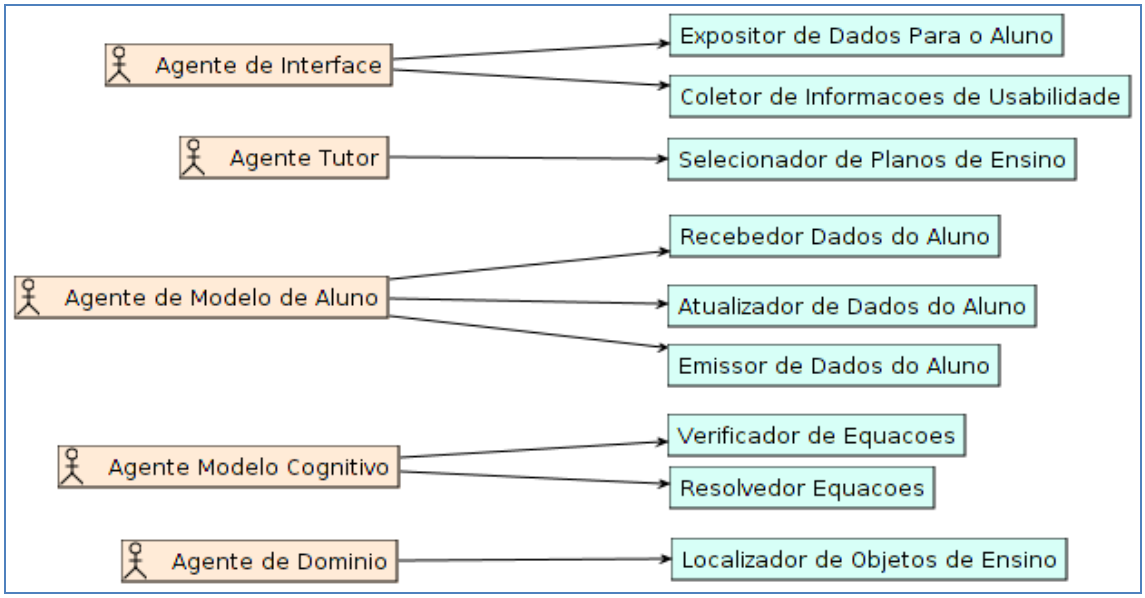

Figura 15: Papéis dos Agentes do Sistema

O Agente de Interface (AI) atua como:

- Expositor de Dados para o Aluno: Toda e qualquer informação que o AT julgue necessário mostrar ao aluno é repassado a este agente, que mostra tais informações no formato especificado, seja na tela do computador ou em um navegador de celular;
- Coletor de Informações de Usabilidade: Toda interação do usuário, que o Agente considere relevante, é mapeada e repassada ao Tutor, para que seja encaminhada à base de Modelo de Aluno. Outra característica é a conclusão de lições/exercícios, que devem ser atualizados como novos conhecimentos que o estudante em questão possui.

O Agente de Modelo de Aluno (AMA) atua como:

- Recebedor de Dados do Aluno: Quando o AT recebe uma informação relevante sobre o aluno através do AI, este encaminha ao AMA.
- Atualizador de Dados do Aluno: Uma alteração sobre o estado do aluno, como ter lido com sucesso um determinado conteúdo, é alterada na Base de Modelo do Aluno através deste papel.
- Emissor de Dados do Aluno: Quando o AT requisita uma informação sobre o aluno que atualmente está utilizando o sistema, o Agente de AMA é responsável por encaminhar tais dados.

O Agente de Modelo Cognitivo (AMC) atua como:

- Verificador de Equações: Quando um determinado aluno executa um passo durante a resolução de equação, o AT se encarrega de processar essa informação e enviar uma mensagem para o AMC para que este valide-a.
- Resolvedor de Equações: O AT pode julgar necessário mostrar a solução completa de uma equação-problema que o estudante esteja trabalhando, enviando uma mensagem para o AMC para que este envie o conjunto de passos a serem executados para tal.

O Agente de Domínio (AD) atua como:

- Localizador de Objetos de Ensino: O AT do sistema pode necessitar de um objeto de aprendizado com características distintas com características distintas, seja com um texto mais aprofundado, ou imagens mais explicativas. Neste momento, ele envia uma mensagem ao AD para que faça a devida varredura na Base de Domínio e retorne-o para a devida construção do plano de ensino por parte do AT.

O Agente Tutor (AT) do Sistema atua como:

- Selecionador de Planos de Ensino: Informações do aluno, características dos Objetos de Aprendizado, disponibilidade de recursos, entre outros fatores são levados em consideração quando o AT vai criar um plano de ensino adequado ao aluno.

7.4 RELAÇÕES DAS BASES DE DADOS COM OS PAPÉIS DOS AGENTES DO SISTEMA

Esta seção mostra a relação das bases de conhecimento com os papéis dos agentes presentes na arquitetura do projeto proposto.

- *Base Cognitiva de Resolução de Equações*: Contém as regras para resolução de uma equação algébrica, organizadas em passos. É utilizada pelo AMC para resolução e validação de passos individuais no desenvolvimento de equações. É acionada sempre que este tenta resolver e validar passos realizados por alunos na resolução de uma equação.

- *Base de Modelo Cognitivo do Aluno*: Contém todos os dados relevantes sobre o aluno, tais como quais exercícios foram resolvidos e quais lições foram lidas dentro do sistema. Trata-se de uma base extremamente importante para o AT, pois esse a consulta para a criação de um plano de ensino mais eficiente para o aluno em questão. Tal base lida como AMC, quando este executa os papéis de atualizar Dados do Aluno, Emissor de Dados do Aluno e Recebedor de Dados do Aluno. Lida também com o Agente de Tutor, quando este recebe dados de usabilidade do sistema por parte do aluno, na execução do papel Coletor de Informações de Usabilidade.
- *Planos de Ensino e Estratégias Pedagógicas*: São bases com elementos comuns no projeto de um plano de ensino, possui estereótipos que o AT deve considerar ao derivar um plano personalizado para um determinado aluno. Interagem diretamente com o AT quando este executa seu papel de Selecionador de Planos de Ensino.
- *Base de Domínio:* Contém todas informações e explicações sobre os variados conteúdos contidos na arquitetura tutora do sistema proposta, sendo característica de tais objetos de aprendizado diversos níveis, para alunos distintos, com fatores cognitivos diferenciados em relação a tais conteúdos.

No diagrama que pode ser visualizado na Figura 16 são retratadas as interações entre os agentes e suas respectivas bases de dados, através de mensagens.

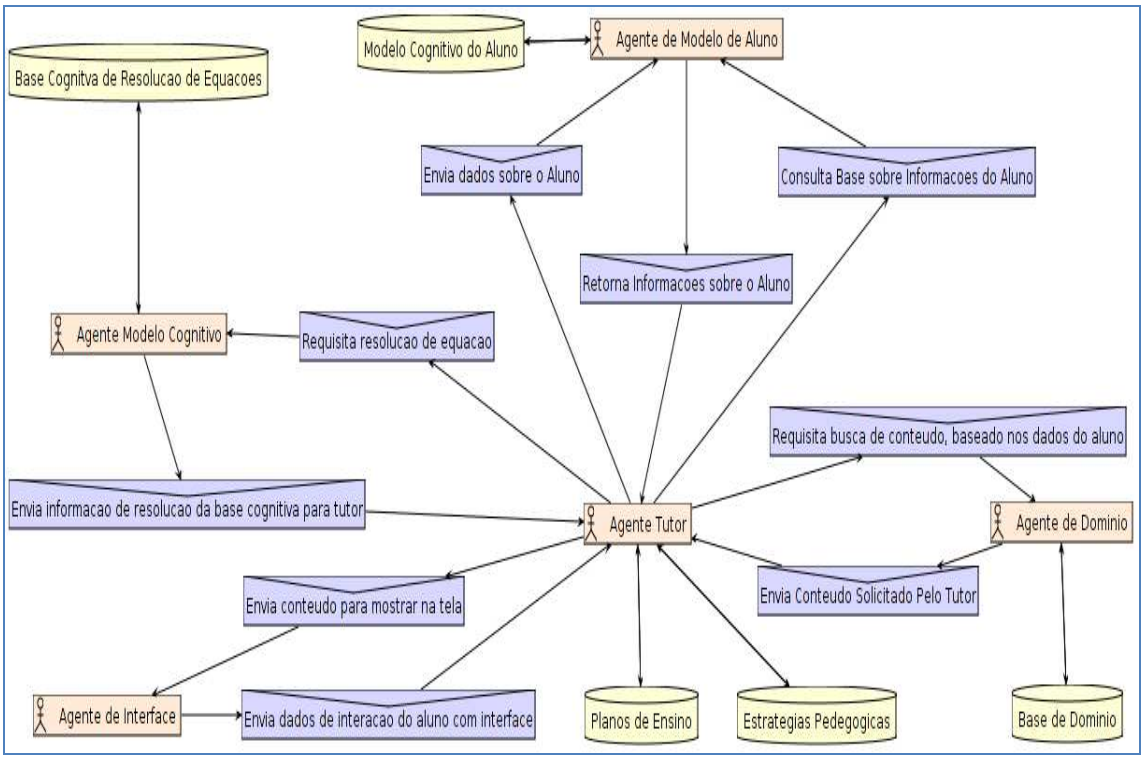

Figura 16: Diagrama Geral do Sistema

O Agente Tutor (AT) pode enviar as seguintes mensagens:

- *Envia dados sobre Aluno:* O AI encaminha informações sobre mudanças no comportamento e características do aluno, como um exercício concluído com sucesso que deve ser atualizado na sua base de conhecimentos.
- *Consulta Base sobre Informações do Aluno*: O AT pode requisitar ao AMA algum dado pertinente sobre o estudante que estiver usando o sistema em determinado momento. Isso seria usado na criação de um plano de ensino adequado ao aluno, por exemplo.
- *Requisita resolução de Equação*: O AT pode requisitar, através de uma mensagem, a resolução, por completo ou em parte, de uma equação-problema que o aluno esteja tentando solucionar.
- *Requisita Busca de Conteúdo, baseado nos Dados do Aluno*: O AT pode requisitar um determinado conteúdo para criação de um plano de ensino, baseando-se em características do aluno que está usando o sistema.
- *Envia Conteúdo para Mostrar na Tela*: O AT, ao receber um determinado Objeto de Aprendizado que havia requisitado ao AD, pode enviar tais informações para a tela através de uma mensagem para o AI.

O Agente de Interface (AI) pode enviar as seguintes mensagens:

- *Envia dados de interação do aluno com interface:* Conforme o aluno interage dentro do sistema proposto, informações relevantes são repassadas ao Agente Tutor, para que este encaminhe ao Agente de Modelo do Aluno para a devida atualização, que reflete em maior precisão de dados no momento de criação de um plano de ensino, por exemplo.

O Agente de Domínio (AD) pode enviar as seguintes mensagens:

- *Envia conteúdo solicitado pelo Agente Tutor*: Quando o Agente Tutor requisita um determinado conteúdo, é retornado um Objeto de Aprendizagem condizente com a requisição através desta mensagem. Tal informação é processada a agregada em um plano de ensino que o Agente Tutor esteja criando, por exemplo.

O Agente de Modelo Cognitivo (AMC) pode enviar as seguintes mensagens:

- *Envia informação de resolução da base cognitiva*: Quando o AT requisita a resolução ou validação de uma equação, o AMC envia os passos da solução/validação através desta mensagem.

O Agente de Domínio (AD) pode enviar as seguintes mensagens:

- *Envia conteúdo solicitado pelo AT*: Quando o AT requisita um determinado conteúdo, é retornado um objeto de Aprendizagem condizente com a requisição através desta mensagem.

O Agente de Modelo de Aluno (AMA) pode enviar as seguintes mensagens:

- *Retorna informações sobre o aluno*: Quando o AT requisita uma determinada informação sobre o aluno que está utilizando o sistema, o AMA retorna esta informação através desta mensagem. É utilizada na construção de planos de ensino por parte do AT.

7.5 ESTRUTURA DAS MENSAGENS

O Agente Tutor (AT), durante o processo de ensino ao estudante, pode requisitar ao Agente de Modelo Cognitivo (AMC) a resolução de uma determinada equação para verificar se um passo realizado pelo aluno está correto.

Isso é realizado quando uma mensagem de sistema, contendo a string representando a equação, é encaminhada do AT para o AMC. É o próprio AMC que resolve as equações. Por fim, este último encaminha uma mensagem com a devida resolução. As mensagens serão construídas sob o seguinte padrão, conforme exemplo na ‘Figura 17: *Template* da Mensagem’:

IA:identificação_do_aluno#IC:ação_realizada#EI:estado_atual_equação
#OP:operador_aplicado#RT:resultado_após_operação

Figura 17: Template da Mensagem

A partir de mensagens, como no exemplo acima, são identificadas siglas e a informação é dividida, para que os agentes do sistema possam fazer o seu trabalho, seja atualizar um dado referente ao exercício que o aluno está fazendo ou que operações estão sendo utilizadas para resolver os mesmos. As siglas possíveis para uma mensagem são as seguintes:

IA - Identificação do aluno
IC - Identificação da ação
EI - Equação inteira
OP - Operador
RT - Resultado

Neste exemplo, mostrado na Figura 17, temos:

- Sigla *IA* representando a identificação de qual aluno foi responsável pelo evento a qual a mensagem se refere;

- Sigla *IC* representando qual ação foi realizada pelo aluno identificado com o *IA,* como ler um conteúdo ou realizar um determinado exercício;
- Sigla *EI* mostra como está atualmente a equação sendo manipulada pelo aluno definido em *IA;*
- Sigla *OP* indica qual a operação que o aluno identificado em *IA* está aplicando na equação atual, mostrada em *EI;*
- Sigla *RT* é o resultado esperado como correto através da aplicação da operação *OP* na equação *EI.*

As operações que o aluno pode fazer na resolução de equações, constituindo a parte da mensagem *OP,* são as seguintes:

AD - Adição
SB - Subtração
DV - Divisão
MT - Multiplicação
MM - Mínimo Múltiplo Comum (MMC)
DM - Distributiva em relação à multiplicação
FC - Fator Comum (Colocar Termo em Evidência)
QS - Produto Notável , Quadrado da Soma
QD - Produto Notável ,Quadrado da Diferença
PS - Produto Notável , Produto de uma soma indicada por uma diferença
BK - Fórmula de Bháskara
OI - Operação Inversa / Oposta
RZ - Raiz Quadrada (extrair)
SP – Simplificação

Estas operações são referentes às habilidades utilizadas pelo Agente de Modelo Cognitivo para resolução de problemas e que estão presentes no Modelo de Aluno de cada estudante.

