FAmCorA: um framework para a construção de ambientes cooperativos inteligentes de apoio a aprendizagem na Internet baseado em *web services* e agentes

José Marques Pessoa[1,2], Hylson Vescovi Netto[2], Crediné Silva de Menezes[2]

[1]ICLMA – UFMT
Rodovia MT-100, Km 4 – 78600.390 – Pontal do Araguaia – MT – Brasil
jmpessoa@npd.ufes.br

[2] PPGEE – UFES
Av Fernando Ferrari, s/n, 29060-900 – Vitória – ES – Brasil
hvescovi@bol.com.br; credine@inf.ufes.br

Resumo. O desenvolvimento de novos Ambiente Virtuais de Aprendizagem demanda recursos de tal ordem que dificulta o surgimento de propostas inovadoras. Em conseqüência há um grande *gap* entre as concepções teóricas e o que temos de fato implementado. O presente trabalho apresenta uma proposta de framework para o desenvolvimento de aplicações do tipo *CSCL*, seja através do reuso dos provedores de serviços (*web services*) implementados pelo *framework*, seja pelo compartilhamento de aplicações entre vários sistemas. O *framework* está baseado na composição de um conjunto de tecnologias: *cgi*, *web services* e agentes. Um protocolo de notificação/comunicação inter-aplicações permite que as aplicações criada segundo esse *framework* possam chamar outras aplicações em outros ambientes, apenas apontando para um *link* e ainda assim manter o controle e monitoramento das ações do usuário, fato necessário em ambientes *CSCL* para a manutenção do *log* das interações. Essa estratégia é fundamental para o surgimento de novas abordagens para ambientes educacionais, simplesmente porque agora podemos construi-los facilmente através de composição de aplicações desenvolvidas em diferentes ambientes, sob diferentes concepções pedagógicas.

Palavras-chave: Ambientes Virtuais de Aprendizagem; Webservices; Telemática e Educação

1. Introdução

Os ambientes telemáticos de suporte à aprendizagem colaborativa constituem-se em uma das mais proeminentes inovações decorrentes da moderna tecnologia da informação e comunicação.

As redes de computadores, em particular a Internet, ampliaram as possibilidades para a construção de ambientes cooperativos nos quais pesquisadores, professores e estudantes se relacionam, visando a troca e a aquisição de conhecimento.

Segundo Bonk & King, citados em [Lehtinen], a rede pode: i) mudar a maneira com que aprendizes e professores interagem; ii) aumentar as oportunidades para a aprendizagem colaborativa; iii) facilitar as discussões; iv) deslocar o estudo de um processo isolado na direção de uma aprendizagem mais social e ativa.

[Rowley 1997] afirma que "as variáveis da aprendizagem colaborativa têm sido extensivamente estudadas, entretanto, é seguro dizer que ainda estamos no estágio inicial com respeito ao projeto e desenvolvimento de ambientes CSCL (*Computer-Supported*

Collaborative Learning". [MCNAUGHT 2001] vai até mais longe quando escreve que "há poucos recursos que provêm *insights* (ambos técnico e teórico) para o desenvolvimento de um tal ambiente de aprendizagem". [Cook] escreve "não temos ainda conhecimento suficientemente detalhado acerca das relações entre teoria, trabalhos empíricos e implementações de ambientes de aprendizagem".

Nos dias atuais, típicos ambientes telemáticos de suporte a aprendizagem cooperativa na web são dotados, com pouca variação, das seguintes funcionalidades, algumas já bem conhecidas na rede: páginas html, *upload/download* de arquivos, *email*, *chat*, *fórum de discussão*, *mural*, *news* e *mensagens instantâneas* [Lehtinen], [Rowley 1997], [Crow 1997], [George], [Jermann 2001], [Koschmann 1996] [Santoro 1998].

Ainda que tais aplicações possam ser vistas como *padrões* em projetos de ambientes dessa natureza, é freqüente a re-implementação de cada uma delas a cada novo ambiente gerado. Isto tem ocorrido em parte devido à arquitetura da *web*, na qual prevalece a existência de aplicações encadeadas por *links* mas isoladas logicamente. Isto é, a comunicação entre aplicações via *link* na *web* possui apenas um contrato de ida, sem nenhuma garantia de volta. Nesse cenário o reuso torna-se impraticável para as necessidades dos ambientes *CSCL*, que requerem o controle e o monitoramento das interações para futuras análises [Guzdial], [Stahl 2002].

O desenvolvimento de novos ambientes virtuais de aprendizagem demanda recursos de tal ordem que dificultam o surgimento de propostas inovadoras. Em conseqüência há um grande *gap* entre as concepções teóricas e o que temos de fato implementado. Em busca de uma estratégia para equacionar o problema estamos desenvolvendo o FamCorA, um *framework* para o desenvolvimento de aplicações para ambientes *CSCL* baseado na *web*. Entre suas principais inovações destacamos: a) viabilização do reuso e, b) o compartilhamento e a comunicação inter-aplicações através da composição de um conjunto de tecnologias: *cgi*, *web service* e agentes.

O texto está assim organizado: A Seção 2 apresenta algumas soluções que vêm sendo utilizadas para a construção de ambientes *groupware*. A Seção 3 apresenta uma discussão sobre as tecnologias correntes para o desenvolvimento de aplicações na *web*. A Seção 4 apresenta o FAmCorA: um *framework* para o desenvolvimento de ambientes *CSCLs* na *web*. A Seção 5 apresenta um estudo de caso, a aplicação *e-group*. A Seção 6 discute as conclusões.

2. Tecnologias de Desenvolvimento de Ambientes Groupware

As tecnologias para o desenvolvimento de aplicações *groupware* têm apontado basicamente em três direções: componentes reutilizáveis de *software*, *frameworks* baseados em ambientes de programação e aplicações *web*.

