Utilizando a tecnologia de agentes na construção de Sistemas Tutores inteligentes em ambiente interativo

Rodrigo R. V. Goulart[1[1]] Lucia M. M. Giraffa[2[2]]
Faculdade de Informática
Programa de Pós-Graduação em Ciência da Computação – PUCRS
Av. Ipiranga 6681 – prédio 16 – Porto Alegre – RS – 90610-900
Fone/Fax: +55 51 320-3611
{rgoulart,giraffa}@inf.pucrs.br

RESUMO

Este artigo apresenta os resultados obtidos com a pesquisa envolvendo Jogos Educacionais interativos modelados com arquitetura de Sistemas Tutores Inteligentes (STI). Utilizamos a abordagem de agentes para tratar as funcionalidades associadas aos módulos de uma arquitetura tradicional de STI. O domínio é modelado com agentes reativos utilizando técnicas de Machine Learning (aprendizagem por reforço). Os alunos e o tutor são modelados através de agentes cognitivos em arquitetura BDI (belief, desire, intention). Este conjunto complexo de escolhas interdisciplinares, tanto em nível de projeto de *software* educacional, como de técnicas de Inteligência Artificial, nos levaram a várias investigações e resultados que trouxeram questionamentos e algumas contribuições para avançar nossa pesquisa na área de ambientes de ensino aprendizagem computadorizada.

Palavras Chave: Sistemas Tutores Inteligentes, Agentes, Jogos Educacionais.

1. Introdução

A tendência para construção de softwares educacionais, a exemplo dos Sistemas Tutores Inteligentes (STI), é propiciar interação entre os alunos, tanto em nível presencial como através da rede. Por presencial entende-se a interação que ocorre quando dois alunos utilizam o mesmo computador, e o resultado da sua interação pode ser percebido pelo sistema através das ações que realizam. Já a interação na rede é realizada entre alunos que estão em computadores diferentes e no mesmo laboratório, ou até mesmo dispersos na Internet. Dentro deste contexto, Silveira e Vicari [16] e Giraffa e Viccari [2], consideram que a utilização de sociedades baseadas em agentes está se consolidando como uma alternativa apropriada para o projeto de STI.

Os STI surgiram no início da década de 70 como uma promessa de transpor os limites dos programas tradicionais (CAI - Computer Assisted Instruction) e geraram muitas expectativas a seu respeito. As limitações de hardware, software e nosso, ainda, incompleto conhecimento de como efetivamente humanos processam informação, geram muitas lacunas a serem preenchidas no que concerne a implementação de STI que tenham qualidade técnica e

[1[1]] Mestrando, Bolsista CAPES.
[2[2]] Orientadora, Bolsista CNPq.

pedagógica ao mesmo tempo. Quando consideramos a questão do paradigma construtivista, este *gap* (lacuna) aumenta consideravelmente. Novamente, a tecnologia de agentes surge como uma alternativa para superar estes *gaps* e avançar na pesquisa e desenvolvimento de STI que efetivamente sejam mais adequados para auxiliar o aluno nas tarefas que devem desempenhar em um determinado ambiente. Em tempos de WEB e sistemas para WEB poderíamos questionar se existe ainda espaço para os STI. Acreditamos que nunca foi tão necessária pesquisa nesta área, pois a navegação solitária na rede não garante resultados e por si só, o fato de um sistema poder ser acessado via WEB, não fará com que o aluno tenha ganhos garantidos. A possibilidade de ter um tutor funcionando como assistente, coletando informações dos alunos e monitorando a sessão de trabalho, auxiliando o professor nas tarefas adequadas, nos permitirá construir ambientes educacionais na WEB mais interessantes e robustos sob o ponto de vista pedagógico.

A utilização da tecnologia de agentes no projeto de STI, propicia a exploração do domínio (conteúdo) de forma mais dinâmica, permitindo representação de domínios complexos com mais baixo custo computacional e utilizando interfaces mais representativas no que concerne aos movimentos dos elementos e suas inter-relações, como no caso de simulações de fenômenos físicos, químicos e biológicos. Quanto à modelagem do aluno, a tecnologia relacionada com os agentes cognitivos (deliberativos), permite a construção de modelos mais robustos e fora do padrão usual de estereótipos, *buggy* e *overlay* [18]. A seleção de múltiplas estratégias de ensino e utilização de diferentes abordagens para o tratamento da questão envolvendo a assistência personalizada ao aluno, é também estendida com o uso de agentes, como demonstram os trabalhos de Aimeur [1] e Giraffa [5].