8. TRABALHO PROPOSTO

O projeto das funcionalidades e a devida implementação compreendem dois dos agentes da arquitetura definida na seção anterior, o Agente Tutor e Agente de Modelo de Aluno, este último contendo informações relevantes que serão utilizadas nas funcionalidades do Agente Tutor. No caso do Agente Tutor foi definida e implementada a sua estratégia pedagógica e demais definições do seu Plano de Ensino segundo os Parâmetros Curriculares Nacionais (PCNs), além da sua estrutura e maneira de adaptação deste plano para remediar um aprendizado incorreto de um aluno. Futuramente este Agente conterá mais funcionalidades, a serem desenvolvidas pelos demais participantes do Projeto PAT2Math. Já no caso do Agente Modelo de Aluno, foram definidas as informações sobre o aluno que são mantidas registro, compondo as sessões que este aluno tem com o sistema. Nestas são registradas as interações com os tópicos, previstos no Plano de Ensino que o Agente Tutor segue, bem como os materiais instrucionais referentes aos mesmos, além dos exercícios realizados e devidas habilidades de resolução de equações exercitadas.

8.1 AGENTE DO MODELO DE ALUNO

Como foi visto anteriormente na seção ‘2.1 Arquitetura Padrão de um Sistema Tutor’, a Modelagem do Estudante em um Sistema Tutor Inteligente é um elemento importante para o cumprimento do seu objetivo, proporcionar ensino individualizado a um aluno em um ambiente inteligente de aprendizagem. Se um agente necessita de uma informação a respeito de um dado aluno, o agente Modelo de Aluno é acionado, bem como quando um agente tem uma nova informação sobre este aluno e precisa que ela seja alterada no seu respectivo modelo. O que está sendo considerado no desenvolvimento do Agente de Modelo de Aluno do PAT2Math envolve informações referentes aos materiais acompanhados, assim como dados sobre os tópicos que levaram o aluno a solicitar ajuda ou cometer algum erro e eventuais *misconceptions* na resolução de exercícios. Estas informações são provenientes do *Model Tracing* das atividades do aluno durante a resolução de equações no módulo PATequation, como será explicado em mais detalhes posteriormente.

O Agente Modelo Cognitivo (AMC), conforme explicado na seção ‘7.1 Agentes do PAT2Math’, possui regras de produção para saber resolver equações, bem como corrigir a

resolução de equações algébricas dos alunos. Cada regra está relacionada a uma habilidade (geralmente habilidades relacionadas ao domínio de operações algébricas) que o aluno utiliza na resolução de problemas algébricos. Por exemplo: saber realizar a adição de termos com incógnitas, saber multiplicar dois números inteiros em uma equação, entre outros. Assim, ao corrigir uma resposta intermediária fornecida pelo aluno na resolução de uma equação, o AMC é capaz de verificar qual habilidade está relacionada à regra necessária para resolver aquele passo da equação e se o aluno a soube utilizar corretamente ou não pela resposta dada. Este tipo de modelagem é chamada de *Model Tracing*, sendo realizada em tempo real com o objetivo de fornecer um *feedback* imediato ao aluno, se encaixando na definição sobre o *Inner Loop* abordada por VanLehn (VANLEHN, 2006). O *Model Tracing* só é possível, pois cada operação para resolução da equação (também chamada de habilidade uma vez que representa uma habilidade que o aluno deve aprender) está associada a uma ou mais regras de produção no Modelo Cognitivo. Com isso, o Modelo Cognitivo pode identificar qual habilidade o aluno utilizou para resolver cada passo da resolução, verificando qual regra foi acionada para chegar a mesma solução do aluno. O Modelo Cognitivo repassa essas informações ao Agente Modelo de Aluno para registrá-la. São exemplos de habilidades: a operação inversa de adição, soma de incógnitas, etc. Na Seção ‘8.1.1 Informações Modeladas’, são citadas as habilidades trabalhadas na versão atual do sistema.

A ‘Figura 18: Exemplo de Resolução de Equação com o PATEquation’ a seguir aborda a resolução de uma equação utilizando o PATEquation:

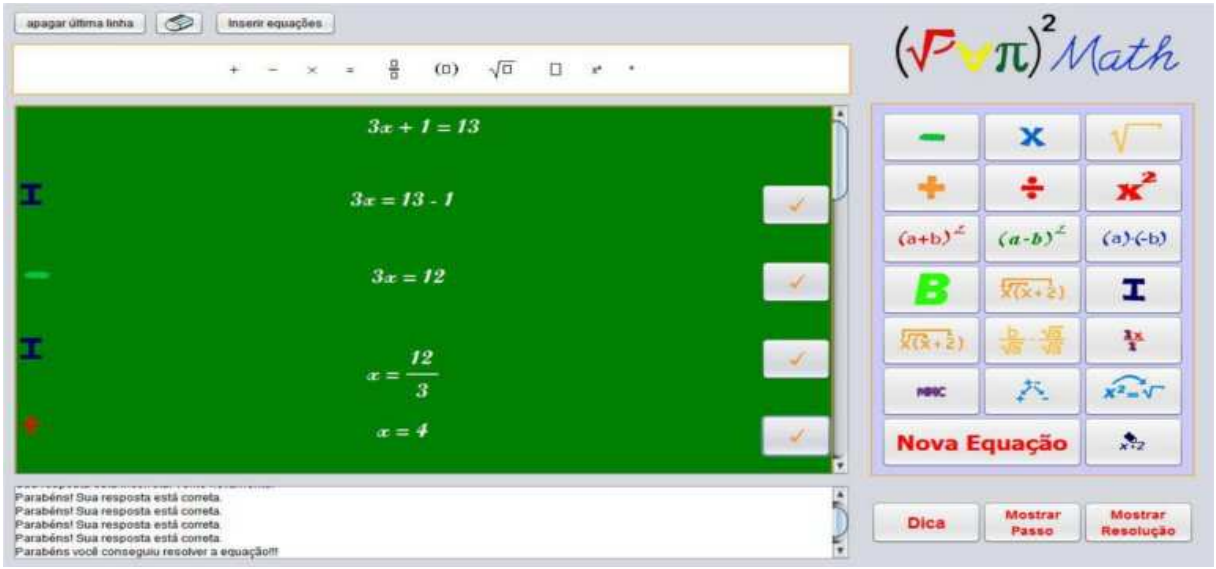

Figura 18: Exemplo de Resolução de Equação com o PATEquation

No exemplo acima, a equação a ser resolvida é “3x + 1 = 13”. O aluno aplica a operação inversa no termo “1”, obtendo a equação “3x = 13 – 1”. Após isso, aplica uma

subtração nos termos “13” e “1”, transformando a equação em “3x = 12”. Com esta equação, o aluno aplicou novamente uma operação inversa entre os dois termos obtendo “x = 12/3”, cujo resultado após a devida divisão efetuada é “4”.

Tais informações levantadas no *Model Tracing*, em conjunto com as informações sobre explicações, exemplos, exercícios e conteúdos que o aluno visualizou e exercitou, farão parte do *Knowledge Tracing* do mesmo aluno. Este sintetiza informações do *Model Tracing*, permitindo a inferência de conteúdos que o aluno domina, fornecendo uma visão global dos conteúdos que o aluno é proficiente ou ainda precisa desenvolver. Por exemplo, se o aluno possui domínio das habilidades soma, subtração, multiplicação e divisão pode-se inferir que o seu domínio no assunto ‘Operações Primárias’ é satisfatório.

Este conjunto de informações é vital para a teoria de *Macro-Adaption*, que é utilizada no *Outer Loop* do sistema para definir que conteúdo deve ser mostrado ao aluno. As informações do *Knowledge Tracing* também serão utilizadas futuramente pelo tutor para definir a customização do plano de ensino do aluno, conforme será explicado na ‘8.2 Agente Tutor’.

No Modelo de Aluno existem estruturas de dados, na forma de tabelas de banco de dados em MySQL, que guardam informações sobre os exercícios realizados pelos estudantes e os seus respectivos passos de resolução, exemplos visualizados e explanações acompanhadas. Este material constituirá as *sessões do aluno* no sistema. Estão detalhadas a seguir:

8.1.1 Informações Modeladas

Esta subseção contém as informações a respeito do aluno que o Modelo de Aluno mantém registro e pode oferecer ao Agente Tutor em suas funcionalidades.

A estrutura da ‘Tabela 2: Tabela de Login Aluno’ contém os nomes de usuário de cada aluno que utiliza o sistema, bem como os seus reais nomes e série atual. É a tabela consultada na validação para entrar no sistema.

Tabela 2: Tabela de Login Aluno

Aluno	**Nome Usuário**	**Senha**	**Nome Aluno**	**Série**
[id aluno]	[nome usuário]	[senha]	[nome do aluno]	[série]

A ‘Tabela 3: Habilidades Aluno’ contém uma estimativa de quanto o aluno domina de cada uma das habilidades usadas na resolução de equações. Se o aluno realizar corretamente o total de exercícios do sistema que utilizem esta habilidade, a crença é que o seu domínio seja pleno. Trata-se de um valor percentual, uma relação do número de exercícios realizados e exercícios que ainda serão propostos que utilize determinada habilidade.

Considerando um tópico do plano de ensino que contenha dez exercícios, em que um aluno hipotético resolveu oito destes exercícios com sucesso, o sistema faz a média deste valor de exercícios totais versus exercícios resolvidos com sucesso atribuindo 80% de proficiência do aluno no tópico em questão. Em relação às habilidades trabalhadas, sabe-se o número de momentos em que uma habilidade deveria ser aplicada ao longo destes exercícios, assim como o número de vezes que este aluno conseguiu aplicá-la com sucesso, podendo-se fazer a média deste valor também, que é atualizado no seu respectivo Modelo de Aluno.

Como parte dos trabalhos futuros envolvendo as características do que foi proposto, será definida uma função para progressão do valor de *mastery* que o aluno adquire realizando exercícios de um conteúdo. Tal função será construída em conjunto com especialistas do domínio e implementada no sistema pelos demais integrantes do projeto PAT2Math.

Tabela 3: Habilidades Aluno

Habilidade	**Aluno**	**Propriedade**	**Valor**
[id habilidade]	[id aluno]	[habilidade]	[valor de *mastery]*

A ‘Tabela 4: Conteúdos Aluno’ contém as informações a respeito dos conteúdos que o aluno pode explorar no sistema *versus* o quanto ele os domina, em percentual. Estes conteúdos são os tópicos previstos no Plano de Ensino, sendo compostos por instâncias dos materiais instrucionais, segundo a Teoria de Design Instrucional de Merril. Este valor é calculado com a média dos exercícios realizados com sucesso em relação ao número total de exercícios disponíveis para o conteúdo em questão.

Tabela 4: Conteúdos Aluno

Id Conteúdo	**Aluno**	**Nome do Conteúdo**	**Proficiência**

[id conteúdo]	[id aluno]	[nome]	[valor %]

Tabela 5: Exercícios Realizados pelo Aluno

ID Aluno	**ID Exercício**	**Passo Realizado (Estado Atual do Exercício)**	**Tempo (Data/Hora)**	**Operação Realizada (Habilidade Utilizada)**	**Operação que Tutor faria**
[id aluno]	[id exercício]	[estado atual]	[dd/mm/AA hh:mm]	[operação]	[operação]

Tabela 6: Operações Realizadas pelo Aluno nos Exercícios

ID Aluno	**ID Exercício**	**Habilidade Utilizada**
[id aluno]	[id exercício]	[operação]

A Tabela 5 contém a identificação dos exercícios realizados, bem como o tempo em que foram resolvidos. Contém também a lista de operações que o aluno realizou e que deveriam ter sido realizadas, que estão mapeadas na ‘Tabela 6: Operações Realizadas pelo Aluno nos Exercícios’. Estas habilidades são comparadas com um Modelo Ideal do Módulo Especialista, avaliando qual habilidade um especialista aplicaria em dado momento e o estado atual de resolução do exercício. Monitorar o quanto um aluno demora em avançar em cada passo de um exercício é importante, pois se pode inferir em quais habilidades (determinado pelas habilidades ativadas) ele possui dificuldade em avançar seu raciocínio. Este monitoramento em cima das habilidades que o aluno utiliza na resolução de exercícios forma o conceito de *Model Tracing* abordado na seção ‘2.2 Modelagem do Componente Tutor de um Sistema Tutor Inteligente (STI)’.

Tabela 7: Sessões do Aluno no Sistema

ID aluno	**ID sessão**	**Data**	**Hora Início**	**Hora Fim**

[id aluno]	[id sessão]	[data DD/MM/ANO]	[Hora hh:mm]	[Hora hh:mm]

As estruturas abordadas anteriormente são utilizadas para construir a estrutura das sessões do aluno no sistema, conforme a Tabela 7, que contém seus respectivos identificadores, instantes de tempo e material consultado pelo aluno. Uma sessão vem a ser o conjunto de interações que o aluno realizou dentro do sistema, incluindo materiais instrucionais acompanhados, exercícios resolvidos e habilidades trabalhadas em um intervalo contínuo de tempo. Para que o sistema registre uma sessão do aluno, este deve se conectar no mesmo no início da interação e se desconectar quando desejar finalizar a sessão.