2.1 Soluções Baseadas em Componentes

[Farias 2000] apresenta uma metodologia para o desenvolvimento de ambientes *groupware* baseada em componentes pré fabricados. Tais componentes são entendidos como elementos autocontidos, com finalidades bem definidas e com capacidade para serem utilizados tanto isoladamente quanto em composição com outros componentes. De modo semelhante [Rees 1999] apresenta a arquitetura *COGAM* (C*omponent-based Groupware Architectural Model*) voltada especificamente para a plataforma *Windows* e seus objetos *COM* (*Common Object Model*).

2.2 Soluções Baseadas em Ambientes de Programação

Alguns *frameworks* são específicos para um dado ambiente de programação, por exemplo, *GroupKit* [Roseman 1997] oferece um framework de desenvolvimento para ambientes *groupware* baseado na linguagem de programação *Tcl*. Habanero [Chabert 1998] é um framework baseado em Java, assim como *JlearningServices* [Rheinheimer 2001].

2.3 Soluções Baseadas na *Web*

Dourish, citado em [Crow 1997], concorda com [Newman 1996] quando propõe que os esforços no desenvolvimento de aplicações para dar suporte a atividades em grupo sejam dispendidos na busca por ferramentas que possam ser juntadas para enriquecer a experiência com o uso da *web*. Isto porque, mesmo com suas limitações, a *web* possui enormes vantagens sobre outras tecnologias; a mais citada é que os programas clientes (*browsers*) estão disponíveis em praticamente todas as plataformas e sistemas operacionais e são de fácil utilização. As sessões seguintes discutem as tecnologias que dão suporte à programação de aplicações na *web*.

2.3.1 HTTP (Hypertext Transmission Protocol)

O *HTTP* é o protocolo sobre o qual está construída a *web*. É o responsável pela comunicação entre as aplicações servidoras de

páginas (servidores *web*) e os leitores de páginas (*browsers*) e em consequência por toda a audiência da Internet. No entanto possui algumas limitações: 1) não garante as taxas de transmissão entre o cliente e o servidor, necessárias para a boa transmissão de vídeo e som ao vivo. 2) não oferece nenhum suporte direto para replicação da informação, a qual pode ser requerida em aplicações que requerem imediato *feedback*. 3) não suporta comunicação servidor/servidor, servidor-iniciado/cliente ou ainda cliente/cliente, o que é uma dificuldade para aplicações em que o servidor deve desempenhar um papel ativo, por exemplo, notificando o cliente sobre mudanças na base de informação. Ou ainda um cliente precisando notificar outro cliente sobre as atividades de um usuário em uma outra aplicação. 4) não guarda o estado ou contexto (sessão) da aplicação entre uma requisição e outra. 5) Do lado cliente, ainda que o HTML suporte a construção de formulários, isto é feito de maneira bastante limitada se comparado com as modernas *GUIs* [BENTLEY 1997], [DALGARNO 2001], [Clarke 2001], [HTTP].

Ainda que tais limitações acabem se tornando cruciais para a grande maioria das aplicações que vêem na web o melhor palco para exibir suas performances, devido ao alcance da rede Internet, é exatamente por sua simplicidade que o HTTP tornou-se um protolo de sucesso, em parte devido a um pequeno coadjuvante, o CGI.

2.3.2 CGI (*Common Gateway Interface*)

A finalidade básica de um servidor *web* é servir as chamadas páginas *html* ao cliente (*browser*). Porém nem sempre a página requisitada está prontamente disponível. Por várias razões pode ser interessante e útil a composição de uma página sob demanda; o protocolo CGI é o responsável por essa flexibilidade [CGI].

O CGI é o protocolo de comunicação através do qual o servidor web intermedia a transferência de informações com um programa, chamado *gateway*, ou script. O script recebe e processa requisições do servidor web, retornando, em geral, um documento HTML dinamicamente gerado a partir de bases de dados. Pelo papel que desempenha e como ele pode alcançar outras aplicações, por exemplo um servidor de banco de dados, para cumprir com a sua missão, um script é também chamado de *extensão* do servidor web.

Os programas *scripts* são os verdadeiros responsáveis pelo dinamismo atual das aplicações na web e pelo sucesso da dupla Browser-Servidor Web e de alguma forma são os precursores de uma nova onda na Internet, os chamados *web services*.

2.3.3 Web Services

Web Services é uma aplicação *SOAP* (*Simple Object Access Protocol*), um protocolo baseado em *XML* capaz de invocar uma função (serviço) sobre um protocolo Internet, usualmente o *HTTP*.

Os *Web Services* foram projetados para atenderem os seguintes objetivos: 1) a implementação do serviço nunca pode ser vista pelo lado do consumidor (encapsulamento). 2) A mudança na implementação do serviço não requer mudança no consumidor do serviço (baixo acoplamento). 3) A descrição do comportamento do serviço, como se ligar a ele, quais são os seus parâmetros de entrada e quais são as suas saídas, estão publicamente disponíveis (contratável).

Um *Web Services* pode usar outros *Web Services* para realizar a sua tarefa (composição baseada no uso de serviços). Assim, uma nova aplicação pode ser construída, simplesmente, compondo e integrando serviços disponíveis na rede.

Uma aplicação *Web Services* está construída sobre padrões negociados no consórcio *W3C* (*World Wide Web Consortium*); isto significa que qualquer sistema que suporte *Web Services* pode trabalhar com qualquer outro sistema que também suporte *Web Services* independentemente do tipo de *hardware*, sistema operacional e linguagem de programação.

Por padrões entende-se que todos os *Web Services* têm suas interfaces de programação descritas usando *WSDL* (*Web Services Definition Language*), isto é, a *WSDL* padroniza como deve ser formatada a requisição e a resposta de um *Web Services*. Todos os *Web Services* publicam/anunciam suas capacidades em um repositório *UDDI* (*Universal Discovery Description Integration*). O *UDDI* registra o serviço e os detalhes técnicos de como acessá-lo (meta-dados) [Gunzer 2002], [McGee 2002], [Gambhir 2001].