Os agentes desenvolvidos para ambientes de ensino-aprendizagem recebem o nome de agentes pedagógicos. Estes agentes podem atuar como tutores virtuais, alunos virtuais ou colegas virtuais que auxiliam no processo de aprendizagem. Estas diferentes formas de atuação podem compor grupos de agentes que distribuem entres si as tarefas. Estas tarefas podem ser de maior abrangência (modelagem do aluno ou seleção da estratégia e táticas) ou de menor abrangência (cada agente é responsável por uma estratégia). Os objetivos de um agente pedagógico podem ser descritos em função do seu comportamento (estratégia). Os comportamentos possíveis para um tutor, segundo Giraffa e Viccari[6], são:

- **Guia**: o agente é diretivo em suas intervenções e monitora o aluno todo tempo, conduzindo o aluno na resolução do problema por todo processo de interação;
- **Assistente**: o agente é menos diretivo e monitora o aluno todo o tempo, intervindo baseado em heurísticas sobre a resolução do problema naquele domínio;
- **Facilitador**: o agente monitora o aluno todo o tempo porém não é diretivo, ele apenas dá dicas sobre a resolução do problema e só intervém quando solicitado.

O tutor pode utilizar um ou vários comportamentos, sendo assim os objetivos podem variar em função das informações recebidas do usuário (ações do aluno) e da situação que se deseja atingir (plano). Alguns exemplos de STI que representam o estado da arte em modelagem de tutores podem ser encontrados nos trabalhos de: LeCS [13], ITStrategic [9], Explanation Agent [22], MarCo [17], Troublemaker de [1] e [3], SmartEgg [10], Framework JADE [16] e AME-A [12].

Este artigo apresenta os resultados correntes da pesquisa iniciada pelo grupo de JOGOS & IA (Inteligência Artificial) do PPGCC-PUCRS, sobre STI modelados em ambientes de jogo.

Acreditamos que o jogo possui um conjunto de característica que permitem a exploração de múltiplos aspectos associados a resolução de problemas, por parte dos alunos. Por possuir características lúdicas, alta complexidade de modelagem e implementação. Os jogos, quando projetados com arquitetura de STI, possibilitam resultados muito interessantes, tanto a nível de Ciência da Computação, bem como de Ciência da Educação.

Este artigo esta dividido em 5 seções. A seção 2 apresenta a descrição do conjunto de protótipos associadas às pesquisas do grupo envolvendo os STI associados a uma interface e regras de um jogo educacional, e as conclusões a respeito das limitações encontradas. Na seção 3 apresentamos uma proposta de solução para estas limitações por meio da análise do projeto de STI com jogos e a inserção de um agente mediador para auxiliar o processo de tutoria. Na seção 4 apresentamos algumas conclusões de nossa pesquisa e trabalhos futuros. Na seção 5 apresentamos a bibliografia utilizada neste artigo.

2. STI em abordagem de Jogos Educacionais

Nossa pesquisa utiliza os resultados obtidos por Giraffa e Viccari [5], e Callegari e Oliveira [2]. O projeto MCOE [5] é o primeiro resultado das pesquisas do grupo que se utiliza da metáfora de jogos, onde a interação do usuário (aluno) com o sistema é acompanhada por um tutor virtual. O comportamento do sistema pode ser descrito através da interação de três elementos: Jogo, Aluno e o Tutor.

O jogo representa o domínio (conteúdo) a ser apresentado ao aluno, através da simulação da cadeia alimentar de um lago e os seus elementos (peixes, plantas, etc). Através da representação gráfica na interface o aluno observa os acontecimentos do jogo e interfere através dos mecanismos de interação disponíveis (botões). Durante a partida o tutor observa o jogo e as ações do aluno, interagindo conforme a necessidade na tentativa de auxiliá-lo na tarefa de manter o ecossistema estável. Assim como nos jogos convencionais, no MCOE estão definidas as regras, formas de interação e os elementos que compõem o jogo. Contudo, determinar todas as situações que podem ocorrer durante a partida a fim de prestar o melhor auxílio pode ser uma tarefa demasiadamente complexa. Os jogos atualmente possuem uma grande variedade de formas de interação e ambientes de simulação. Na tentativa de torná-los mais realistas, técnicas de IA estão sendo empregadas para tornar os ambientes mais ricos em termos de situações [4].