A ‘Tabela 8: Materiais Instrucionais Vistos pelo Aluno’ atua como tabela auxiliar. Ela registra os materiais que foram acompanhados pelo aluno ao longo das suas sessões com o sistema. Cada registro de sessão do aluno aponta para várias linhas nesta outra estrutura da ‘Tabela 8: Materiais Instrucionais Vistos pelo Aluno’.

Tabela 8: Materiais Instrucionais Vistos pelo Aluno

ID Aluno	**ID sessão**	**Tipo de Material Utilizado pelo Aluno**	**ID Material**
[id aluno]	[id sessão]	[Exercício/Explanação/Exemplo]	[id material tabelas Explanação/ Explicação/Exemplos]

.

O diagrama a seguir, representado pela ‘Figura 19: Diagrama ER do Modelo de Aluno’, aborda as relações entre as tabelas descritas anteriormente.

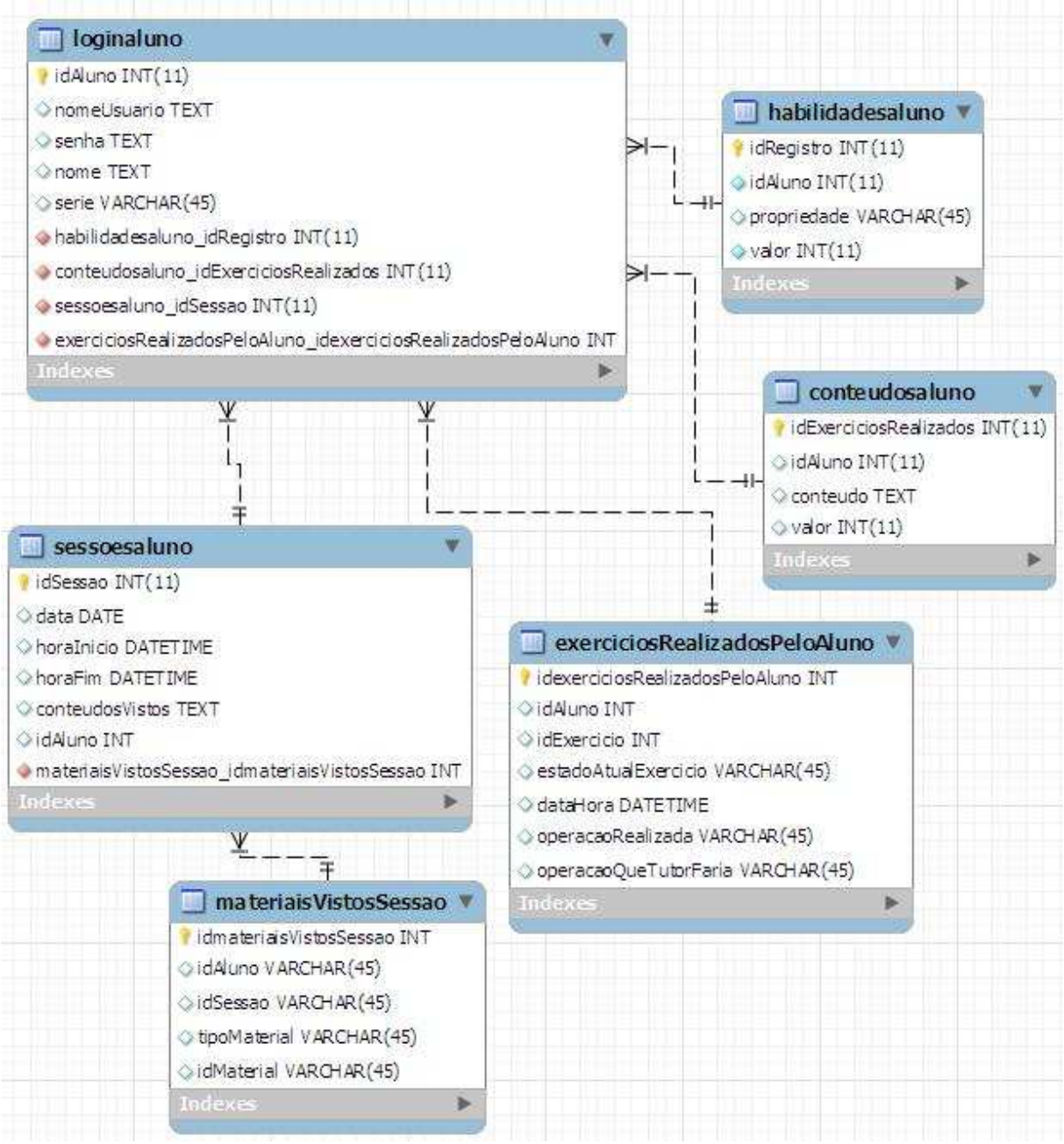

Figura 19: Diagrama ER do Modelo de Aluno

8.1.2 Diagnóstico e Levantamento das Informações do Aluno

O Agente de Modelo Cognitivo pode informar quais habilidades que o aluno está utilizando para resolver os exercícios propostos, sendo estas mapeadas no Modelo de Aluno. O aluno realiza uma série de vários exercícios que envolvem estas habilidades.

X+4 = 0; <Equação Inicial>
X = 0 – 4; <Operação Inversa>
X = - 4; <Resultado>

Figura 20: Exemplo de Resolução de Equação

Em um cenário em que o estudante esteja trabalhando no exercício “x+4=0”, conforme a ‘Figura 20: Exemplo de Resolução de Equação’, ele pode aplicar a operação inversa entre os termos “+4” e “0”, chegando ao resultado “x = -4”, resultando na correta aplicação da habilidade. Se o resultado informado por ele for “x = 4” o AMC informaria o erro cometido e informaria ao Agente Tutor para que isso fosse registrado e se tomasse as providências necessárias, no caso apresentando um material sobre operações inversas e ofertando um novo exercício para o aluno praticar. O exemplo abaixo ilustra uma mensagem destas, segundo a estrutura abordada na Figura 17: Template da Mensagem. No caso o aluno “Paulo” aplicou a operação inversa na equação “x+4=0” obtendo como solução a equação “x=-4”.

IA:Paulo #IC:Operação_Inversa #EI: x+4=0 #OP:Operação_Inversa #RT: x=-4

No caso do aluno aplicar corretamente tais habilidades ao longo dos exercícios propostos de um conteúdo incrementa-se o valor que corresponde a certeza de domínio por parte do aluno, como foi explicado no exemplo de um aluno hipotético anteriormente. Em caso negativo, pode-se saber então quais habilidades o aluno está aplicando de forma incorreta e informar o Agente Tutor para que altere o plano de ensino vigente para remediar o aprendizado incorreto.

8.2 AGENTE TUTOR

Este agente possui três principais funções: aplicar estratégias pedagógicas, seguir um plano de ensino padronizado e customizar este para o aluno em caso de erro, constituindo o processo de ensino do sistema.

8.2.1 Aplicação dos Planos de Ensino

Um plano de ensino contém uma estrutura linear que define a ordem em que os conteúdos devem ser expostos ao aluno em um Sistema Tutor Inteligente. Esta estrutura, no PAT2Math, é fornecida por uma ontologia, construída para definir a relação entre os conteúdos a serem ensinados. Tal modelagem hierárquica deste plano de ensino segue os Padrões Curriculares Nacionais (PCN), o que o torna adaptado aos conteúdos escolares brasileiros. Este plano de ensino contém em sua estrutura todos os tópicos[1] que podem ser trabalhados com o aluno, sendo necessário saber em que ponto situar o aluno para construir então as lições que constituirão o conteúdo a ser trabalhado.

Neste cenário entram as estratégias pedagógicas, que utilizam as informações contidas no Modelo de Aluno para tomar uma decisão sobre como ensinar, ou seja, de que forma ofertar um determinado conteúdo. A aplicação desta estratégia pedagógica, no caso a *Macro Adaption* (CORBETT e ANDERSON, 1995; SHUTE, 1993), acontece para mostrar conteúdos apropriados para os alunos. Através dessa informação o Agente Tutor poderá selecionar um ponto do plano de ensino padrão e o seguir apropriadamente, o fazendo selecionar materiais instrucionais e exercícios que irão compor as lições. Isto é feito considerando as respectivas habilidades de resolução de equação do aluno, assim como as informações sobre os tópicos que ele domina ou possui dificuldades em aprender. Em caso de erro por parte do aluno, este agente customiza o plano de ensino e oferece material instrucional referente ao erro cometido.

Para concluir a sua função inicial, o Agente Tutor precisa saber como expor este conjunto selecionado de conteúdos ao aluno, fazendo uso das Teorias de Design Instrucional (TDI) para tal. Esta teoria define a ordem dos tipos de material instrucional, de cada conteúdo, a serem expostos ao aluno.

Através das diferentes maneiras de como acontece o ensino, já teorizadas por Bloom (1956, 1986) e Merril (2001, 2002, 2007), percebe-se que o processo de ensino de um conteúdo pode variar consideravelmente. Nos tópicos a seguir são detalhadas as variáveis que compõem a tomada de decisão do Agente Tutor quanto ao ensino do aluno bem como de que

---

[1] A denominação ‘tópico’ se aplica aos conteúdos que estão disponíveis para o aluno trabalhar. Um tópico representa um destes conteúdos, que possuem no Modelo de Domínio seus respectivos ‘Materiais Instrucionais’, segundo a Teoria de Design Instrucional de Merril.

maneira são customizados os planos de ensino para um determinado aluno. Fatores afetivos, como emoção do aluno através de reconhecimento facial (OLIVEIRA, VICARI, *et al.*, 2008), e demais estratégias pedagógicas para mostrar conteúdos e exercícios aos alunos (RUBI, 2010) que farão parte do Agente Tutor, são trabalhos correntes de outros pesquisadores. O foco deste trabalho está na aplicação da *Macro-Adaption*, seguir plano de ensino padrão definido em ontologia e remediação de *misconceptions* quando acontece algum erro na resolução de exercícios, através da customização deste plano padronizado.

8.2.2 Ontologia do Plano de Ensino

O Agente Tutor segue um plano de ensino pré-determinado, seguindo a estrutura de uma ontologia de conteúdos que podem ser expostos ao aluno. Esta contém uma estrutura linear customizada a partir dos PCN (Parâmetros Curriculares Nacionais) (MINISTÉRIO DA EDUCAÇÃO/SECRETARIA DO ENSINO FUNDAMENTAL, 1998; MINISTÉRIO DA EDUCAÇÃO, SECRETARIA DE ENSINO FUNDAMENTAL, 2000; FIORENTINI e LORENZATO), assegurando a sua adaptação às sala de aula nacionais.

A estrutura da ontologia contém três tipos gerais de classes, sob as quais os tópicos, habilidades e *misconceptions* estão especificadas e relacionadas através de suas instâncias, conforme a Figura 21: Relação entre Classes da Ontologia:

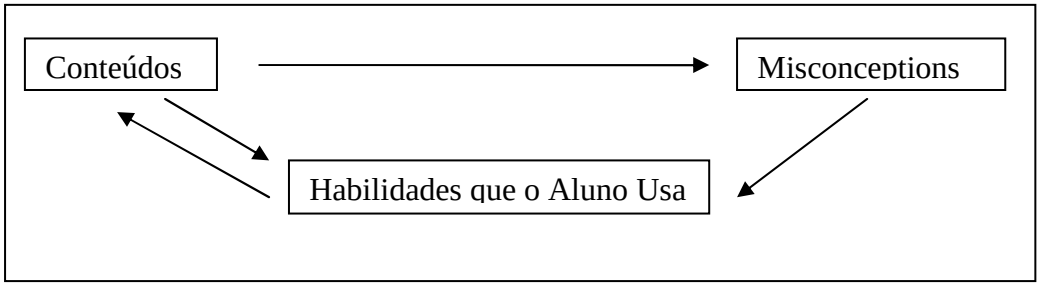

Figura 21: Relação entre Classes da Ontologia

A classe ‘Conteúdos’ representa os tópicos que o aluno pode aprender dentro do sistema, possuindo uma lista de possíveis *misconceptions* que servirão como fonte de consulta para o Agente Tutor quando o aluno cometer algum erro em um exercício correlato. A classe ‘Conteúdos’ possui uma estrutura hierárquica, na qual cada conteúdo possui um conteúdo pai e um conteúdo filho, organizando desta forma os passos previstos pela estrutura dos PCN. Cada conteúdo também possui uma relação que define as habilidades que o aluno exercita ao

trabalhá-los, sendo muito importante na consulta do Agente de Domínio para seleção de materiais instrucionais apropriados, bem como para o Agente Modelo de Aluno na atualização do Modelo de Aluno.

A classe ‘Misconceptions’ possui uma lista de habilidades relacionadas à falsa concepção, sendo consultada no momento de procurar um material instrucional que vá remediar o aprendizado incorreto. Por sua vez, estas habilidades remetem aos tópicos da classe ‘conteúdos’, sendo necessária esta consulta em caso de erro recorrente e devida atualização destas habilidades do aluno no seu respectivo Modelo de Aluno, no momento em que o aluno aprende e corrige o seu aprendizado.

Essa estrutura ontológica permite a descoberta, a partir de um conhecimento que o aluno está trabalhando, de quais conteúdos o aluno teve de aprender para utilizá-lo (através da hierarquia proveniente dos PCN e representada no trabalho pelas ontologias), assim como quais habilidades está exercitando (pela lista de habilidades relacionadas), quais poderá utilizar aprendendo-o de fato (também pela hierarquia PCN) e o que pode ter levado tal aluno a errar (consultando a lista de *misconceptions* relacionadas ao tópico vigente).