Para [Glass 2000] "As aplicações do futuro serão construídas a partir de *Web Services* dinamicamente selecionados em tempo de execução, baseados no custo, qualidade e disponibilidade".

A seguir apresentamos uma proposta de *framework* para a construção de ambientes *CSCL* na *web* que reúne as tecnologias *CGI*, *Web Sevice* e *Agentes* para viabilizar o reuso, o compartilhamento e a comunicação inter-aplicações na *web*. A proposta está baseada no reuso de aplicações provedoras de serviços (*Web*

Services) mas vai além quando propõe o reuso de aplicações baseadas em um protocolo de notificação/comunicação entre agentes.

3. FAmCorA: Um Framework para a Construção de Ambientes Cooperativos de Aprendizagem na Web

O FAmCorA busca contemplar as necessidades da construção de ambientes cooperativos de aprendizagem através de um conjunto de aplicações provedoras de serviços (*Web Services*) e de uma arquitetura que possibilita a comunicação inter-aplicações.

Um provedor de serviço está implementado com a tecnologia de W*eb Services;* nesse sentido, no FAmCorA, o reuso dá-se no nível do consumo de serviços (*API/Application Program Interface*), que podem residir em qualquer parte da *web*, diferentemente do modelo de objetos ou componentes, nos quais o reuso dá-se através do reaproveitamento de código. Porém o FAmCorA vai mais longe e propõe o reuso de aplicações inteiras a partir de um protocolo de notificação que possibilita o monitoramento e a troca de informações entre aplicações.

A partir da nossa experiência no desenvolvimento de um ambiente de aprendizagem chamado AmCorA [Menezes 2000] e do estudo da literatura de referência, chegamos a um conjunto inicial de aplicações que estão comtempladas no *framework*: Provedores de Serviços Básicos, Provedores de Engenho e Provedores de Aplicação.

3.1 Provedores de Serviços Básicos

Trata-se de um conjunto de aplicações básicas: Gerenciador de Banco de Dados, Servidor de Email, Sevidor de Lógica e de um sub-conjunto de *APIs* do Sistema Operaracional (Sistema de Arquivos), sobre as quais são construídos os ambientes cooperativos na *web*. O FAmCorA encapsula estas funcionalidades em *Web Services*.

A Figura 1 a seguir, apresenta alguns provedores de serviços básicos.

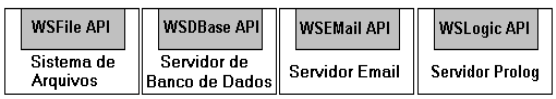

Figura 1. *Web Services* Básicos.

O provedor *WSFile* é uma camada de abstração para as operações básicas do sistema de arquivos. O provedor *WSDBase* é uma camada de abstração para o sistema gerenciador de banco de dados. O provedor *WSEMail* é uma camada de abstração para o servidor de *email*, disponibilizando os serviços tais como enviar, ler, validar emails. O provedor *WSLogic* é uma camada de abstração para o servidor de deduções (máquina prolog).

3.2 Provedores de Engenho

Um engenho encapsula um tipo de aplicação capaz de realizar algum tipo de raciocínio inteligente/automático em benefício do usuário. Os seguintes engenhos têm se mostrado de grande utilidade para a construção de ambientes computacionais inteligentes:

IndexSearch Engine: *IndexSearch Engine* é uma máquina de propósito geral para indexar e pesquisar documentos. A busca pode ser feita por uma ou mais palavras chaves combinadas por operadores lógicos (*and/or*) em um sistema de arquivo local ou remoto. Utiliza uma técnica conhecida por árvore invertida [Watson 1996].

LatentSemantic Indexing Engine: é uma máquina que implementa a indexação baseada no princípio de que um documento possui uma semântica latente [LSI].

MBR Engine: é uma máquina que implementa o *MBR* (*Memory Based Reasoning*/Raciocínio Baseado em Memória), uma técnica do tipo *k-nearest-neighbors* (*vizinho mais proximo*). Coletando dados estatísticos em situações passadas e as respectivas ações tomadas, a *MBR engine* surge com uma predição e, possivelmente, o seu grau de confiança nesta predição. Uma situação é representada por um conjunto de atributos, cujos valores em um dado instante caracterizam um estado. A distância entre uma nova situação e uma situação memorizada é calculada como uma soma ponderada das distâncias entre os valores de cada atributo [Stanfill 1986].

Traductions Engine: *Traductions Engine* é uma máquina de tradução. De fato, trata-se de uma interface baseada em *Web Services* para alguns *sites* populares especializados em tradução. A figura 2 a seguir, apresenta a arquitetura desses serviços.

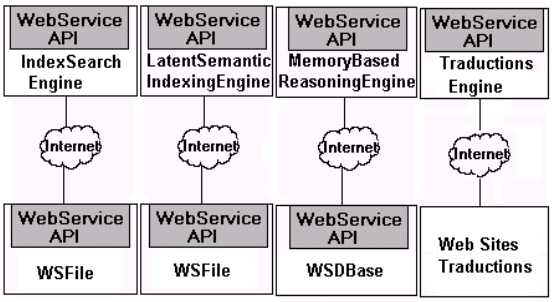

Figura 2. *Web Services* Provedores de Engenho

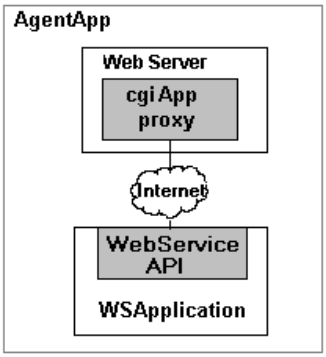

Figura 3. Agente Provedor de Aplicação

3.3 Provedores de Aplicação

Os provedores de aplicação, de fato, surgem primeiramente enquanto solução para um dado domínio de problema, entretanto não é surpresa que alguns desses domínios tornam-se recorrentes, configurando assim um padrão (*pattern*) que pode ser reutilizado como parte da solução de outros domínios de problemas. Para viabilizar este ciclo, no FAmCorA, as funcionalidades básicas de uma aplicação são modeladas e implementadas primeiramente enquanto uma estrutura provedora de serviços (*Web Services*), em seguida, um *proxy cgi*, que implementa uma interface para o W*eb Services* subjacente, é aí acoplado atuando no papel de consumidor do serviço e elemento de comunicação/notificação inter-aplicação, dando origem a um agente.