O MCOE utiliza técnicas de IA tanto na simulação do Ecossistema como no auxílio ao aluno. Sua arquitetura é composta por um sistema multiagente híbrido que permite a exploração dinâmica e interativa do ambiente [5]. Ela é composta por agentes reativos, que compõem uma sociedade chamada SMAR (Sistemas Multiagentes Reativos), e agentes cognitivos, ou SMAC (Sistemas Multiagentes cognitivos).

Cada agente da SMAR implementa o comportamento reativo de um indivíduo do ecossistema do lago. Desta forma, a cadeia alimentar desenvolve um comportamento autônomo semelhante ao de uma cadeia alimentar real, onde os peixes e demais seres podem morrer, nascer, se alimentar quando estão com fome, etc. Já os agentes cognitivos SMAC (que representam o comportamento dos alunos e do tutor) são modelados utilizando uma arquitetura BDI (*belief*, *desire*, *intention*). Segundo Mora [11], as idéias básicas desta abordagem baseiam-se na descrição do processamento interno de um agente utilizando um conjunto básico de estados mentais (crença, desejo e

intenções) e na definição de uma arquitetura de controle através da qual o agente seleciona racionalmente o curso de suas ações. Uma crença de um agente corresponde às informações que o agente tem sobre o mundo (ambiente), um desejo de agente (ou objetivo em um sistema) intuitivamente corresponde a tarefas estabelecidas pelo próprio agente e, intenção é um desejo com o valor agregado do comprometimento em alcançá-lo (e torná-lo uma crença). Com abordagem BDI o modelo do aluno foge ao padrão usual de estereótipos, *overlay* e *buggy* para um modelo composto por estados mentais, crenças e desejos, que são enviados para um *kernel* cognitivo (parte deliberativa do agente tutor).

No trabalho de Mora [11], foi desenvolvida a ferramenta X-BDI que permite a implementação do *kernel* do tutor. Esta ferramenta faz parte de uma arquitetura distribuída em camadas, e permite a comunicação do *kernel* através de uma rede de computadores.

O MCOE utiliza a capacidade de interação através da rede do X-BDI, que permite separar em termos de implementação o ambiente de jogo (seus agentes, aluno e sistema) do *kernel* cognitivo do agente tutor. Conseqüentemente, a manutenção e ampliação do ambiente ou do tutor podem ser realizadas de forma independente. Mas mais do que isto, o tutor tem a possibilidade de interagir com jogos (interfaces de ambientes) em diferentes plataformas ou computadores.

A proposta apresentada por Callegari e Oliveira [2] trouxe novas contribuições para o MCOE, baseada no fato de que um ambiente mais dinâmico (que ofereça uma maior variedade de situações para os alunos) possa ser utilizado de forma mais ampla do ponto de vista pedagógico. Para isto, Callegari e Oliveira propuseram alterações na arquitetura dos agentes do sistema MCOE para acomodar um método de Aprendizagem por Reforço (*Negotiated W-Learning*), que apresentou resultados bastante motivadores. Esta técnica foi aplicada aos componentes da cadeia alimentar (para fins de prototipação inicialmente foram testados com os peixes, que podem ser divididos em três subcategorias: grandes, médios e pequenos), tornando o ambiente de jogo mais competitivo. Neste contexto são identificados três processos importantes na execução do sistema:

- **Looping principal**: A cada ciclo o sistema gera um conjunto de informações sobre a partida que devem ser avaliadas e/ou enviadas para outros processos. Estas informações são geradas pelo ambiente do jogo (SMAR), pelo *kernel* do tutor e pela interface. Na proposta original do sistema [5] é proposta a interação ente dois alunos através do sistema, no entanto neste trabalho consideraremos apenas a interação de um único usuário com o sistema.
- **Ambiente do jogo**: o funcionamento do ecossistema permite que seus elementos (peixes, plantas, etc) simulem (de forma reducionista) um comportamento autônomo no ambiente (agora estendido pelas contribuições de [2]). Estes elementos podem sofrer interferência interna ou externa ao ambiente do jogo, como por exemplo a inserção de um agente poluidor, a alteração de energia decorrente da intervenção do usuário, ou a perda natural de energia de cada elemento durante a partida. A execução deste processo de simulação no MCOE é um processo autônomo dentro do contexto global do sistema, dependente apenas do looping principal (o looping controla cada execução de um ciclo no ambiente do jogo).
- ***Kernel* Cognitivo**: Apesar do *kernel* poder ser executado em outro computador, ele é dependente das informações oriundas da simulação do ecossistema e do processo que controla o looping principal. Sua deliberação se da a partir das

informações enviadas via rede. Após a deliberação o tutor envia ou uma mensagem de resposta.