Em caso de algum erro ser cometido por parte do aluno, o plano de ensino individualizado é criado, levando em conta o que este aluno sabe em determinado momento e o que o levou a cometer o engano, constituindo o caráter individual de ensino por parte do Agente Tutor. A criação deste plano acontece com a customização do plano pré-determinado, descrito no parágrafo anterior. No caso, a ontologia é consultada para descobrir quais habilidades o conteúdo envolve e quais conteúdos estas remetem, mostrando então um material relacionado. Em caso do erro acontecer novamente, a ontologia é consultada para descobrir um conteúdo prévio cujo aprendizado incorreto pode ser a causa do erro atual. Por exemplo, se o aluno comete um erro em um conteúdo sobre potência e este erro persiste, a causa pode estar no aprendizado errôneo sobre multiplicação.

Não acontecendo o erro novamente, segue-se o plano pré-determinado novamente. Conforme o aluno interage, as informações abordadas na seção ‘8.1 Agente do Modelo de Aluno’ são alteradas, pois este está lidando com materiais instrucionais e exercícios. A ‘Figura 22: Ontologia de Plano de Ensino’ mostra a estrutura desta ontologia comentada anteriormente. Dentro da classe ‘conteúdos’ está a hierarquia de tópicos que o aluno deverá seguir. No caso, inicialmente é apresentado o conteúdo sobre Números, seguindo com Números Naturais, posteriormente apresentando operações e problemas com os mesmos, e assim por diante.

Os nomes das classes, no caso habilidades, *misconceptions* e conteúdos, possuem os mesmos nomes que as linhas das tabelas Tabela 3: Habilidades Aluno e Tabela 4: Conteúdos Aluno respectivamente, do Modelo de Aluno comentando anteriormente, para que a identificação de um termo na ontologia já permitisse a consulta ou atualizações na estrutura correspondente às informações do aluno.

Figura 22: Ontologia de Plano de Ensino

Nesta ontologia encontram-se três tipos de classes: conteúdos, habilidades e *misconceptions*, conforme descrito na 'Figura 21: Relação entre Classes da Ontologia'. Cada uma destas possui uma instância, que pode conter um número de propriedades, utilizadas para

descrever as relações existentes entre os elementos conforme citado anteriormente. A ‘Figura 23: Propriedades da Ontologia de Planos de Ensino’ mostra três tipos de propriedade que uma instância pode ter, que a relaciona com as classes da ontologia. Isso resulta em uma instância com diversas propriedades, descrevendo relações da mesma com as demais presentes na estrutura ontológica. Um exemplo desta está na ‘Figura 24: Exemplo de Instância de Classe da Ontologia’.

Figura 23: Propriedades da Ontologia de Planos de Ensino

Figura 24: Exemplo de Instância de Classe da Ontologia

8.2.3 Aplicação das Teorias de Design Instrucional (TDI) para apresentação do Material Instrucional

No momento em que o Agente Tutor selecionou um conteúdo a ser mostrado para o aluno em questão, a exposição do mesmo seguirá as definições abordadas na seção ‘3.2 Estágios de Aprendizagem (Taxonomia de bloom e Teorias de Design Instrucional)’. Inicialmente o conteúdo será exposto na forma de um *Problema*, pois o aprendizado é promovido quando aprendizes estão engajados em resolução de problemas do mundo real. A

sequência continua com um material que efetive a *Ativação,* pois o aprendizado é promovido quando o conhecimento existente é ativado como base para um novo conhecimento. Posteriormente acontece a *Demonstração,* em que o respectivo aprendizado é promovido quando o novo conhecimento é demonstrado ao aprendiz. Isso serve como base para a *Aplicação,* momento em que o aprendizado é promovido quando o novo conhecimento é aplicado pelo aprendiz, para encerrar com a *Integração,* integrando o material ao mundo do aprendiz.

Cada tópico dos conteúdos do Plano de Ensino a serem expostos está dividido nestas cinco categorias citadas anteriormente, representando os estágios de aprendizado pelos quais o aluno vai passar ao interagir com os materiais. No Modelo de Domínio os materiais instrucionais possuem uma coluna que identifica em qual estágio da TDI o material se enquadra, conforme descrito na

8.2.4 Adaptação do Plano de Ensino segundo a teoria Ausubeliana de Aprendizagem Significativa

A ontologia descrita anteriormente na seção ‘8.2.2 Ontologia do Plano de Ensino’ possui relações entre as instâncias de suas respectivas classes, apontando dependências e relações entre os tópicos, habilidades e *misconceptions* que um aluno pode ter. As classes de conteúdo presentes nesta estrutura podem ser vistas como os conceitos subsunçores que um aluno pode ter acerca dos conteúdos abordados, e estas relações entre suas instâncias concretizam a interferência de um conteúdo nos demais presentes, permitindo que atividade relacionada a um afete um conjunto de demais saberes, mapeados no Modelo de Aluno.

A ‘Tabela 9: Exemplo de Relações entre classes da Ontologia’ exemplifica como acontece este tipo de relação entre as classes. Observando a coluna ‘tópico’, pode-se ver o tópico ‘potência’, que está relacionado com a habilidade ‘multiplicação’ e com a *misconception* ‘errar regra de sinais’. Se for observada a classe ‘habilidade’, na multiplicação percebe-se a relação com o tópico ‘potência’ bem como a *misconception* ‘errar regra de sinais’. Por fim, se a classe de *misconceptions* for observada, é visível a sua relação com a habilidade de ‘multiplicação’ e com o tópico ‘potência’. Os ‘x’ presentes na tabela representam as linhas e colunas do mesmo tipo de classe, não podendo ser relacionadas.

Tabela 9: Exemplo de Relações entre classes da Ontologia

	Tópico	**Habilidade**	**Misconception**
	Potência	*Multiplicação*	*Errar Regra Sinais*
Habilidade	*Multiplicação*	*x*	*Multiplicação*
Misconception	*Errar Regra Sinais*	*Errar Regra Sinais*	*x*
Tópico	*x*	*Potência*	*Potência*

8.2.5 WorkFlow de Ensino no Agente Tutor

O sistema indica o estudo de determinados conteúdos (segundo a teoria de Macro-Adaption) e oferta exercícios para que o aluno exercite tal conhecimento e suas respectivas habilidades necessárias. Ao aluno interagir com o sistema, através do botão ‘Conteúdo’ (descrito na seção “Implementação” deste documento) ocorrem os seguintes passos, que estão dispostos no Diagrama da Figura 25: Diagrama de Caso de Uso Interação Aluno:

1. Aluno faz login no sistema;
2. Sistema valida o login;
3. Verificam-se as habilidades do aluno;
4. Aluno requisita conteúdo;
5. Verifica-se o grau de proficiência percentual nos conteúdos;
6. Verificam-se quais conteúdos possuem um valor abaixo de determinada porcentagem (efetivando a teoria de Macro-Adaption);
7. Se faz a leitura da Ontologia de Conteúdos;
8. Verifica-se o estágio TDI do último material instrucional apresentado referente ao conteúdo atualmente sendo trabalhado (Problema / Ativação / Demonstração / Aplicação / Integração);
9. Seleciona-se o material instrucional para o conteúdo a ser trabalhado, que se enquadra dentro da próxima categoria de Teoria de Design Instrucional (obedecendo a ordem Problema/Ativação/Demonstração/Aplicação/Integração);
10. Realiza-se a leitura da ontologia sobre quais habilidades são trabalhadas com este conteúdo (para atualização no Modelo de Aluno);

10.1. Selecionam-se os exercícios referentes a este material;

10.2. Verificam-se os que já que foram realizados;

10.3. Seleciona-se aleatoriamente um dos que sobraram na lista;

10.4. Aluno realiza o exercício;

10.4.1. Se algum erro foi cometido seleciona-se material instrucional a respeito da *misconception* correlata, que remete a habilidades que foram usadas incorretamente;

10.4.2. Oferece-se outro exercício caso o aluno queira resolver;

10.4.3. Caso ocorra o mesmo erro novamente, a ontologia é consultada através da implementação baseada na OWL API para saber qual a *misconception* atual, que remete a um conteúdo que pode ter sido aprendido de maneira errada, ser a causa do erro;

10.4.4. Seleciona-se este conteúdo referente a *misconception* que ocorreu e o apresenta ao aluno;

10.4.5. Caso o aluno volte a errar, sobe-se um nível na hierarquia de conteúdos e se oferece o material instrucional referente a este conteúdo em nível superior;

11. Atualiza-se no Modelo de Aluno o exercício realizado e habilidades que foram utilizadas;

12. Retorna ao passo cinco.

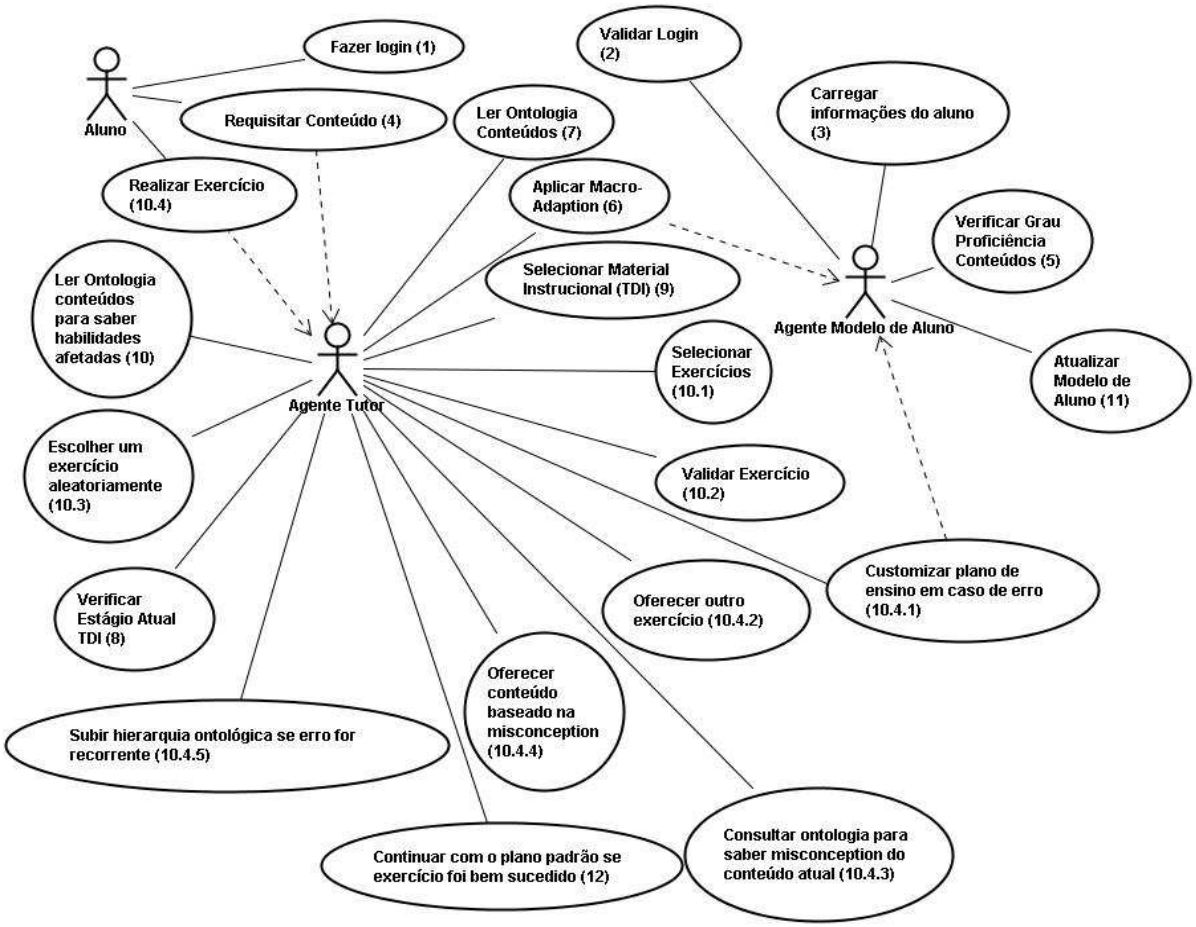

Figura 25: Diagrama de Caso de Uso Interação Aluno

Os princípios da Aprendizagem Significativa, vista na seção Aprendizagem Significativa de Ausubel (AUSUBEL, 1982), *Macro-Adaption* de Corbett, Anderson & Shute (CORBETT e ANDERSON, 1995; SHUTE, 1993) e Teorias de Design Instrucional (MERRIL, 2001; MERRIL, 2007; MERRIL, 2002) estão presentes nesse processo. A teoria ausubeliana embasa o aprendizado do aluno e a utilização da ontologia como estrutura que representa os conceitos subsunçores que o aluno vai desenvolver. A teoria de *Macro-Adaption* embasa o processo de interação do sistema com o aluno, na verificação de habilidades que ele ainda não domina, variável decisiva no momento de escolher qual conteúdo deve ser trabalhado. Por fim, a teoria de Merril (Teoria de Design Instrucional) coordena a apresentação de materiais instrucionais referentes ao conteúdo sendo mostrado ao aluno, organiza uma ordem de exploração destes, consequentemente definindo grupos de exercícios-chave que podem ser realizados em cada passo do aprendizado.

Com esta lista de conteúdos, pode ser feita a consulta no Modelo de Aluno para descobrir de que forma o aluno interagiu com eles, se os visualizou ou exercitou. Se já foram

explorados previamente, textos introdutórios podem não compor o plano, partindo para uma abordagem mais expositiva com mais exemplos, por exemplo.

A customização deste plano de ensino considera as informações do aluno, tais como que exercícios realizou, além das informações presentes na ontologia de conteúdos, para que se descubra que habilidades o aluno está atualmente trabalhando e com que conteúdos estas se relacionam. Neste cenário entra o auxílio que o Agente Tutor pode oferecer ao aluno. Está dividido em duas categorias:

- *Feedback* imediato nos passos de resolução que o aluno faz nos exercícios;
- *Planos de Ensino* baseados no domínio do aluno inferido sobre os conhecimentos abordados.