Agente é um termo que tem sido utilizado para nomear um tipo de programa capaz de realizar autonomamente ações em busca do objetivo para o qual foi concebido. Para os objetivos do presente trabalho, agentes são programas que assistem um usuário (ou outro agente) durante a realização das suas atividades, seja através da automação de tarefas rotineiras e enfadonhas, seja notificando sobre eventos importantes que precisam de atenção ou ainda comunicando-se com outros agentes (ou humanos) para mantê-los informados sobre modificações no ambiente. [Thaiupathump 1999], [Keeble 2000], [Maes 1994].

A figura 3 a seguir, apresenta a arquitetura desse tipo de aplicação.

As seguintes aplicações estão projetadas no *framework*:

Agente Forum, Agente Chat, Agente Mural (Quadro de Avisos), Agente Agendador de Reuniões, Agente Editor de Questionários, Agente Editor de Mapas Conceituais, Agente de Email, Agente de Arquivo, Agente QSabe, Agente Mediador de Discussão Centrada em Documentos, Agente *BigBrother*, Agente *Link* e o Agente Sabiá.

A seguir descrevemos as funcionalidades de algumas dessas aplicações:

Agente de Email: Oferece os serviços básicos de enviar, receber, filtrar e indexar mensagens, além de possuir uma agenda para disparo automático de mensagens. A importância de tal ferramenta na construção de ambientes telemáticos de aprendizagem é ressaltada, por exemplo, em [George] "Os computadores provêem a oportunidade para aprender a se comunicar, bem como possibilitam que a comunicação melhore a experiência do aprender".

Agente de Arquivo: O Agente de Arquivo oferece os serviços básicos de um sistema de arquivos, tais como, mover arquivos (*upload/download*) bem como criar/remover pastas e os serviços de indexação e busca. Em ambientes de aprendizagem são úteis como repositório para o compartilhamento de documentos.

Agente QSabe: A troca de informações é a base da cooperação em ambientes de aprendizagem suportado por computador. Uma ferramenta central nesse processo é uma que dê suporte inteligente à formulação de perguntas e respostas. O QSabe [Pessoa 2000] utiliza teorias e técnicas de compreensão de texto e aprendizagem automática para aprender dinamicamente o perfil de cada participante em um ambiente de

aprendizagem de modo a endereçar as questões ao indivíduo mais apropriado.

Agente Sabiá: Segundo [Hooper 2000] a maneira como o aprendiz faz suas anotações durante a leitura afeta significativamente o valor dessa atividade. Entretanto os navegadores padrões (*browsers*) não possuem recursos especializados que auxiliem na marcação, indexação, inserção e anotações marginais e outras tarefas de manipulação do texto contido em sua área de trabalho. Na prática, a maior parte das leituras *online* ficam apenas como vaga lembrança na mente do estudante! O Agente Sabiá [Goulart 2001] é uma ferramenta que incorpora, automaticamente, *scripts* compatíveis com o *Document Object Model* do consórcio *W3C* (*World Wide Web Consortium*) [DOM] a um documento html, o que permite a interação inteligente com o texto sendo visualizado no *browser* e também com a coleção de documentos armazenada no servidor; entre outras funcionalidades o Sabiá permite modificar, destacar e comentar partes do texto e armazenar as anotações com referências para o documento. Tais tarefas estão relacionadas com o chamado "fichamento" de artigo.

Agente Mediador de Discussão Centrada em Documentos: Uma anotação à margem do documento quando compartilhada em ambientes de aprendizagem é também um poderoso suporte para a comunicação e a colaboração [Cadiz 2000]. A idéia central é que uma anotação compartilhada providencia um fórum privilegiado de comunicação/discussão para um grupo de trabalho. [Gay 1999] ressalta que "em um ambiente de aprendizagem colaborativa o conhecimento não é transmitido do professor para o aprendiz mas, antes, o conhecimento é criado em um diálogo ativo entre todos aqueles que buscam entender e aplicar os conceitos". [Lehtinen] afirma ainda que "do ponto de vista cognitivo é particularmente importante transformar processos de investigação em uma forma pública para que eles possam ser examinados e imitados". O Agente Mediador é uma ferramenta de grupo que incorpora, automaticamente, *scripts* compatíveis com o *Document Object Model* do consórcio W3C [DOM] a um documento html, o que permite a interação inteligente com o texto sendo visualizado no *browser* e também com a coleção de documentos armazenada no servidor; entre outras funcionalidades o Mediador permite modificar, destacar, comentar, questionar partes do texto e armazenar as interações com referências apontadas para o local em que elas ocorreram no documento, além disso, notifica os participantes sobre as atividades realizadas em torno do documento.

Agente *BigBrother*: Notifica sobre quais os participantes estão *online* para troca de mensagens instantâneas.

Agente *Link*: Atua em *background* "plantando" *links* relacionados aos documentos do usuário, inspirado em [BROWN 1996].

4. Arquitetura do FAmCorA

A arquitetura *Model-View-Controlle* (MVC) foi originalmente desenvolvida para mapear a tradicional tarefa de entrada, processamento e saída do modelo de interação baseado em interfaces gráficas [Beck 1996]. No entanto, é perfeitamente possível mapear estes conceitos no domínio de aplicações multi-camadas baseadas na *web* .