Observa-se, que assim como nos jogos convencionais, estes três processos podem ser executados de forma concorrente, o que tornaria mais dinâmica a sua execução. No entanto, a possibilidade de um processo, como o do Ambiente do Jogo, ser totalmente independente (do looping principal), não parece ser interessante pois os processos paralelos de atualização do conteúdo da interface, ciclo de execução do ambiente do jogo e envio de mensagens para o *kernel* poderiam prejudicar a simulação em virtude de uma possível falta de sincronia. No entanto é possibilidade a ser analisada em trabalho futuros.

A interface neste contexto também pode desempenhar tarefas independentes como um processo em paralelo, mas optamos por agregar sua análise ao looping principal pois estamos considerando apenas os aspectos básicos de entrada e saída.

Estes três processos desempenham um papel central no funcionamento do sistema. No entanto algumas limitações foram identificadas:

- O *kernel* cognitivo do tutor é dependente da entrada de dados oriunda do sistema (interface + jogo) o que limita sua atuação autônoma, pois ele somente atua quando recebe uma entrada (conforme identificado em [5]).
- O volume de informações monitoradas pelo tutor em uma partida, como monitorar o aluno e as mudanças do ambiente (jogo), pode se tornar demasiadamente grande, o que afastaria o tutor de seu real foco: elaborar o plano pedagógico que auxilie o aluno. Diversos trabalhos exemplificam as alternativas existentes na literatura para tratar este problema ([13], [9], [22], [17], [1], [3], [10], [16] e [12]).
- Este volume de informações pode gerar deliberações equivocadas por parte do *kernel* uma vez que a comunicação se da por meio de uma rede de computadores, e esta sujeita a problemas de atraso no envio de mensagens.
- A modelagem (com estados mentais) e implementação do tutor podem tornar sua manutenção ou ampliação muito complexa. A grande variedade de situações possíveis no ambiente, bem como a descrição dos comportamentos do agente tutor modelado com BDI, podem se tornar extensas e de difícil manipulação.
- Com o uso desta abordagem, existe a falta de uma definição clara dos papéis dos agentes do sistema, dentro do processo de monitoração destas informações.

A limitação do recebimento de mensagens do *kernel* pode ser analisada de duas formas.

A primeira é a possibilidade do *kernel* manter certas informações ao invés de aguardar por elas. Isto permitiria uma maior autonomia para sua tomada de decisão. No entanto, se considerarmos o tempo de partida, esta e outras informações dificilmente poderiam ser mantidos pelo tutor já que estas podem ser alteradas pelo usuário (por exemplo, quando é solicitada pausa na execução do jogo), o que tornaria as informações mantidas pelo *kernel* inconsistentes. A segunda forma de analise é manter o *kernel* dependente das entradas geradas pelo sistema, mas "bem informado", enviando a ele somente as informações importantes e no momento correto. Esta inclusive é o foco de nossa pesquisa.

Observa-se que um jogo é um sistema onde as mudanças no ambiente são muito rápidas e o processamento dos eventos do ambiente deve ser preciso e veloz. Mesmo em jogos que utilizam técnicas de IA para criar comportamentos mais complexos no ambiente, a relação do

controle dos elementos do jogo versus inteligência aplicada no comportamento dos mesmos ainda é um problema em aberto em virtude da sua modelagem, performance e manutenção [4].

Neste mesmo contexto o jogo educacional tem uma tarefa agregada mais complexa (tão importante quanto que gerar comportamentos mais elaborados em um cenário) que é a tutoria.

3. E-MCOE (Extended MCOE)

Para tratar estas questões apresentamos uma proposta para solução destas limitações descrita através das seguintes medidas:

- determinação o fluxo de informações entre cada agente;
- definição dos papéis de cada agente dentro arquitetura e suas responsabilidades no processo de comunicação na sociedade;
- adição um agente mediador que irá sintetizar as informações do ambiente e da interface com a finalidade de agregar tarefas específicas da comunicação entre os agentes, retirando esta responsabilidade dos mesmos e simplificando a troca de mensagens. Este agente utiliza as considerações de [8] e [19] para definição do agente mediador.