Com o conteúdo selecionado, o Agente Tutor pode então seguir o plano de ensino pré-determinado a partir do ponto que melhor se aplique a este aluno, informando ao sistema de que forma as informações devem ser expostas, segundo as Teorias de Design Instrucional. Este plano de ensino tem um papel muito importante durante a apresentação de material instrucional, é ele quem informa o que deve ser exposto ao aluno. Quando o aluno posteriormente cometer erros durante a resolução de exercícios, entra a customização deste, conforme citado anteriormente.

8.3 IMPLEMENTAÇÃO

Este capítulo aborda a implementação do trabalho proposto, o protótipo construído com o objetivo de efetivar o modelo descrito ao longo deste trabalho, bem como as tecnologias utilizadas para tal.

O PAT2Math é composto por alguns módulos, já sendo desenvolvidos na linguagem de programação Java, então o Módulo Tutor e Modelo de Aluno deste trabalho também seguiram este padrão. As implementações realizadas para concretizar o protótipo envolvem:

- Criação e conexão com Banco de Dados em MySQL para o Modelo de Aluno (Com a interface MySQL Workbench);
- Projeto Java com OWL API (Leitor e interpretador de ontologias no formato OWL);
- Projeto em JDeveloper que mapeia o banco MySQL em classes Java;
- Projeto em JDeveloper que cria a interface do sistema, para o protótipo.

Projeto Java com OWL API: São as classes Java responsáveis pela leitura, modificação e interpretação da ontologia de conteúdos do trabalho proposto. Possui funções para:

- Extrair instância do próximo conteúdo na hierarquia ontológica;
- Extrair instância do conteúdo prévio referente a habilidade utilizada pelo aluno;
- Carregar ontologia a partir de um arquivo OWL;
- Carregar *reasoner* (motor de inferência para ontologias);
- Incluir instância em uma classe da ontologia;
- Incluir propriedade entre duas instâncias;
- Listar super classes;
- Listar subclasses;

Possui as seguintes classes:

Principal: Classe responsável por invocar o módulo tutor e chamar sua rotina de início.

Tutor: Classe responsável por chamar os métodos da OWL API que são necessários para realizar as operações na ontologia. Possui os seguintes métodos:

- *Inicia*: Inicia o módulo tutor, carregando uma ontologia a partir de um arquivo e iniciando o *reasoner*;
- *getInstanciaProximoConteudo*: Método que retorna a instância da classe de conteúdo seguinte, a partir de uma outra instância informada como argumento;
- *getProximoConteudo*: Método que retorna o nome do próximo conteúdo, se informado o nome de uma determinada classe na ontologia;
- *splitNomeClasse*: Método que extrai o nome da classe a partir de uma IRI completa;
- *getInstanciaSubsuncorDaInstanciaMisconception*: Método que retorna os nomes das instâncias de um conteúdo relacionado a uma misconception;
- *listaValoresDePropertyDeInstancia*: Método que lista valores de uma propriedade de uma instância, como por exemplo ‘conteudosRelacionados’;
- *listaInstanciasDeClasse*: Método que, a partir de um nome de uma classe da ontologia, extrai todas as suas respectivas instâncias;
- *listaSubClasses*: Método que, a partir do nome de uma classe da ontologia, retorna uma lista com os nomes das suas respectivas subclasses;

- *carregaOntologiaDeArquivo*: Método que, ao informado um endereço de IRI, carrega a ontologia em questão;
- *incluiInstanciaDeClasseNaOntologia*: Método que inclui uma instância a partir de um nome de classe presente na ontologia;
- *incluiPropertyEntreDuasInstancias*: Método que permite a relação entre duas instâncias, com a criação de uma propriedade entre elas.

Para a implementação destes métodos, a classe Tutor conta com os seguintes atributos, herdados da própria OWL API:

- OWLOntology ontoFile;
- OWLOntologyManager manager;
- SimpleIRIMapper mapper;
- OWLDataFactory owlDatafactory;
- OWLReasoner reasoner;
- IRI ontologyIRI;
- IRI documentIRI;
- PrefixManager prefixManager;
- String stringBaseOntologia;
- File file;
- String IRIBase;

TutorAgent: Classe que representa o Módulo Tutor como um Agente. Trabalha em conjunto com a classe *TutorBehavior*, que descreve o seu comportamento segundo a tecnologia JADE. Possui os seguintes métodos:

- Setup: Método que seleciona um comportamento da classe *TutorBehavior* e define-o como padrão para execução.

TutorBehavior: Classe que contém os comportamentos disponíveis para o agente tutor, seguindo a tecnologia JADE. Possui os seguintes métodos:

- *Action:* Responsável por iniciar o agente tutor e controlar as mensagens que este envia aos demais agentes;
- *Done:* Método que encerra o agente.

Criação e conexão com Banco de Dados MySQL: As tabelas referentes ao modelo de aluno estão persistidas neste banco de dados, sendo necessária a conexão com o mesmo a partir das classes Java dos demais projetos para a devida atualização. A plataforma JDeveloper contém ferramentas que mapeiam bancos sendo executadas na máquina servidor, sendo possível criar a conexão de projetos com estes bancos através de pequenos *wizards* dentro da IDE. Com eles, uma camada adicional é criada entre o banco de dados em si e as classes na IDE que representam as tabelas e demais objetos a serem persistidos, com o uso da tecnologia *Java beans.*

Um número de tabelas foi criado para trabalhar em conjunto com o protótipo, fazendo o papel de Base de Domínio, ofertando materiais instrucionais, visto que tal agente não está implementado em sua totalidade. Os materiais contidos na sua estrutura serão mapeados na base OMDOC posteriormente, com trabalhos futuros de demais pesquisadores envolvidos no projeto PAT2Math. Informações relacionais entre estes materiais e exercícios também tornaram necessária a criação de tabelas auxiliares. Estão descritas e detalhadas a seguir:

Tabela 10: Materiais Instrucionais

Material Instrucional	**Conteúdo**
[id material instrucional]	[conteúdo do material]

Tabela 11: Exercícios/Materiais Instrucional

ID Exercício	**ID Material Instrucional**	**Equação**
[id exercício]	[id material]	[equação exercício]

Tabela 12: Conteúdos/Materiais Instrucionais/Teoria Design Instrucional

ID	**Nome Conteúdo**	**ID Material**	**Tipo Conteúdo**
[id]	[nome]	[id material instrucional]	[Tipo segundo teoria de design instrucional]

Tabela 13: Exercícios por Conteúdo

ID	Conteúdo	ID Exercício
[id]	[nome conteúdo]	[id exercício relacionado]

A ‘Tabela 10: Materiais Instrucionais’ contém os conteúdos a serem ofertados ao aluno, variando entre os tipos descritos pelas Teorias de Design Instrucional de (MERRIL, 2007). Já a ‘Tabela 11: Exercícios/Materiais Instrucional’ faz a relação destes materiais instrucionais com os exercícios que podem ser expostos ao aluno, bem como que exercícios estão relacionados com determinados materiais instrucionais. A ‘Tabela 12: Conteúdos/Materiais Instrucionais/Teoria Design Instrucional’ relaciona os materiais instrucionais com as categorias previstas pela Teoria de Design Instrucional de Merril, em que cada um destes pode estar enquadrado como *problema*, *ativação*, *demonstração*, *aplicação* ou *integração.* A ‘Tabela 13: Exercícios por Conteúdo’ organiza os exercícios disponíveis relacionando-os com os conteúdos que podem ser expostos ao aluno. O diagrama ER de tais tabelas é o seguinte:

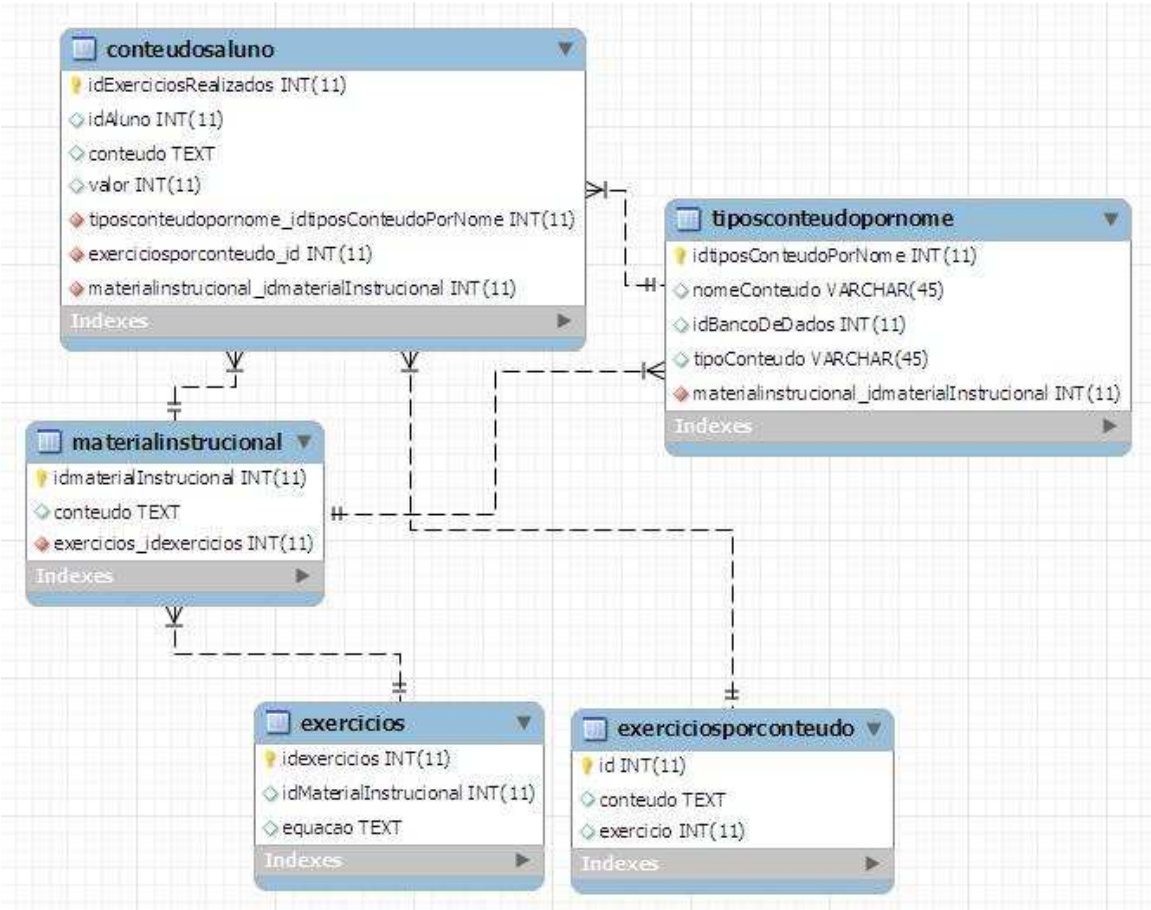

Figura 26: Diagrama ER Tabelas Auxiliares que representam o Domínio do sistema

Neste diagrama percebe-se que cada conteúdo que o aluno pode ter trabalhado no sistema interage com uma série de tabelas, descritas anteriormente. No caso, cada conteúdo terá sua lista de materiais de instrucionais e exercícios. Materiais Instrucionais também podem ter sua vez terá sua lista de exercícios, pois podem se referir a habilidades do aluno ou misconceptions a ser trabalhadas. Estes materiais instrucionais estão categorizados segundo as TDIs, conforme a ‘Tabela 12: Conteúdos/Materiais Instrucionais/Teoria Design Instrucional’.

Projeto em JDeveloper que mapeia o banco MySQL em classes Java: Projeto resultante dos *wizards* comentados anteriormente. Trata-se do projeto que serve de ponte para a atualização física dos dados no banco de dados selecionado. Os demais projetos alteram as classes deste, que chama a sua rotina de persistência para a devida atualização. Possui as seguintes classes:

- Classes Java que representam as tabelas e seus atributos e colunas do banco de dados;
- SessionEJBBean: Classe que contém os métodos para persistência e consulta nas tabelas do banco de dados em MySQL, utilizando as classes do tópico acima para tal. No caso os valores de atributos são alterados nestas classes inicialmente, para posteriormente a função de persistência da IDE JDeveloper ser chamada e ocorrer a atualização física do banco em questão.

Projeto em JDeveloper que cria a interface do sistema: Trata-se da parte visual, a interface do protótipo com que os avaliadores puderam expressar suas opiniões a respeito do projeto. Consistido basicamente de páginas JSP e seus respectivos *backing beans* que invocam as funções pertinentes do projeto, como leitura de ontologias, validação de *login*, entre outras operações com banco de dados por exemplo. A ‘Figura 27: Interface de Login do Sistema’, mostrada abaixo, consiste na primeira interação do aluno com o sistema, realizando o seu *login* através dos campos de ‘usuário’ e ‘senha’. O botão ‘Entrar’ chama a função pertinente que valida estes dados e permite a entrada do aluno no sistema, de acordo com seu respectivo modelo de aluno.