O FAmCorA toma como partida esse paradigma e propõe uma arquitetura em camadas que visa simplificar e dar unicidade à construção de aplicações para ambientes de aprendizagem cooperativa na *web*. Na camada de base encontram-se os chamados *provedores de serviços (Web Services)* que são utilizados para comporem a camada do *modelo*, aqui chamada de provedor de aplicação, sobre a qual são construídas as interfaces das aplicações (*visão-controles*). Os elementos constitutivos de cada camada são de fato *padrões* recorrentes percebidos no estudo da literatura de referência [Santoro 1998]. *Padrões* aqui são entendidos como *design patterns* (padrões de projeto) ou seja pedaços de software que se repetem dentro de um domínio de problema [Buschmann 1996].

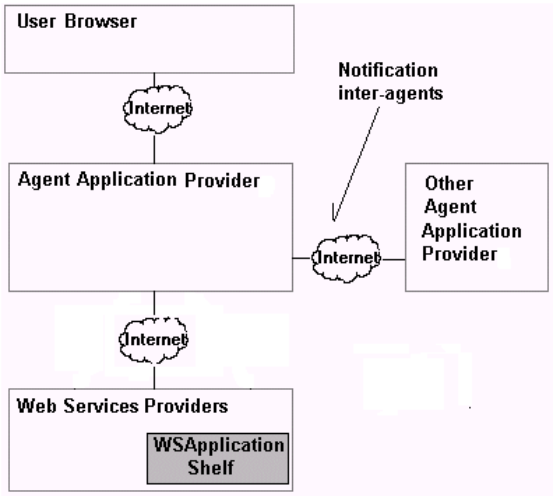

Figura 4 – Arquitetura do FamCorA

A figura 4, apresenta uma visão macro da arquitetura do FAmCorA. Nesta arquitetura, um agente provedor de aplicação está modelado a partir de serviços oferecidos pela camada de provedores de serviços.

O protocolo de notificação/comunicação inter-agente permite que uma aplicação munida de um *passport*, definido pelo framework, chame outra aplicação e ainda assim mantenha uma trilha das ações do usuário na outra aplicação, o que abre espaço para várias possibilidades, por exemplo: reuso da mesma aplicação em diversos ambientes *CSCL*, manutenção/monitoramento do *log* das interações mesmo quando o usuário se desloca para uma outra aplicação, oferecer alguma ajuda ao usuário para lidar com uma aplicação desconhecida, incluindo aí chamar de volta o usuário, etc. Desse modo o FAmCorA supera uma dificuldade imposta pelo *HTTP* em relação à comunicação servidor/servidor e oferece suporte para o compartilhamento de aplicações entre vários ambientes.

Internamente um agente provedor de aplicação no FAmcorA é composto de dois módulos: um W*eb Services* que modela o domínio da aplicação e uma aplicação *cgi* que funciona como um *proxy*, implementando as interfaces para o W*eb Services* subjacente e é responsável pela comunicação inter-aplicação, como exemplificado a seguir na seção 5.

O provedor de serviços *WSApplication-Shelf* (prateleira de aplicações) é uma espécie de "páginas amarelas" do *framework*, ele mantém o registro do conjunto de aplicações disponíveis. De fato, uma aplicação é tida como compatível com o FAmCorA quando ela implementa as interfaces de *WSApplication-Shelf* e o protocolo de notificação/comunicação definido pelo *framework*.

[BENTLEY 1997] sugere cinco tipos de eventos para "perceber" em detalhes "o que" foi feito dentro do ambiente, "quando" e por "quem" em relação a um dado objeto: *New*: o objeto foi criado; *Read*: o objeto foi postado/lido por alguém ; *Change*: o objeto foi modificado (''*edited*'', ''*renamed*'', ''*changed description*'' etc); *Move*: o objeto foi movido ("*changed location*", ''*deleted*'' and ''*undeleted*'', ''*cut*'' e ''*dropped*''); *Touch*: o objeto foi tocado. No FAmCorA o mecanismo de notificação expande essa noção, assumindo de fato, uma espécie de linguagem de comunicação inter-agentes, um subconjunto da *KQML*(*Knowledge Query and Manipulation Language*) [KQML].

A seção seguinte discute a estrutura das aplicações no FAmCorA a partir de um estudo de caso: uma aplicação *e-group*.

5. Estudo de Caso: Aplicação *e-groups*

[Hiltz 1998] avalia que o projeto de um ambiente *online* que suporta a aprendizagem colaborativa é mais efetivo do que aquele cuja ênfase dá-se a partir do estudo individual, e que a premissa básica do ensino *online* deve ser a construção de comunidades de aprendizes para facilitar a troca de idéias, informações e sentimentos. Ou como escreve [Thaiupathump 1999] "a ênfase deve estar nas pessoas aprendendo com outras pessoas, via uma rede de aprendizagem". [George] também assinala que "aprendizes aprendem a trabalhar coletivamente compartilhando, organizando, planejando e coordenando tarefas". Ou ainda "o componente básico da abordagem baseada na aprendizagem colaborativa é a aprendizagem em grupo" (Johnson & Johnson) citado em [Jianhua].

As afirmações e avaliações acima indicam que um elemento central que devemos modelar na construção de ambientes cooperativos de apoio à aprendizagem é a noção de grupo. Aqui os seguintes termos também são entendidos com o mesmo significado: projeto, turma, disciplina, evento, etc.

Ambientes de apoio à construção de grupos, os chamados *e-groups*, não são novidades na *web*. Porém, tais ambientes, por serem de propósito geral, carecem de requisitos convenientes para as atividades de ensino e aprendizagem. Além de mecanismos inteligentes/automáticos de acompanhamento, auxílio e notificação das atividades/interações desenvolvidas por cada um participante/aprendiz.

A Figura 5 a seguir, apresenta o Agente de Grupo como sendo um estudo de caso para o esquema geral de uma aplicação no FAmCorA.

Para viabilizar o ciclo de reuso no FAmCorA, as funcionalidades básicas de uma aplicação são modeladas e implementadas primeiramente enquanto uma estrutura provedora de serviços. Na Figura 5, o W*eb Services WSGroup* (*model*) expressa essa idéia. Um provedor de aplicação, no caso *WSGroup*, anuncia/registra os seus serviços em *WSApplicationShelf*, a prateleira de aplicações.