A arquitetura estendida é descrita através de quatro módulos (Figura **Erro! Use a guia Início para aplicar ao texto que deverá aparecer aqui.**-**Erro! Indicador não definido.**): Interface, Jogo, Tutor e Mediador. Cada módulo desempenha uma ou um conjunto de tarefas distintas com a finalidade de dividir e clarificar as responsabilidades que devem ser assumidas no sistema.

Cada módulo se comunicará com outro através de mensagens. Estas mensagens obedecem a um padrão determinado que define um protocolo de comunicação simples e funcional. Desta forma, caso ocorram a alterações ou ampliações das funcionalidades de um ou mais módulos estas não tenham maior impacto na forma com que são executadas as demais tarefas de cada módulo.

Cada mensagem tem o seguinte formato: **mensagem(remetente, cabeçalho, conteúdo)**. **Remetente** é o módulo que enviou a mensagem, **destinatário** é o módulo que irá receber a mensagem, **cabeçalho** é o código da informação utilizada pelo remetente e **conteúdo** são as informações adicionais, sendo que estas podem ser de formato variado (legíveis para o destinatário através da identificação do cabeçalho);

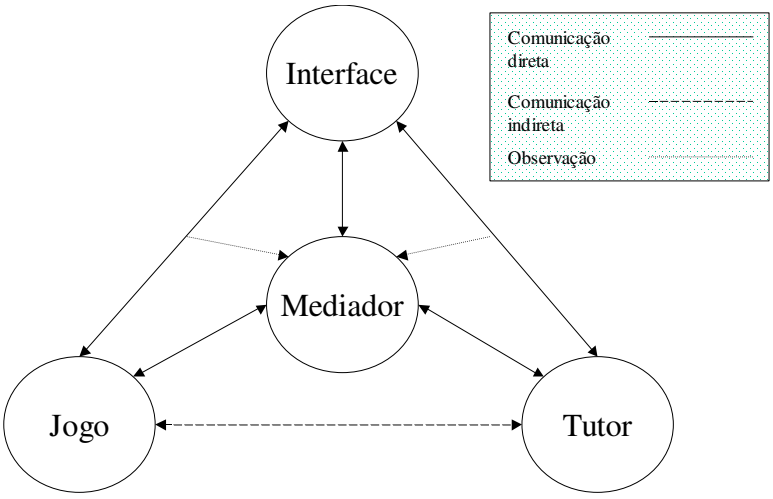


Figura **Erro! Use a guia Início para aplicar ao texto que deverá aparecer aqui.-Erro! Indicador não definido.**: Arquitetura do E-MCOE

Este formato é utilizado por cada módulo para transmissão de informações no sistema, entre os módulos. Cada módulo é responsável pela manutenção de suas tarefas internas e o envio de mensagens. Estas tarefas são definidas conforme a arquitetura descrita por Giraffa e Viccari [5] e ampliadas por Callegari e Oliveira[2]. Em síntese cada módulo executa as seguintes tarefas:

- **Módulo Interface**: Concentra os mecanismos de interação com o usuário e a viabilização gráfica do ambiente do jogo. Neste módulo é gerenciada a interface que disponibiliza para o usuário ferramentas para interagir com o jogo. Através de mensagens o módulo informa que ferramentas estão sendo utilizadas, como também recebe mensagens sobre mudanças no estado dos elementos do jogo (nascer, morrer, mover, etc) oriundas do módulo Jogo.
- **Módulo Tutor**: O Módulo tutor na realidade é uma interface do sistema com o *kernel* X-BDI. Através dele é possível enviar quaisquer mensagens do sistema para o *kernel* cognitivo, assim como receber mensagens do mesmo e retransmiti-las para os módulos correspondentes.
- **Módulo Jogo**: O Módulo jogo envia mensagens para interface sobre a atualização da posição, morte ou nascimento de novos elementos. Ele envia também mensagens para o Tutor informando o estado atual do ambiente e seus elementos, como por exemplo energia e quantidade. Este módulo recebe mensagens do módulo Interface sobre a inserção de agentes poluidores, ou da interferência do aluno através das ferramentas.