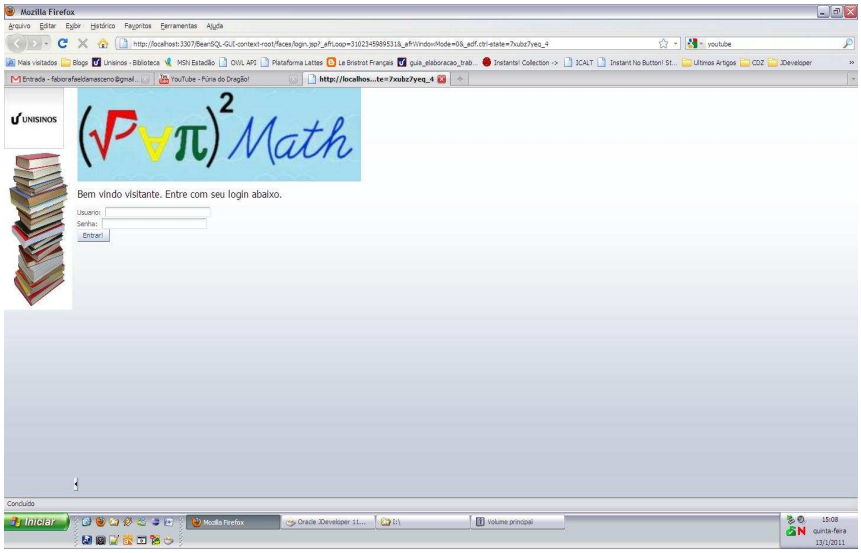

Figura 27: Interface de Login do Sistema

A ‘Figura 28: Interface de exploração de conteúdo’ representa a interface que o estudante dispõe para explorar conteúdos. Possui um botão ‘Conteúdo’ para que seja mostrado o próximo material instrucional ao aluno, segundo as Teorias de Design Instrucional abordadas anteriormente neste documento. O botão ‘Continua’ efetiva a conclusão do exercício realizado na interface de resolução de equações, verificando se algum passo errado foi cometido e mostrando um material de acordo com tal. O botão ‘*Logout’* permite ao aluno sair do sistema.

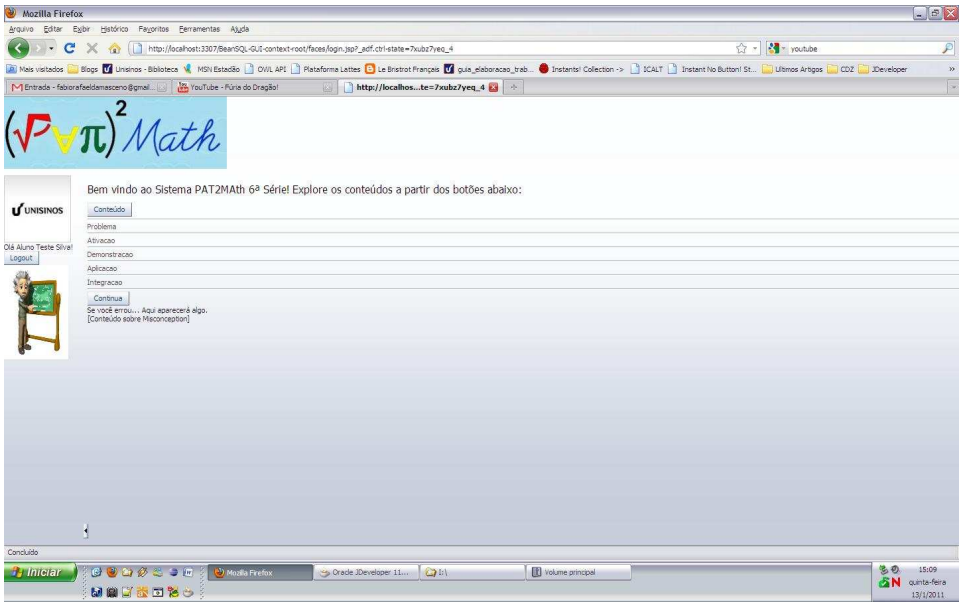

Figura 28: Interface de exploração de conteúdo

8.4 EXEMPLO DE ESTUDO DE CASO

Nesta seção é ilustrado o caso de um aluno hipotético aprendendo um conteúdo sobre equações e resolvendo uma equação-problema que o sistema oferta para exercitar as suas respectivas habilidades. Acontecem os seguintes passos, sumarizados na 'Figura 25: Diagrama de Caso de Uso Interação Aluno' e descritos abaixo:

- O aluno acessa o endereço em que o sistema PAT2Math está hospedado, através do seu navegador;
- Ao carregar a página, são mostrados os campos 'usuário' e 'senha', em que o aluno põe seu nome de usuário e senha para entrar no sistema. Por exemplo, usuário 'joao' e senha 'minhaSenha'. Tais informações servem para definir em que tabelas do modelo de aluno os dados da sessão com o sistema devem ser inseridos e/ou alterados;
- Ao ser validado o *login* deste aluno, entra-se no sistema, carregando outra página em que o botão 'Conteúdo' está disponível para ser utilizado;
- No momento em que tal botão é pressionado, o sistema verifica as habilidades do aluno, aplicando a estratégia de *Macro-Adaption*, mostrando então o material instrucional sobre o conteúdo que é necessário o aluno aprender. No cenário sendo descrito neste caso de uso, a estratégia selecionou o conteúdo de 'Equações' para ser trabalhado com o aluno;
- No primeiro instante, será mostrado o material instrucional de categoria 'Problema' ao aluno, sendo exibido este texto: "*Introduzir o conceito de equações algébricas, em especial, equações do 1º grau com uma incógnita de maneira intuitiva. E resolver equações do 1º grau.*"
- Após acompanhar o material ofertado pelo sistema, o aluno pode realizar o exercício proposto junto a interface do PATEquation, que se enquadraria na categoria 'Problema' da teoria TDI, por exemplo "2x + 6 = 10";
- Com esta equação de exemplo a ser resolvida, o aluno tem uma lista de operações a aplicar no problema, dispostas em botões na interface. Neste exemplo o aluno pode utilizar a *operação inversa* no termo "6", resultando na equação "2x=10-6". Com este passo realizado no sistema, o aluno pede a validação através de um botão de checagem do PAT2Math, o botão 'Continua', que mostra visualmente que o passo está correto;

- Após, o aluno realiza a subtração dos números inteiros “10” e “6”, que resultaria na equação “2x = 4”;
- Agora a equação-problema é “2x = 4”, exigindo que o aluno aplique novamente uma operação inversa no termo 2x e realizando diretamente a divisão, resultando em “x = 2”. Em caso de erro, se o aluno colocasse como resposta “x = 1”, por exemplo, o sistema, ao verificar o passo, mostra na interface dos materiais instrucionais acerca do conteúdo, um material referente à *misconception* cometida e suas respectivas habilidades que precisam ser melhor trabalhadas;
- No cenário em que a resolução do exercício está correta, a ativação do botão ‘conteúdo’ traz ao aluno um material instrucional que avança seus estudos no que se está trabalhando, no caso um material segundo a categoria TDI ‘Ativação’. O texto a ser exibido seria: *“A Álgebra é um campo da Matemática que utiliza letras (incógnita ou variável) para representar números ainda não descobertos. Os elementos da álgebra são os números e as letras. E o estudo das equações faz parte da álgebra.”*
- Um novo exercício é ofertado por sua vez, avançando um tópico nas Teorias de Design Instrucional, sendo então mostrado ao aluno um material TDI de categoria “Demonstração”. O material exibido seria este: *“Equações Algébricas são igualdades que contém números desconhecidos, geralmente representados por letras. Para compreendermos as manipulações necessárias para a resolução de uma equação devemos pensar em uma balança de dois pratos. Assim, para termos o seu equilíbrio, ou seja, para que os dois pratos fiquem na mesma altura (iguais) precisamos ter o mesmo peso dos dois lados. Então, na nossa equação algébrica temos o “x” que representa este valor que irá equilibrar. Neste sentido, a letra “x” é a Variável e/ou Incógnita da equação que pode ser representada por qualquer outra letra ou símbolo que represente o número desconhecido da equação. Normalmente são utilizadas as letras x, y, z, a, b, entre outras. Mas todas representam o número a ser descoberto”*.
- Um novo exercício é oferecido ao aluno, que ao concluir com sucesso avança no seu estudo fazendo com que o sistema mostre um material instrucional do tipo ‘Aplicação’, segundo as Teorias de Design Instrucional. O material apresentado seria o seguinte: *“Pensando matematicamente como resolver uma dada equação: 2x+30=44 (Subtraímos 30 dos dois lados da igualdade) // 2x+30-30=44-30 (Realizamos as subtrações) // 2x=14 (Dividimos por 2 os dois lados da equação) // 2x/2=14/2 (Realizamos as divisões) // x=7”*

- Outra equação seria apresentada ao aluno para que o mesmo trabalhasse a sua resolução. Obtendo sucesso, o material instrucional de categoria ‘Integração’ é apresentado, sendo exposto o seguinte texto ao aluno: *“As equações algébricas são utilizadas intuitivamente no nosso dia a dia, por exemplo: Adriana pesa 68 quilos. Juntas, ela e Laura pesam 125 quilos. Quanto pesa Laura? 68+x=125 >> x=125-68 >> x=57 >> Laura pesa 57 kg!”*.
- Mais um exercício é proposto ao aluno, que obtendo sucesso conclui os seus estudos neste tópico, podendo avançar no seu plano de ensino e sendo atualizadas suas informações no seu respectivo Modelo de Aluno.

9. AVALIAÇÃO

Percebe-se que a característica multiagente está presente nos sistemas avaliados previamente e que se trata de uma alternativa interessante para modelar funcionalidades em um sistema de ensino, o PAT2Math também possui esta característica, aqui reside o porque desta avaliação. Mais precisamente, falando do ensino, estes sistemas oferecem um apoio individualizado, mostrando importância das características cognitivas de cada aluno e o do ensino individual defendido por autores como Bloom e demais pesquisadores da área de STIs. O PAT2Math contém um Agente de Modelo de Aluno, que monitora estas características e um Agente Tutor, que é responsável por monitorar tais dados e considerá-los na hora de definir ou adaptar o plano de ensino para um determinado aluno. De uma maneira geral os sistemas avaliados anteriormente permitem a interação do aluno livremente, oferecendo auxílio de maneira não-intrusiva, em níveis diferentes. Este tipo de *feedback* gradual e diferenciado está presente no PAT2Math da mesma forma, pois o aluno pode requisitar ajuda livremente, além do sistema mostrar material relacionado a um erro cometido no mesmo momento em que este acontece.

Este capítulo compreende a avaliação realizada do trabalho proposto, conforme está a seção ‘8. TRABALHO PROPOSTO’, seguindo princípios abordados na seção ‘2.5 Avaliação Qualitativa’. Tal avaliação envolve a avaliação do protótipo construído para sua concretização, através de uma avaliação do tipo qualitativa envolvendo quatro avaliadores, cuja formação acadêmica é Mestre (três integrantes) e Bacharel (um integrante). As informações sobre a formação acadêmica dos avaliadores compreendem:

- Dois dos Mestres e a Bacharel possuem raiz acadêmica na Graduação em Matemática;
- Dois Mestres são da área de Matemática Aplicada;
- Um Mestre possui formação acadêmica na Psicologia e no seu Mestrado estudou uso de jogos e softwares educativos para ensino;
- Os avaliadores de formação Matemática possuem ampla experiência com docência para alunos do Ensino Fundamental.

Estes interagiram com o protótipo desenvolvido e posteriormente responderiam a um questionário a respeito das características do projeto, customizado a partir do trabalho proposto em (GLADCHEFF, 2001), conforme se encontra no Apêndice I:

A seção de *vocabulário* foi retirada da análise pois não são consideradas pertencentes ao escopo do trabalho tais questões linguísticas e gramaticais. As questões *TUT 7 e 8* não estão sendo levadas em consideração pois serão trabalhadas por futuras pesquisas no desenvolvimento do base de domínio do PAT2Math. O questionamento abordado em *TUT 11* foi retirado por envolver uma reflexão que não faz parte do escopo do trabalho proposto, assim como as *TUT 14, 15 e 16* foram removidas da análise pois focam as questões de inclusão e definição de materiais do domínio. Os pontos abordados em TUT 18, 19, 20 e 21 abordam as diferentes categorias de exercícios que podem estar presentes no sistema. No caso do protótipo construído o foco está nas equações algébricas, e não em demais categorias, o que invalida o porque de sua avaliação. As categorias *Apresentação de Problemas* e *Usabilidade (exceto TUT 27)* estão fora do escopo do trabalho proposto, pois abordam a forma de apresentação dos materiais do domínio e questões de interface, que não foram consideradas no desenvolvimento do protótipo. Sendo assim, o questionário empregado é o apresentado na ‘

9.2 Avaliação dos Resultados’.

9.1 RESULTADOS

Após realizar a avaliação, obtiveram-se os seguintes resultados:

Tabela 14: Sumarização dos Resultados

Questão	**Resposta**			
TUT.5	SIM: 4		OUTRAS: 0	
TUT.6	SIM: 4		OUTRAS: 0	
TUT.7	SIM: 4		OUTRAS: 0	
TUT.10	SIM: 1	QUASE: 1	POUCO: 1	BRANCO: 1
TUT.12	SIM: 4		OUTRAS: 0	
TUT.13	SIM: 4		OUTRAS: 0	
TUT.17	SIM: 4		OUTRAS: 0	
TUT.22	SIM:2		QUASE INTEIRAMENTE: 2	
TUT.23	SIM: 4		OUTRAS: 0	
TUT.27	SIM: 4		OUTRAS: 0	

TUT.28	PISTA: 4		OUTRAS: 0
TUT.29	SIM: 2		QUASE INTEIRAMENTE: 2
TUT.30	SIM: 4		OUTRAS: 0
TUT.31	SIM: 4		OUTRAS: 0
TUT.32	SIM: 4		OUTRAS: 0
TUT.32.1	SIM: 3		QUASE INTEIRAMENTE: 1
TUT.33	SIM: 2	QUASE: 1	POUCO: 1
TUT.34	SIM, COM POUCAS INFORMAÇOES: 4		OUTRAS: 0
TUT.35	SIM: 4		OUTRAS: 0

A avaliação destes resultados obtidos com o questionário aplicado com os avaliadores encontra-se na seção seguinte.