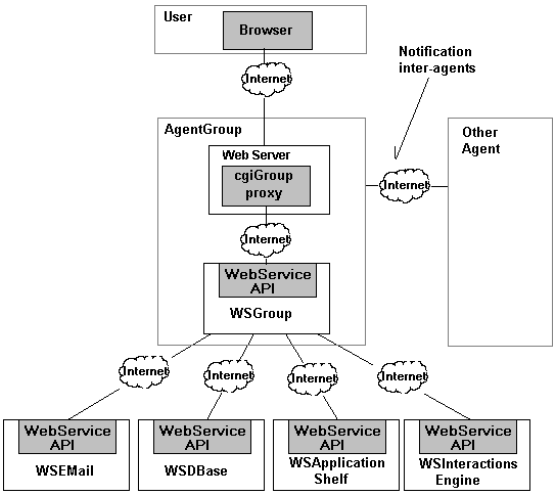

Figura 5. Uma aplicação no FAmcora: O Agente de Grupo.

A aplicação, nesse caso *AgentGroup*, é uma composição dos serviços oferecidos pelo *web service* provedor de aplicação WSGroup e de uma interface cgi *cgiGroup*, responsável pela construção da interface com o usuário e pelo mecanismo de notificação entre aplicações. Note ainda na figura 5 que WSGroup é por sua vez uma composição de *web services* básicos, no caso WSEMail, WSDBase, WSApplicationShelf e WSInteractionsEngine.

O AgentGroup provê uma interface (*view-controls*) com o usuário via *browser* e se comunica com outros agentes através de um protocolo de notificação.

Dentre outras funcionalidades, o *AgentGroup* coordena a criação de grupos, *logins*, a distribuição de *passport* para outras aplicações (na figura 5 OtherAgent) e guarda a memória das interações em *WSInteractionsEngine* para futuras inferências/deduções. O mecanismo de notificação entre aplicações permite ao Agente de Grupo acompanhar as interações de um participante mesmo quando este se desloca para outra aplicação compatível com o framework. Assim o ambiente pode prover o monitoramento das interações oferecendo para a comunidade de mediadores e aprendizes uma espécie de "consciência" do trabalho individual e de grupo.

O AgentGroup pode ser entendido como um portal (ou *broker* como sugere [Navarro 2001]) que intermedia uma rede de interações em um complexo ambiente de aprendizagem multi-aplicação.

Para exemplificar e possibilitar uma melhor compreensão do framework proposto, apresentamos alguns cenários que revelam o seu funcionamento interno.

5.1 Cenários de Uso

Cenário 1: Usuário solicita a criação de um grupo

1 – Usuário *anonymous* invoca *AgentGroup* (internamente *cgiGroup*), através de uma *url*, para conhecer quais são as informações necessárias para a criação de um grupo;

2 – *cgiGroup* invoca a interface *ILogin* em *WSGroup* para conseguir um *passport* de acesso a esse serviço.

2.1 – *cgiGroup* invoca a interface *IInfoPedido* em *WSGroup* para conseguir quais são os campos do formulário de preenchimento.

2.1.1 - *WSGroup* invoca a interface *ILogin* em *WSDBase* para conseguir um *passport* de acesso a esse serviço.

2.1.2 – *WSGroup* invoca a interface *IInfoTable* em *WSDBase* para conseguir os nomes dos campos que devem ser preenchidos no formulário. (obs. *WSGroup* mantém um *log* apenas com os nomes das tabelas que ele criou em *WSDBase*);

3 – *AgentGroup* (internamente *cgiGroup*) apresenta o formulário *html* ao usuário para preenchimento;

4 – Usuário submete o formulário preenchido ao *AgentGroup* (internamente *cgiGroup*) solicitando a criação de um grupo;

5 – *AgentGroup* notifica *AgentRoot* (um assistente virtual do *root* do sistema) sobre o pedido de criação do grupo.

6 – *AgentRoot* baseado em casos passados aprova (alternativamente: reprova, ou solicita orientação do root do sistema) o pedido de criação do grupo notificando *AgentGroup* (internamente *cgiGroup*) e via email notifica o root do sistema sobre a sua atividade.

7 – *cgiGroup* invoca a interface *IApproved* em *WSGroup* ;

8 – *AgentGroup* invoca três vezes a interface *IInsert* em *WSDBase* para criar o usuário (Entidade User) e o grupo (Entidade *Group*) e o papel (coordenador) do participante (Entidade *UserGroup*);

9 – *AgentGroup* (internamente *WSGroup*) invoca a interface *IGetApplicationPassport* em

WSApplicationShelf para conseguir *passport* para cada uma das aplicações disponíveis no *framework*.

10 – *AgentGroup* invoca a interface *IInsert* em *WSDBase* para guardar o *passport* do usuário para cada uma das aplicações disponíveis (Entidade *UserApplicationPassport*);

Cenário 2: Usuário login

1 – Usuário de *login* "aprendiz@mail.edu.br" e senha "1234" invoca *AgentGroup* (internamente *cgiGroup*), através de uma *url*, para ter acesso a sua *home* (sua área pessoal).

2 – *cgiGroup* invoca a interface *ILogin* em *WSGroup* para logar o usuário.

2.1 – *WSgroup*, invoca a interface *ISelect* em *WSDBase* para conseguir os *passports* das aplicações disponíveis para o usuário (Entidade *UserApplicationPassport*);

3 – *cgiGroup* monta a *home* do usuário com *links* (menu) para todas as aplicações para as quais o usuário tem um *passport*, por exemplo, na figura 4 a aplicação *OtherAgent*.