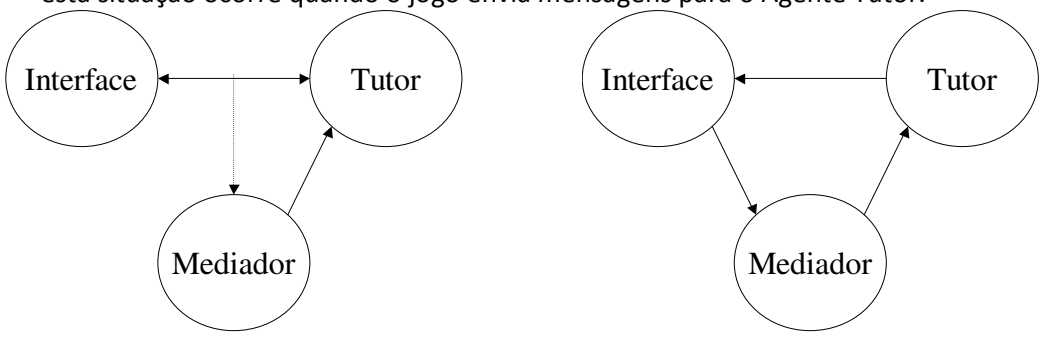
Cada um destes módulos executa determinadas tarefas e algumas delas são para o envio e recepção de informações. Analisando as mensagens podemos identificar o fluxo de informações do sistema, pontos de gargalo e determinar que informações são úteis para o tutor.

Estas considerações permitem que o Módulo Mediador (Agente Mediador) possa utilizar este mecanismo para analisar o sistema, suas mensagens e prestar um serviço para o Agente Tutor. O Agente Mediador desempenha então dois comportamentos no sistema:

- • **Observação**: o agente observa as mensagens que são enviadas de um módulo para o outro sem interferir na comunicação direta entre os mesmos; este tipo de leitura de mensagens ocorre, por exemplo, quando há comunicação entre a interface e o tutor

(Figura **Erro! Use a guia Início para aplicar ao texto que deverá aparecer aqui.**-**Erro! Indicador não definido.**a).

- **Mediação**: o agente recebe mensagens de um módulo destinadas a um terceiro e, após a sua análise, envia ou não mensagens para o destinatário (Figura **Erro! Use a guia Início para aplicar ao texto que deverá aparecer aqui.**-**Erro! Indicador não definido.**b); esta situação ocorre quando o jogo envia mensagens para o Agente Tutor.



(a) Observador (b) Mediador

Figura **Erro! Use a guia Início para aplicar ao texto que deverá aparecer aqui.**-**Erro! Indicador não definido.**: Comportamentos do Agente Mediador

Quando o agente observa a comunicação entre dois módulos, é realizada a leitura das mensagens com a finalidade de identificar se o fluxo de mensagens entre os módulos está normal. Por exemplo, quando o tutor recebe mensagens da interface (Figura **Erro! Use a guia Início para aplicar ao texto que deverá aparecer aqui.**-**Erro! Indicador não definido.** a) significa que o aluno esta interagindo com o sistema. Caso as mensagens parem de ser enviadas por um determinado espaço de tempo isto pode sinalizar que o aluno não deseja interagir ou quer observar o colapso do jogo.

Já o comportamento de mediação é aplicado onde o fluxo de comunicação é muito intenso, o que determina a necessidade de uma filtragem das informações. Um exemplo é a comunicação entre o Agente Tutor (através do Módulo Tutor) e o Jogo (Módulo Jogo). O estado atual de cada elemento do jogo deve ser monitorado pelo tutor, mas muitas vezes as mudanças no estado destes elementos não significam diretamente informação útil, pois somente tornam-se importantes em determinadas ocasiões. Por exemplo, se o nível médio de energia dos peixes de tamanho médios estiver abaixo de 5, em uma escala de 0 à 100, o Agente Tutor deve ser informado da situação crítica destes peixes, mas acima de 20 o envio de mensagens sobre o estado atual podem ser enviadas em espaços de tempo mais longos.

O Agente Mediador pode analisar as mensagens enviadas para o Tutor e de forma sintetiza-las e então enviar mensagens mais ricas do ponto de vista de importância para o Tutor.

As regras que descrevem a forma de análise das mensagens é descrita através de uma gramática que permite a adaptação do Agente Mediador a alterações e ampliações nos outros módulos. Por exemplo, se uma nova ferramenta for inserida na interface, ou mudanças nas propriedades dos peixes podem gerar novas mensagens para o Tutor. Contudo, o detalhamento desta gramática não será realizado neste trabalho.

A utilização de uma gramática permite a expansão do sistema sem a necessidade de alterações na forma com que o Agente Mediador realiza suas tarefas (a exemplo do *kernel* BDI do tutor, descrito através de uma gramática como também interfaces e jogos que utilizam scripts [20]).