9.2 AVALIAÇÃO DOS RESULTADOS

A interação dos avaliadores com o protótipo ocorreu de forma agradável, no laboratório educacional do ambiente de Pós-Graduação em Informática da Educação (PGIE) na Universidade Federal do Rio Grande do Sul (UFRGS). A idéia de uma ferramenta que visa contribuir no ensino-aprendizagem de Álgebra motivou os pesquisadores a participar da avaliação para evoluir o estudo de possibilidades futuras e dar um fechamento a este trabalho de Mestrado. Houve interação e discussão entre eles conforme utilizavam o protótipo, buscando entender a proposta e pensar em possibilidades de aplicação reais com suas respectivas turmas de alunos.

Após realizar a avaliação do protótipo com o grupo de avaliadores comentados na seção anterior, percebe-se que houve uma tendência em respostas positivas para os questionamentos propostos. Isto, segundo (GLADCHEFF, 2001), indica um forte valor de qualidade no protótipo construído, efetivando o seu valor educacional. De dezenove questões aplicadas, em treze ouve resultado unanimemente positivo, a favor da qualidade do protótipo, um valor próximo a 70% (68,5% do total). Nas demais questões houve uma maior distribuição de opiniões, mas estas sempre iguais ou menores ao valor de respostas positivas apontadas pelos avaliadores.

Os avaliadores acreditam que o protótipo é uma ferramenta muito bem organizada e de fácil exploração para o estudante independente do seu amadurecimento de conteúdo quanto à Matemática, havendo possibilidade de maiores explorações conforme o próprio professor inserir conteúdos e modificar a base de domínio do sistema com mais exercícios e materiais, por exemplo. Em geral, foi considerada boa a sua metodologia de uso tanto em sala de aula como extraclasse, destacando-se que o número de exercícios que o estudante deseja fazer é uma opção interessante e importante ao processo de aprendizagem do aluno, ficando como sugestão de variável sob algum controle do mesmo no protótipo. Por exemplo, o sistema pode acreditar que o aluno tem domínio de um conteúdo, mas se por ventura o aluno não se sentir seguro, deve poder realizar quantos exercícios desejar até sentir segurança para avançar em seus estudos.

Foi ressaltado pelos avaliadores que é interessante observar que a utilização de uma ferramenta, como este protótipo, proporciona uma forma diferenciada de aprendizado, com diversas possibilidades através da interface, considerada clara e organizada. Esta diversidade de opções, vista por eles como uma tentativa de atender todos os modos de aprendizagem busca mobilizar a aprendizagem de todos os estudantes de Matemática, resultando na consideração deste trabalho por eles muito relevante.

Também foi salientado o ponto que por ser uma ferramenta de informática este protótipo não vem a se equiparar a mesma interação que ocorre pessoalmente entre o professor e o aluno. Para isso, complementar a esta seria o professor observar quando possível as ações dos estudantes para obter um pouco mais de informações sobre o seu respectivo desempenho e aprimorar o protótipo, cenário em que a utilização acompanhada em sala de aula se torna mais desejável.

10. CONSIDERAÇÕES FINAIS

Esta seção do trabalho aborda o estado atual da pesquisa deste trabalho, possibilidades futuras, bem como quais são os próximos objetivos.

Foi realizada uma pesquisa sobre o estado da arte em Sistemas Tutores Inteligentes, através da avaliação dos trabalhos de (ANDERSON, CORBETT, *et al.*, 1992; ANDERSON, BOYLE, *et al.*, 1990; CORBETT e ANDERSON, 1995; CORBETT, KOEDINGER e ANDERSON, 1997; MURRAY, 1999; VANLEHN, 2006) e seus representantes na área de ensino de Álgebra, como os descritos em (MELLIS, ANDRÈS, *et al.*, 2001; NICAUD, BITTAR, *et al.*, 2006; COHEN, BEAL e ADAMS, 2008; ANDERSON, BOYLE, *et al.*, 1990), assim como Web Semântica e Ontologias (DEVEDZIC, 2006; GRUBER, 1995; GUANGZUO, 2004; GUARINO, 1995; ISOTANI e BITTENCOURT, 2009; JACOB, 2003; LEE, HENDLER e LASSILA, 2001; RICHARDSON, 2006). Tal pesquisa apontou pontos fortes e limitações nos trabalhos correntes e forneceu uma base concreta para o desenvolvimento dos módulos propostos na seção anterior. Um estudo sobre ensino de Álgebra e suas características foi realizado, com a leitura dos trabalhos de (STACEY e MACGREGOR, 2000; MOYSÉS, 1997) e também serviu para complementar o estudo do cenário em que se propôs este trabalho, em conjunto com a pesquisa sobre Avaliação Qualitativa e Avaliação de Softwares Educacionais (ATAYDE, 2003; CHILDREN TECHNOLOGY REVIEW, 1998; DEMO, 2005; GLADCHEFF, 2001; GODOI e PADOVANI, 2009).

A arquitetura multiagente do sistema já existente (MELLO, RUBI, *et al.*, 2009) foi remodelada, com seus respectivos agentes, papéis, mensagens e demais características (DAMASCENO, CRUZ e JAQUES, 2010) sob a metodologia Prometheus (WINIKOFF, 2004). Isto alavancou a definição do Modelo de Aluno do sistema bem como as características do Agente Tutor com base em tal modelo. Para o embasamento da definição deste Módulo Tutor foram estudadas teorias de ensino sob aspecto pedagógico, aplicado à computação, tais como Ausubel e sua Aprendizagem Significativa (AUSUBEL, 1982), Taxonomia de Bloom (BLOOM, 1956; BLOOM, 1984), Teorias de Design Instrucional de Merril (MERRIL, 2001; MERRIL, 2002; MERRIL, 2007) e *Macro-Adaption* de Corbett, Anderson & Shute (CORBETT e ANDERSON, 1995; SHUTE, 1993).

Com esta base pedagógica e funcionalidades definidas, pôde-se estudar a estrutura que conteria a definição do Plano de Ensino para os alunos, bem como sua customização para

remediar eventuais erros cometidos pelos alunos – uma ontologia. Trabalhos como os de (GRUBER, 1995; GUARINO, 1995; JACOB, 2003) foram estudados, além do cenário em que estas ontologias se encontram, a Web Semântica, definida e avaliada por Tim Berners Lee em (LEE, HENDLER e LASSILA, 2001) e suas possibilidades de aplicação no ensino em ambientes virtuais, como as descritas em (EYHARABIDE, 2009).

Com este estudo realizado foi possível propor os Módulos Tutor e Modelo de Aluno no Sistema Tutor Inteligente PAT2Math. Para a efetivação do que foi proposto um protótipo foi construído, com a utilização da linguagem de programação Java, na plataforma de desenvolvimento JDeveloper, em conjunto com um banco de dados em MySQL. Tal protótipo concretiza o que foi descrito previamente além de ter permitido que fosse realizada a avaliação qualitativa com um grupo de docentes. Os resultados obtidos apontam para uma boa aprovação do trabalho e suas características, tornando a sua aplicação em sala de aula um passo interessante na maturação do projeto e posteriores avaliações.

Entre os trabalhos futuros envolvendo este projeto podem-se destacar as avaliações com alunos e um maior número de docentes para testes em sala de aula. Este feedback, do público-alvo do projeto como um todo, será extremamente importante para o amadurecimento da pesquisa que o envolve. Embora a avaliação com professores trouxe dados interessantes e aprovação da qualidade do que foi apresentado, os alunos pensam de uma maneira diferente e poderão contribuir e muito no desenvolvimento dos atuais e futuros módulos dentro do PAT2Math.

A pesquisa em demais estratégias pedagógicas para seleção de conteúdo para o aluno poderá levar a questão de individualização do aprendizado, focado nas características de cada aluno, a um novo nível – concretizando o PAT2Math como um sistema mais flexível e adaptável no seu ensino como um todo. Em cada estágio do aprendizado para cada tipo de aluno poderão ser estudadas alternativas para seleção e apresentação de conteúdo, acompanhamento de exercícios e remediação de aprendizados incorretos. Tendo em vista o caráter independente de domínio da solução aqui proposta, deve-se considerar a aplicação de tal método em outras áreas do conhecimento.

A definição da função que aponta o valor de *mastery* das habilidades de um aluno, na resolução de problemas algébricos, é outra questão que quando estudada envolve os processos internos de aprendizado, modelagens da maneira com que um conteúdo se fixa na mente do aluno. Para tanto, seu estudo necessita de um forte estudo e embasamento pedagógico e psicológico, unindo as questões de clareza no ensino, personalidade e empenho do aluno. Tais áreas são subjetivas, a modelagem de como o processo se desenvolve para afirmar que um

aluno possui determinado valor do conteúdo aprendido efetivamente parece ser bastante complexa.

BIBLIOGRAFIA

ANDERSON, J. et al. General principles for an intelligent tutoring architecture. In: _____ **Cognitive Approaches to Automated Instruction**. Mawah, NJ: Erlbaum, 1992.

ANDERSON, J. R. et al. Cognitive Modeling and Intelligent Tutoring. In: CLANCEY, J. W.; SOLOWAY, E. **Artificial Intelligence and Learning Enviroments**. [S.l.]: MIT/Elsevier, 1990.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. Parte 1: Modelo de qualidade. In: _____ **Engenharia de software - Qualidade de produto**. [S.l.]: [s.n.], 2003.

ATAYDE, A. P. **Metodologia de avaliação de qualidade de Software Educacional Infantil**. UFMG. [S.l.]. 2003.

AUSUBEL, D. P. **A aprendizagem Significativa:** a teoria de Ausubel. [S.l.]: [s.n.], 1982.

BALACHEFF, N. A modelling challenge: untangling learners' knowing. **Journées Internationales d'Orsay sur les Sciences Cognitives: L'apprentissage JIOSC2000**, 2000.

BEAL, C. L.; QU, L.; HYOKYEONG, L. Classifying learner engagement through integration of multiple data sources. **PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE**, 2006.

BECK, J.; STERN, M.; HAUGSJAA, E. Applications of AI in Education. **The ACM Student Magazine**, 1996.

BLOOM, B. S. **Taxonomy of Educational Objectives:** The classification of educational goals: Handbook I, cognitive domain. New York: Longman, 1956.

BLOOM, B. S. The 2 Sigma Problem: The search for methods of group instruction as effective as one-to-one tutoring". In: _____ **Educational Researcher**. [S.l.]: [s.n.], 1984. p. 4-16.

BUCKLEITNER, W. The State Of Children's Software Evaluation- Yesterday, Today And In The 21st Century. **Information technology in childhood education**, 1999.

BULL, S.; BRNA, P. Enhancing Peer Interaction in the Solar System. **C-LEMMAS: Roles of Communicative Interaction in Learning to Model in Mathematics and Science**, 1999. 202-208.

CHIAPPINI, G.; LEMUT, E. Construction and interpretation of algebraic models. **Proceedings of the Fifeenth International Conference for Psychology of Mathematics Education**, 1991.

CHILDREN TECHNOLOGY REVIEW. **Young Children & Computers, A Parent's Survival Guide**. [S.l.]: [s.n.], 1998.

CLANCEY, W. Intelligent Tutoring Systems - A tutorial Survey, 1986.

COHEN, P. R.; BEAL, C. R.; ADAMS, N. M. The Design, Deployment and Evaluation of the AnimalWatch Intelligent Tutoring System. In: _____ **Proceeding of the 2008 conference on ECAI 2008:** 18th European Conference on Artificial Intelligence. [S.l.]: IOS Press, 2008.

CORBETT, A. T.; KOEDINGER, K. R.; ANDERSON, J. R. Intelligent Tutoring Systems. In: SCIENCE, E. **Handbook of Human-Computer Interaction, 2nd edition**. [S.l.]: [s.n.], 1997. p. 849-870.

CORBETT, A.; ANDERSON, J. Knowledge Tracing: Modeling the acquisition of Procedural Knowledge. **User Modeling and User Adapted Interaction**, 1995.

COSTA, E. B.; LOPES, M. A.; FERNEDA, E. Mathema: A Multi-Agent Learning Enviroment. **Simpósio Brasileiro de Inteligência Artificial**, 1995.

COSTA, E. B.; PERKUSHI, A.; FIGUEIREDO, J. A Multi-Agent Based Enviroment to aid in the Design of Petri Nets Based Software Systems. **International Conference on Software Engineering and Knowledge Engineering**, 1996.

CURY, H. N.; KONZEN, B. Análise de resoluções de questões em matemática: as etapas do processo. **Revista da Sociedade Brasileira de Educação Matemática**, v. 7, n. 7, p. 33-41, 2006.

DAMASCENO, F. R.; CRUZ, A. D.; JAQUES, P. A. **Sistema Tutor Inteligente PAT2Math:** Proposta de Arquitetura Multi-Agente. Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações. Rio Grande: [s.n.]. 2010.

DAMASCENO, F. R.; JAQUES, P. Sistema Tutor Inteligente PAT2Math: Caráter Pedagógico. **Simpósio Brasileiro de Informática na Educação**, João Pessoa, Novembro 2010.

DEMO, P. **Avaliação Qualitativa - Polêmicas do Nosso Tempo**. [S.l.]: Autores Associados, 2005.

DEVEDZIC, V. **Semantic Web and Education**. [S.l.]: Springer, 2006.

DIX, A. **Human-Computer Interaction**. Londres: Prentice Hall, 1998.

EYHARABIDE, V. Ambientes personalizados de e-learning: considerando os contextos dos alunos. **Informática na educação: Teoria e Prática.**, 2009.

FIORENTINI, D.; LORENZATO, S. Investigação em educação matemática: percursos teóricos e metodológicos, Campinas, n. 3.

FLORES, C. et al. Projeto AMPLIA – uso da informática na educação médica. **Workshop de Informática Médica - WIM**, Fortaleza, 2003.

GAUTHIER, C. **Por uma teoria da pedagogia:** pesquisas contemporâneas sobre o saber docente. Ijuí: Onijuí, 1998.