Cenário 3: Usuário invoca a Aplicação *OtherAgent*

1 – Estando na aplicação *AgentGroup*, o usuário invoca o *link* para a aplicação *OtherAgent* (internamente *cgiOther*) no menu;

2 – *cgiOther* invoca *ILogin* em *WSOther*;

2.1 – *OtherAgent* (internamente *cgiOther*) notifica *AgentGroup* (internamente *cgiGroup*) que o *login* foi realizado;

2.2 – *cgiGroup* invoca *IITeraction* em *WSGroup*;

2.3 – *WSGroup* invoca *IAssert* em *WSIteractionsEngine* para o *login* do usuário em *OtherAgent*;

3 – Usuário invoca uma funcionalidade, digamos, "Foo", na aplicação *OtherAgent*;

4 – *cgiOther* invoca "*IFoo*" em *WSOther*;

4.1 – *cgiOther* notifica *cgiGroup* que "*Foo*" foi realizada;

4.2 – *cgiGroup* invoca *IITeraction* em *WSGroup*;

4.3 – *WSGroup* invoca *IAssert* em *WSIteractionsEngine* para a realização de "Foo" do usuário em *OtherAgent*;

6. Conclusões e Perspectivas

Os ambientes computacionais de suporte à aprendizagem colaborativa tornaram-se amplamente beneficiados pela Internet. O seu propósito básico é permitir que os aprendizes desenvolvam suas atividades através de uma rede de computadores. Tais ambientes têm se mostrado significativos, tanto em aspectos cognitivos como em aspectos sociais [Menezes 2000].

Nesse trabalho apresentamos uma abordagem que visa propiciar um melhor desenvolvimento dos ambientes CSCL, de um lado pelo reuso de provedores de serviços (*web services*) implementados pelo framework e, por outro lado, os ambientes construídos conforme a arquitetura proposta podem fazer reuso no nível das aplicações. Isto tem um significado que julgamos importante: Uma aplicação criada para um ambiente pode ser utilizada em outros ambientes, apenas apontando para um *link.* O protocolo de notificação inter-aplicações se encarregará de manter a aplicação-mãe com todas as informações sobre as atividades do seu usuário, em qualquer lugar para onde ele se desloque na rede de aplicações do FAmCorA. Essa estratégia é fundamental para o surgimento de novas abordagens para ambientes educacionais, simplesmente porque agora podemos construí-los facilmente através de composição de aplicações desenvolvidas em diferentes ambientes, sob diferentes concepções pedagógicas.

A perspectiva atual da nossa pesquisa está voltada para a identificação dos elementos essenciais ao acompanhamento das interações do usuário pelas várias aplicações, o que nos permitirá a construção de uma ontologia. Tal ontologia será parte do protocolo de notificação entre aplicações, podendo assim modelar a comunicação de uma forma mais abrangente.

7. Referências

[Beck 1996] Beck,Kent. *Smalltalk Best Practice Patterns*. Prentice Hall -1996.

[BENTLEY 1997] BENTLEY R., APPELT W. , BUSBACH U.; HINRICHS E.;KERR D.; SIKKEL K.; TREVOR J, and WOETZEL, G. *Basic support for cooperative work on the World Wide Web*. Int. J. Human Computer Studies (1997).

[BROWN 1996] Brown, John Seely Brown and Dugui, Paul . *The Social Life of Documents.* www.firstmonday.dk/issues/issue1/documents

[Buschmann 1996] F. Buschmann, R. Meunier, H. Rohnert, P.Sommerlad, M. Stal. *Pattern-Oriented Software Architecture - A system of Patterns* John Wiley and Sons Ltd, Chichester, UK.

[Cadiz 2000] Cadiz, J.J. and Grudin, Jonathan *Using Web Annotations for Asynchronous Collaboration Around Documents* www.research.microsoft.com/research/coet/Annotations/CSCW2000

[CGI] www.w3.org/CGI/

[Chabert 1998] Chabert, A., Grossman, E., Jackson, L., Pietrowicz, S., and Seguin, C., *Java Object-Sharing in Habanero*. Communications of the ACM, Vol. 41 # 6, June 1998.

[Clarke 2001] Clarke, Dave and Dix, Alan. *Interfaces for the Active Web (Part 2)* Interacting with Computers Volume 13, Issue 6, August 2001, Pages 627-629

[Cook] Cook, John. *The Role of Dialogue in Computer-Based Learning and Observing Learning: an evolutionary approach to theory* Journal of Interactive Media in Education http://www-jime.open.ac.uk/

[Crow 1997] Crow, David; Parsowith, Sara; Bowden, and Wise, G. Bowden. *The Evolution of CSCW - Past, Present and Future Developments*. Acm sigchi Vol.29 No.2, April 1997.

[DALGARNO 2001] DALGARNO, BARNEY. *Technologies Supporting Highly Interactive Learning Resources on the Web: An Analysis*. Journal of Interactive Learning Research - Summer-Fall/2001

[DOM] www.w3.org/DOM

[Farias 2000] Farias C. R. G.; Pires, L. F.; Sinderen, M. *A Component-Based Groupware Development Methodology* .In Proceedings of the 4th International Enterprise Distributed Object Computing Conference (EDOC), Japan, 2000.

[Faruqui 2002] Faruqui, Masood. *SOAP raises the bar for CORBA*, www.borland.com/webservices

[Gambhir 2001] Gambhir, Sahil; Muchmore,Michael W.; *Web services: Revolution in the making*. PC Magazine, November/2001.

[Gay 1999] Gay, Geri; Sturgill, Amanda; Martin, Wendy. *Document-centered Peer Collaborations: An Exploration of the Educational Uses of Networked Communication Technologies.* Journal of Computer-Mediated Communication, 03/1999.

[George] George, Sébastien Leroux,Pascal. *Project-Based Learning as a Basis for a CSCL nvironment: An Example in Educational Robotics*. http://www-ic2.univ-lemans.fr/~george

[Glass 2000] Glass, Graham. *Applying Web services to applications .* The Mind Electric, 2000 www-106.ibm.com/developerworks

[Goulart 2001] Goulart, Alexandre M. A.; Pessoa, J. M Pessoa; Mensezes, C. S. *Um Ambiente Cooperativo para Apoio à Revisão Bibliográfica*. In Simpoósi Brasileiro de Informática na Educação(SBIE), Vitória/Brasil, 2001.