A separação da forma de comunicação, e a maneira com que são realizadas aumenta o potencial exploratório do sistema (a nível pedagógico), uma vez que novas interfaces e jogos que utilizem o mesmo formato de mensagens podem ser incorporados ao sistema utilizando também linguagens de scripts [15]. No entanto esta capacidade ainda esta sendo investigada.

Para avaliação desta proposta foi elaborado um protótipo para comprovar nossas hipóteses. Esta tarefa foi divida em duas etapas: a primeira é o desenvolvimento de um novo protótipo do ambiente MCOE, em escala reduzida e com poucos recursos gráficos, que incluam as alterações propostas por [2]; a segunda etapa foi a inserção do novo agente na arquitetura do protótipo para avaliação das alterações na modelagem/manutenção/execução do sistema.

A fim de permitir a análise e ampliação de nossos resultados em outros trabalhos (futuros), analisamos a utilização de uma linguagem para desenvolvermos nosso protótipo, que permitisse também sua aplicação em diferentes plataformas. Desta forma optamos pela linguagem Java a qual atende as necessidades de portabilidade, expansão e documentação que desejamos obter. No entanto foi necessária a conversão do protótipo de [2] de C++ para Java. A linguagem Java trouxe também contribuições importantes para o estágio de prototipação. A possibilidade de implementar estruturas de dados e suas formas de manipulação, *Threads*, interfaces gráficas web e comunicação via *sockets* de forma simplificada auxiliou o processo de prototipação.

4. Considerações finais

A construção de STI utilizando arquitetura de sistemas multiagentes permite um ganho de qualidade tanto no aspecto de representação do domínio, aumentando as possibilidades pedagógicas do ambiente, quanto nas possibilidades de se construir modelos de aluno e tutor mais robustos [5].

Este trabalho propõe uma extensão para arquitetura multiagente de [5], cuja modalidade de interação e representação do domínio é um jogo educacional. Esta modalidade traz consigo muitas limitações para o processo de auxílio desempenhado pelo tutor. A fim de auxiliar na solucionar estas limitações realizamos um processo de análise detalhado da arquitetura do testbed MCOE e das características inerentes a um jogo desta natureza. Com base nestes dados propomos a extensão da arquitetura através da inclusão de um agente que auxilia o tutor em suas tarefas, observando continuamente o ambiente do jogo e a utilização da interface por parte dos alunos.

Para construção de um STI que utiliza jogos como domínio, há a necessidade de se observar aspectos como: quais são os processos robustos do sistema e seus papéis frente ao fluxo de informações que estão sendo geradas e compartilhadas. Além destes, aspectos mais técnicos devem ser consideradas tais como particularidades das tecnologias de redes e programação. Acreditamos que estas considerações podem influir também no desempenho do tutor, e portanto devem ser levadas em conta pois se refletem nos aspectos pedagógicos do sistema.

Quanto às questões de projeto, salientamos a possibilidade de tornar o projeto e manutenção do *kernel* de um tutor (suas estratégias e táticas) mais flexíveis através de uma categorização de elementos (jogo e da interface). Não objetivamos a possibilidade de extensão para outros domínios, mas ampliação gradativa das possibilidades de exploração do domínio e da interface. Para comprovarmos estas hipóteses desenvolvemos um protótipo, o E-MCOE, a fim de analisarmos as potencialidades pedagógicas, o ganho na qualidade dos serviços prestados pelo tutor com o auxílio do agente mediador, e os prováveis frutos com a descrição do sistema para trabalhos futuros. Desta forma, poderemos saber se a extensão feita na arquitetura implicou em ganhos no que diz respeito ao potencial exploratório do ambiente sob o ponto de vista pedagógico, modelagem dos agentes cognitivos e performance do sistema. Como exemplo dos ganhos citamos: decréscimo da complexidade em modelar, implementar os agentes cognitivos e enriquecimentos das potencialidades da representação do domínio.

Como trabalhos futuros, identificamos que processos de negociação e sincronismo entre computadores, distribuídos poderão trazer novos resultados para este trabalho.

A fim de viabilizar um teste com alunos, como realizado por [5], pretende-se elaborar um novo projeto para contemplar as extensões do E-MCOE para todas as fases do MCOE.

5. Referências

[1] AIMUER, E.; FRASSON, C. Analysing a new learning strategy according to different knowledge levels. Computer Education, Londres, v.27, n.2, 1996.