GIRAFFA, L.; VICARI, R. M.; SELF, J. Multi-agent Based Pedagogical Games. In: HEIDELBERG, S. B. /. **Lecture Notes in Computer Science**. [S.l.]: [s.n.], 1998.

GLADCHEFF, A. P. **Um Instrumento de Avaliação da Qualidade para Software Educacional de Matemática**. Universidade de São Paulo. [S.l.]. 2001.

GODOI, K. A.; PADOVANI, S. Avaliação de material didático digital centrada no usuário: uma investigação de instrumentos passíveis de utilização por professores. In: _____ **Produção**. [S.l.]: [s.n.], 2009.

GRUBER, T. R. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. **International Journal of Human Computer Studies**, 1995.

GUANGZUO, C. OntoEdu:Ontology-based Eucation Grid System for e-Learning. **FIFTH AGRICULTURAL ONTOLOGY SERVICE (AOS) WORKSHOP**, 2004.

GUARINO, N. Formal Ontology, Conceptual Analysis and Knowledge Representation. **International Journal of Human Computer Studies**, 1995.

HOLT, P. et al. The state of student modelling. In: GREER, J. E.; MACALLA, G. I. **Student Modelling:** The Key to Individualized Knowledge-Based Instruction. [S.l.]: Springer, 1994.

ISOTANI, S.; BITTENCOURT, I. Estado da Arte em Web Semântica e Web 2.0: Potencialidades e Tendências da Nova Geração de Ambientes de Ensino na Internet. **Revista Brasileira de Informática na Educação**, v. 17, 2009.

JACOB, E. Ontologies and the Semantic Web. **Bulletin of the American Society for Information Science and Technology**, p. 3, 2003.

JAQUES, P. A. **Avaliando um Modelo Afetivo de Aluno baseado em uma Abordagem Cognitiva**. Simpósio Brasieliro de Informática na Educação. Fortaleza: [s.n.]. 2008.

JAQUES, P. A.; LEHMANN, M.; SYLVIE, P. **Evaluating the Affective Tactics of an Emotional Pedagogical Agent**. Proceedings of the 24th annual acm symposium on applied computing 2009. [S.l.]: [s.n.]. 2009.

JENSEN, F. V.; OLSEN, K. G.; ANDERSEN, S. K. An Algebra of Bayesian Belief Universes for Knowledge-Base Systems. In: _____ **Networks**. [S.l.]: John Wiley & Sons, v. 20, 1990.

JOVANOVIC, J. et al. Leveraging the Social Semantic Web in Intelligent Tutoring Systems. **Proceedings of the International Conference on Intelligent Tutoring Systems**, 2008. 563-572.

KOEDINGER, K. R. et al. Intelligent tutoring goes to school in the big city. **International Journal of Artificial Intelligence in Education**, 1997.

KOEDINGER, K. R. et al. Opening the door to non-programmers: authoring intelligent tutor behavior by demonstration. **Proceedings of Seventh International Conference on Intelligent Tutoring Systems**, Berlin, 2004.

KOEDINGER, K. R.; ALIBALI, M. W.; NATHAN, M. J. Trade-Offs Between Grounded and Abstract Representations: Evidence From Algebra Problem Solving. **Cognitive Science**, 2008.

KOENDINGER, K.; NATHAN, M. The real story behind story problems: Effects of representation on quantitative reasoning. **Journal of the Learning Sciences**, 2004.

LEE, T.-B.; HENDLER, J.; LASSILA, O. The Semantic Web. **Scientific American**, 2001.

LÉVY, P. **A inteligência coletiva:** por uma antropologia do ciberespaço. São Paulo: Loyola, 2000.

LIBBRECHET, P.; ULLRICH, C. Knowledge Representation and Management in ActiveMath. **Annals of Mathematics and Artificial Intelligence**, 2003. 47-64.

MELLIS, E. et al. ActiveMath: A Generic and Adaptive Web-Based Learning Environment. **International Journal of Artificial Intelligence in Education**, 2001. 385-407.

MELLO, G. et al. **Implementando o Agente de Base de Domínio do Sistema Tutor Inteligente PAT2Math**. XIII Ciclo de Palestras sobre Novas Tecnologias na Educação. Porto Alegre: [s.n.]. 2009.

MERRIL, M. D. First Principles of Instruction. **Education Technolgy Research & Development**, 2001.

MERRIL, M. D. First Principles of Instruction. In: MERRIL, M. D. **Educational Technology Research and Development**. [S.l.]: [s.n.], 2002. p. 43-59.

MERRIL, M. D. First Principles of Instruction: A Synthesis. In: MERRIL, M. D. **Trends and Issues in Instructional Design and Technology 2nd Edition**. [S.l.]: Merril/Prentice Hall, 2007. p. 62-71.

MINISTÉRIO DA EDUCAÇÃO, SECRETARIA DE ENSINO FUNDAMENTAL. **Parâmetros curriculares nacionais:** Matemática. 2ª. ed. Brasília: [s.n.], 2000.

MINISTÉRIO DA EDUCAÇÃO. SECRETARIA DE ENSINO FUNDAMENTAL. **Parâmetros Curriculares Nacionais:** Matemática. 3ª. ed. Brasília: [s.n.], 1998.

MINISTÉRIO DA EDUCAÇÃO/SECRETARIA DO ENSINO FUNDAMENTAL. **Parâmetros Curriculares Nacionais:** Matemática. Brasília: [s.n.], 1998.

MIZOGUCHI, R.; HAYASHI, Y.; BOURDEAU, J. Inside Theory-Aware Authoring System. **Proceedings of the Int. Workshop on Ontologies and Semantic Web for E-Learning (SWEL)**, 2007.

MOREIRA, M. A. **Teorias de Aprendizagem**. São Paulo: EPU - Editora Pedagógica e Universitária LTDA., 1999.

MOYSÉS, L. **Aplicações de Vygotsky à Educação Matemática**. Campinas: Papirus, 1997.

MURRAY, T. Authoring intelligent tutoring systems: An analysis of the state of the art. **International journal of artificial intelligence in education, vol. 10**, 1999.

NICAUD, J. F. et al. Experiments with Aplusix in Four Countries. **International Journal for Technology in Mathematics Education**, Fevereiro 2006.

NICAUD, J. F.; BOUHINEAU, D.; CHAACHOUA, H. Mixing Microworld and CAS Features in Building Computer Systems that Help Students Learn Algebra. **International Journal of Computer for Mathematical Learning**, 2004.

OLIVEIRA, E. et al. **Proposal of utilization of an animated pedagogical agent to interact affectively with students who have compromised attentional cognitive processes**. Young Researchers Track of the 9th International Conference on Intelligent Tutoring Systems. Montreal: [s.n.]. 2008.

POLSON, M.; RICHARDSON, J. **Foundations of Intelligent Tutoring Systems**. New Jersey: Lawrence Erlbaum Associates Publishers, 1988.

PROGRAMA NACIONAL DE INFORMÁTICA. **PROINFO**. [S.l.]. 1998.

REZENDE, S. O. **Sistemas Inteligentes:** Fundamentos e Aplicações. Barueri: Manole. 2005.

RICHARDSON, W. **Blogs, Wikis Podcasts and Other Powerful Tools for Classrooms**. [S.l.]: Corwin Press, 2006.

RUBI, G. L. **Estratégias e Táticas Pedagógicas para o ensino de Equações Algébricas no Sistema Tutor Inteligente PAT2Math.** São Leopoldo: Trabalho de Conclusão de Curso (Graduação em Licenciatura Plena em Matemática), 2010.

RUSSEL, S.; NORVIG, P. **Artificial Intelligence:** A Modern Approach. [S.l.]: [s.n.]. 1996.

SEFFRIN, H. et al. **Um resolvedor de equações algébricas como ferramenta de apoio à sala de aula no ensino de equações algébricas**. Workshop de Informática na Escola. Bento Gonçalves: [s.n.]. 2009.

SEFFRIN, H. et al. Resolvendo equações algébricas no STI PAT2Math. **Simpósio Brasileiro de Informática na Educação**, João Pessoa, 2010.

SHUTE, V. J. A macroadaptative approach to tutoring. **Journal of Artificial Intelligence in Education**, 1993.

SIEKMANN, E.; MELIS, J. ActiveMath: An Intelligent Tutoring System for Mathematics. **Artificial Intelligence and Soft Computing - ICAISC**, 2004.

STACEY, K.; MACGREGOR, M. Learning the Algebraic Method of Solving Problems. **Journal of Mathematical Behavior**, 2000.

SUSAN, B.; SMITH, M. A Pair of Student Models to Encourage Collaboration. [S.l.]: Springer, 1997.

VANLEHN, K. The Behavior of Tutoring Systems. **International Journal of Artificial Intelligence in Education**, 2006.

VICENT, A. et al. The Cognitive Tutor Authoring Tools (CTAT): Preliminary Evaluation of Efficiency Gains. **Proceedings of the 8th International Conference on Intelligent Tutoring Systems**, Berlin, 2006.

VIGOTSKY, S. L.; LURIA, R. A.; LEONTIEV, N. A. **Linguagem, Desenvolvimento e Aprendizagem**. [S.l.]: [s.n.], 1934-1986.

WEBBER, C.; PESTY, S. Ambiente de Aprendizagem Baghera: Uma comunidade de agentes artificiais e humanos. **Simpósio Brasileiro de Informática na Educação**, 2004.

WEISS, G. **Multiagent Systems:** A Modern Appoach to Distributed Artificial Intelligence. [S.l.]: Massachusetts Institute of Technology. 1999.

WINIKOFF, P. L. The Prometheus Metodology, 2004.

WOOLDRIDGE, M. **An Introduction to MultiAgent Systems**. [S.l.]: Wiley. 2002.

ZHOU, L. Ontology Learning: state of the art and open issues. In: SPRINGER **Information Technology and Management**. [S.l.]: [s.n.], 2007.

APÊNDICE I – QUESTIONÁRIO APLICADO NA AVALIAÇÃO

Conceitos Matemáticos

TUT.5 – Os conceitos matemáticos definidos pelo tutorial estão corretos?

( ) SIM ( ) NÃO

TUT.6 – Os conceitos matemáticos definidos pelo tutorial são precisos, ou seja, os conceitos são definidos de forma clara, sem utilização de palavras ambíguas?

( ) SIM ( ) NÃO

TUT.9 – As notações são adequadas e coerentes com as utilizadas pelo professor em sala de aula? ( ) SIM ( ) NÃO

Conteúdo

TUT.10 – O tutorial é abrangente no sentido de abordar o máximo possível dentro do assunto a ser trabalhado, e coincide com a organização do currículo adotado na escola?

( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

TUT.12 – O tutorial caminha do básico ao profundo de forma suave? ( ) SIM ( ) NÃO

TUT.13 – O tutorial permite a revisão do conteúdo que já foi trabalhado?

( ) SIM ( ) NÃO

TUT.17 – O tutorial permite que o conteúdo seja particularizado para cada aluno, ou seja, permite que o conteúdo a ser abordado seja limitado de acordo com o que se deseja trabalhar com cada aluno individualmente? ( ) SIM ( ) NÃO ( ) NN

Exercícios

TUT.22 – Há um equilíbrio entre o conteúdo exposto e os exercícios propostos?

( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

TUT.23 – O tutorial pode gerar problemas dinamicamente levando em conta as necessidades do aprendiz? (O software gera um modelo do aprendiz e a partir deste modelo pode criar problemas dinamicamente) ( ) SIM ( ) NÃO

Usabilidade

TUT.27 – O tutorial permite que "sessões" interrompidas sejam reiniciadas a partir do "ponto de parada"? ( ) SIM ( ) NÃO

Feedback

TUT.28 – Qual é a forma de *feedback* emitida pelo tutorial quando o aluno erra a resposta do exercício proposto?

( ) repetição – O tutorial simplesmente reapresenta a pergunta anteriormente feita ao aluno.

( ) pista – O tutorial fornece uma mensagem na intenção de chamar a atenção do aluno sobre o fundamento do erro cometido, com o objetivo de fazer com que o aluno descubra o que "implicitamente" já sabe.

( ) explicação através de mensagem padrão – O tutorial fornece uma única mensagem de explicação para todo e qualquer erro, no sentido de simplesmente "não ser a resposta correta".

( ) explicação em função da resposta do aluno – A resposta do aluno é analisada na sua originalidade e uma explicação é colocada de acordo com esta resposta.

TUT.29 – O *feedback* realizado pelo tutorial permite que o aluno reflita sobre seu erro e tente corrigi-lo, sem intervenção ostensiva do professor?

( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

TUT.30 – Você considera a forma de *feedback* emitida pelo tutorial adequada?

( ) SIM ( ) NÃO

TUT.31 – As respostas do aluno são verificadas corretamente?

( ) SIM ( ) NÃO

TUT.32 – O tutorial justifica suas ações ou raciocínios? (Quando, por exemplo, mostrar ao aluno a forma de resolução de algum exercício) ( ) SIM ( ) NÃO

TUT.32.1 – Se SIM, as justificativas ou raciocínios estão corretamente empregados?

( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

TUT.33 – O tutorial consegue acessar e analisar as razões das respostas do aprendiz?

( ) SIM ( ) QUASE INTEIRAMENTE ( ) POUCO ( ) NÃO

Desempenho dos Alunos

TUT.34 – O tutorial oferece um relatório sobre o desempenho do aluno para que seja possível verificar se os objetivos da lição foram alcançados? (número de respostas certas, número de respostas erradas para cada sessão, etc.)

( ) SIM ( ) SIM, MAS COM POUCAS INFORMAÇÕES ( ) NÃO

TUT.35 – O tutorial mantém um histórico de utilização por parte do aluno? (número de sessões que o aluno realizou, tempo gasto em cada módulo, etc.)

( ) SIM ( ) SIM, MAS COM POUCAS INFORMAÇÕES ( ) NÃO