[Gunzer 2002] Gunzer, Hartwig. *Introduction to Web Services*, www.borland.com/webservices

[Guzdial] Guzdial, Mark. *Information Ecology of Collaborations in Educational Settings: Influence of Tool* http://guzdial.cc.gatech.edu/papers/infoecol

[Hiltz 1998] Hiltz, Starr Roxanne. *Collaborative Learning in Asynchronous Learning Networks: Building Learning Communities* in "WEB98" Orlando Florida, November 1998.

[Hooper 2000] Hooper, S. and Hokanson B. *Computers as cognitive media: examining the potential of computers in education* Computers in Human Behavior Volume 16 - 2000

[HTTP] www.w3.org/Protocols/HTTP/

[Jermann 2001] Jermann, Patrick; Soller, Amy; Muehlenbrock, Martin . *From Mirroring to Guiding: A Review of State of the Art Technology for Supporting Collaborative Learning* www.mmi.unimaas.nl/euro-cscl/Papers/197.pdf

[Jianhua] Jianhua, Zhao; Kedong,Li; Chong,Ng S. T. and AKAHORI, Kanji. *Peer Modeling and Its Application in Web-Based Intelligent Collaborative Learning Systems* www.icce2001.org/P04.html

[Keeble] Keeble R. J.; and Macredie, R. D. *Assistant agents for the world wide web intelligent interface design challenges* Interacting with Computers. Volume 12, 2000

[Koschmann 1996] Koschmann, T. (Ed.)(1996). *CSCL: Theory and practice of an emerging paradigm*. Mahwah, NJ: Lawrence Erlbaum

[KQML] www.cs.umbc.edu/kqml/

[Lehtinen] Lehtinen, Erno; Hakkarainen, Kai; Lipponen, Lasse; Rahikainen, Marjaana and Muukkonen, Hanni *Computer Supported Collaborative Learning: A Review*. www.comlab.hut.fi/opetus/205/etatehtava1.pdf

[LSI] lsi.research.telcordia.com/lsi/LSIpapers.html

[Maes94] Maes, Pattie. *Agents that Reduce Work and Information Overload*. In Communications of the ACM, vol. 37, Julho,1994.

[McGee 2002] McGee, Jeremy. *Simplifying Web Service development and integration* www.borland.com/webservices

[MCNAUGHT 2001] MCNAUGHT, CARMEL and AMORY, ALAN. *Collaboration, Design, and Technology: Themes in the Architecture of Interactive Learning Environments*. Journal of Interactive Learning Research - 2001.

[Menezes 2000] Menezes, Crediné S., Cury, Davidson, Tavares, Orivaldo L., Campos, Gilda H. B., Castro Jr., Alberto N. *An Architecture of an Environment for Cooperative Learning (AmCorA)*. ICECE - International Conference on Engineering and Computer Education, Brasil 2000.

[Navarro 2001] Navarro, Francisco;Keith, Jones; Gordhan,Sagar and Garnham,Nigel. *An Agent-Based Service Brokering Architecture for Multiservice Next-Generation Networks* http://magazine.fujitsu.com/us/vol37-1/paper13.pdf

[Newman 1996] Newman, D. R. *How can WWW-based groupware better support critical thinking in CSCL?* Proceedings of the ERCIM workshop on CSCW and the Web, , Germany, 1996, http://orgwis.gmd.de/projects/W4G/proceedings

[Pessoa 2000] Pessoa, J. M and Menezes, C. S. *QSabe II: A Cooperative Service for Knowledge Appropriation and Diffusion Using the Internet*. International Conference on Engineering and Computer Education-ICECE, Brasil, 2000.

[Rees 1999] Rees, Michael J.; Herring,Charles. *A Component-Based Groupware Architecture Model (COGAM)* http://comet.it.bond.edu.au/borg

[Rheinheimer 2001] Rheinheimer, Letícia Rafaela; Crespo C. S. Pinto, Sérgio. *JLearningServices: Um Framework para Serviços Síncronos em Ambientes* para EAD In: Simpósio Brasileiro de Informática Na Educação (SBIE); Vitória; Brasil, 2001.

[Roseman 1997] Roseman, M. and Greenberg, *Building Groupware with GroupKit*. In M. Harrison (Ed.) Tcl/Tk Tools, O'Reilly Press.

[Rowley 1997] Rowley, Kurt. *A design approach for the engineering and construction of CSCL-ITS environments.* In proceedings of the AI-ED 97 Workshop on Computer-supported collaborative learning environments and AI, Kobe, Japan, August 20, 1997.

[Santoro 1998] Santoro, F.M.; Borges, M.R.S.; Santos, N., *Um Framework para Estudo de Ambientes de Aprendizagem Cooperativa Apoiados por Computadores*, in Anais do Simpósio Brasileiro de Informática e Sociedade (SBIE), Fortaleza, Ceará, 1998.

[Stahl 2002] Stahl, Gerry *Contributions to a Theoretical Framework for CSCL* in CSCL-2002 http://www.cscl2002.org/proceedings.html

[Stanfill 1986] C. Stanfill and D. Waltz, *Toward Memory-based Reasoning*. In Communications of the ACM, vol. 29, December, 1986.

[Taiupathump 1999] Thaiupathump, Choonhapong *Intelligent Agents for Online Learning*. JALN Volume 3, Issue 2 - November 1999.

[Thiry 1998] Thiry, Marcello . *Intelligent Agent-Based Approach for Distance Learning* www.ime.eb.br/icee98/papers/212.pdf

[Watson 1996] Mark Watson. *Programming Intelligent Agents for the Internet*. Computing McGraw-Hill, 1996.

[WS] www.w3.org/2002/ws/

[WSDL] www.w3.org/TR/wsdl