[2] CALLEGARI, D. A. Aplicando aprendizagem por reforço a uma arquitetura multiagente para suporte ao ensino de educação ambiental. PPGCC/PUCRS, Porto Alegre, 2000. Dissertação de Mestrado.

[3] FRASSON, C.; MENGELLE, T.; AIMEUR, E. Using pedagogical agents. In: WORLD CONFERENCE ON ARTIFICIAL INTELLIGENCE IN EDUCATION, 1997, kobe, Japão. Anais. . . 1997.

[4] FUNGE, John David. AI for Computer Games and Animation – A Cognitive Modeling Approach. A. K. Peters Ltd, 1999.

[5] GIRAFFA, L. M. M. Uma arquitetura de tutor utilizando estados mentais. CPGCC/UFRGS, Porto Alegre, 1999. Tese de Doutorado.

[6] GIRAFFA, L. M. M.; VICCARI, ROSA M. Estratégias de Ensino em Sistemas Tutores Inteligentes modelados através de agentes. SBIE 1998 – Simpósio Brasileiro de Informática na Educação, Fortaleza ,Ceará; Anais… 1998.

[7] GIRAFFA, L.M.M; MORA, M.; VICCARI, R.M. Modelling an interactive ITS usinga MAS approach: from design to pedagogical evaluation. 3rd International Conference on Computational Intelligence and Multimedia Applications - ICCIMA'99. Proceedings... New Delhi, India, September 1999.

[8] LIEBERMAN, H. Autonomous interface agents. CHI, 1997, p.67–74, Anais. . . 1997. v.1.

[9] MARIETTO, MARIA G. B.; ORNAR, NIZAM. Definição dinâmica de estratégias instrucionais em sistemas de tutoria inteligentes: uma abordagem multiagentes na WWW. SBIE 2000 – SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 2000, Maceió, Alagoas, p.154–159, Anais. . . 2000.

[10] MITROVIC, A.; SURAWEERA, P. Evaluating an Animated Pedagogical Agent. Em: ITS 2000 – Intelligence Tutoring Systems, 2000, Montreal, Canada. Anais. . . 2000. p.73–82.

[11] MÓRA, M. C.; Um modelo de agente executável. Porto Alegre: CPGCC/UFRGS, 2000. Tese de Doutorado.

[12] PEREIRA, ADRIANA S.; GEYER, CLAUDIO F. R. Um Agente para Seleção de Estratégias de Ensino em Ambientes Educacionais na Internet. Em: IBERAMIA – SBIA 2000, 2000, Atibaia, São Paulo. Anais. . . 2000. p.362-369.

[13] ROSATELLI, M. C. Um ambiente inteligente para aprendizado colaborativo no ensino a distância utilizando o método de casos. Florianópolis: PPEP/UFSC, 1999. Tese de Doutorado.

[14] SELF, JOHN. The deøning characteristics of intelligent tutoring systems research: itss care, precisely. International Journal of Artificial of Artificial Intelligence in Education, Leeds, England, 1999.

[15] Schneider, J.; Nierstrasz, O. Components, Scripts and Glue. Software Architectures — Advances and Applications, Leonor Barroca, Jon Hall and Patrick Hall (Eds.), pp. 13-25, Springer, 1999.

[16] SILVEIRA, Ricardo A. Modelagem Orientada a Agentes Aplicada a Ambientes Inteligentes Distribuídos de Ensino. 2001. Tese (Ciência da Computação) - Universidade Federal do Rio Grande do Sul, (Orientador) Rosa Maria Viccari.

[17] TEDESCO, P. A.; SELF, J. Using Meta-Cognitive Cnflicts to support Group Problem Solving; Intelligent Tutoring Systems, ITS 2000; p.232-241; Anais … 2000.

[18] WENGER, E. Artificial intelligence and tutoring systems. [S.l.]: Morgan Kaufmann Publishers, INC, 1987. p.486.

[19] WEXELBLAT, A.; MAES, P. Issues for software agent ui. Disponível por WWW In http://wex.www.media.mit.edu/people/wex/agent-uipaper/agent-ui.htm (1997).

[20] WOODCOCK, S. Game AI: The State of the Industry. Capturado em Março de 2001. Online. Disponível na Internet em: http://www.gamasutra.com/features/19990820/game_ai_01.htm

[22] ZOUAQ, A.; FRASSON, C.; ROUANE K. The Explanation Agent. Intelligent Tutoring Systems, ITS 2000, p.554-564, Anais … 2000.