Um Framework para Construção Cooperativa de Ambientes Virtuais de Aprendizagem na Web

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

[1]PPGEE – Universidade Federal do Espiríto Santo (UFES)
Av. Fernando Ferrari s/n – Goiabeiras – Vitória – ES – Brasil

[2]ICLMA– Universidade Federal de Mato Grosso (UFMT)
Rod. MT 100, Km 4, Pontal do Araguaia – MT– Brasil

jmpessoa@npd.ufes.br, credine@inf.ufes.br

Resumo. *O desenvolvimento de novos Ambientes 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. Neste trabalho apresentamos uma proposta de framework, para o desenvolvimento cooperativo de ambientes do tipo CSCL, baseado em padrões abertos, com ênfase no reuso de aplicações executáveis na Web, e não de código fonte, serviços (APIs) ou componentes dependentes de um ambiente de execução específico (middleware). O objetivo da proposta é possibilitar a construção de ambientes virtuais de aprendizagem nos mesmos moldes da composição de documentos do tipo hipertexto, em que o conhecimento expresso pela linguagem de formatação é automaticamente gerado por um ambiente visual de autoria.*

Palavras-chave: *Ambientes Virtuais, CSCW, CSCL, Web Framework, Famcora, Famcoralet, Composição Web.*

Abstract. *The development of new Computer-Supported Cooperative Learning (CSCL) environment demands such an amount of resources that make it very difficult to arise innovative proposals. As a consequence there is a huge gap between the theoretical conceptions and the actually implemented work. This paper presents a framework for developing CSCL applications also cooperatively based on Web open standards, emphasizing the reuse of ready-to-run applications and not on source code, services (APIs) or a middleware dependent binary components.The objective of this work is to ease the building of virtual environment for learning activities very similar to that of hypertext documents composition. This expressed knowledge built by the markup-language is automactically generated by the visual authoring environment.*

Key words: *Virtual Environment, CSCW, CSCL ,Web Framework, Famcora, Famcoralet, Web Application Composition.*

1. Introdução

A utilização da informática na educação, no Brasil, é tida como uma experiência promissora. Ao contrário de outros paises, onde a introdução dos computadores no ambiente ensino/aprendizagem se deu a partir de pressupostos puramente tecnológicos, a comunidade científica nacional tem reafirmado a primazia dos aspectos sócio-culturais-cognitivos sobre estes. Isto tem uma implicação importante: profissionais da educação, das mais variadas áreas de formação intelectual, participam ativamente da elaboração/construção/seleção dos artefatos de softwares que visam mediar a aplicação da informática na educação do Brasil, em particular na educação à distância.

A comunidade brasileira de pesquisadores da área de informática na educação participa desse esforço e tem feito importantes contribuições para o desenvolvimento de ambientes telemáticos de apoio ao ensino e aprendizagem, tanto em aspectos conceituais - pedagógicos quanto no desenvolvimento de ferramentas computacionais. Grande parte dessa contribuição pode ser observada em trabalhos apresentados nas

várias edições do Simpósio Brasileiro de Informática na Educação: ROODA [Behar 2001], Eureka [Eberspächer 1999], AulaNet [Crespo 1998], AVA [Crespo 2002], AmCorA [Menezes 2002].

Entretanto, ainda que o número de ambientes propostos seja significativo, a realidade qualitativa aponta que típicos ambientes telemáticos de apoio à aprendizagem na Web, relatados, dão suporte a um conjunto padrão de funcionalidades: documentos em hipertexto, áreas específicas para compartilhamento de arquivos (upload/download), distribuição de mensagens (email), conversação síncrona (chat), fóruns de discussão, murais (quadro de avisos), serviço de notificação de novidades, notificação de presença para troca de mensagens instantâneas (IM) e serviço de esclarecimento de dúvidas (FAQ) [Crow 1997], [Jermann 2001], [Santoro 1998].

Apesar da nítida intersecção do conjunto básico de aplicações/ferramentas utilizadas nos sistemas CSCL relatados, devido a questões tecnológicas e do escopo do domínio de problema, a construção de um ambiente novo quase sempre requer a re-implementação de cada uma dessas ferramentas. Paralelamente, existem muitas outras funcionalidades que são originais a cada sistema, mas que não são passiveis de reaproveitamento devido a questões tecnológicas e de concepção. Esse cenário apresenta algumas sinalizações importantes: primeiro indica que muito do esforço da comunidade é despendido "reinventando a roda". Segundo indica que a satisfação de uma determinada abordagem pedagógica não pode ser alcançada por nenhum sistema isoladamente, acarretando a invenção de um novo sistema e alimentando o círculo vicioso.

Este trabalho propõe: 1) um padrão de arquitetura para o desenvolvimento cooperativo de ambientes do tipo CSCL na Web, com ênfase no reuso e na interoperabilidade de aplicações executáveis, capaz de suportar dinamicamente a conexão entre aplicações nos mesmos moldes da Web, isto é, pela simples composição de *hyperlinks*; 2) um *site* para catalogar as aplicações desenvolvidas segundo o *framework* proposto e para dar suporte a autoria e hospedagem de ambientes virtuais de aprendizagem, ao alcance de pesquisadores e professores das mais diversas áreas de conhecimento, não necessariamente *experts* em programação de computadores.

O texto está organização da seguinte maneira: a seção 2 apresenta as abordagens típicas para reuso em CSCL/CSCW; a seção 3 apresenta uma proposta de framework, para o desenvolvimento cooperativo de ambientes do tipo CSCL/CSCW na Web; a seção 4 descreve o *site* da Fábrica de Ambientes de Apoio à Aprendizagem e a seção 5 é apresenta as considerações finais.

2. Abordagens para reuso em CSCL/CSCW

As tecnologias para o desenvolvimento de aplicações do tipo CSCW/CSCL com ênfase no reuso têm apontado basicamente em três direções: componentes (dependentes de um middleware), bibliotecas de classes e composição de serviços (Web services) [Farias 2000], [Rees], [Roseman 1996], [Chabert 1998], [Gambhir 2001].

Segundo [Jia 2002] em desenvolvimento baseado em componentes (DBC) os requisitos do sistema (no domínio do problema) são mapeados diretamente para componentes de terceiros (no domínio da solução), reduzindo assim algumas fases tradicionais de desenvolvimento para aumentar a eficiência e qualidade do produto final. A figura 1 apresenta a principais ênfases do DBC.

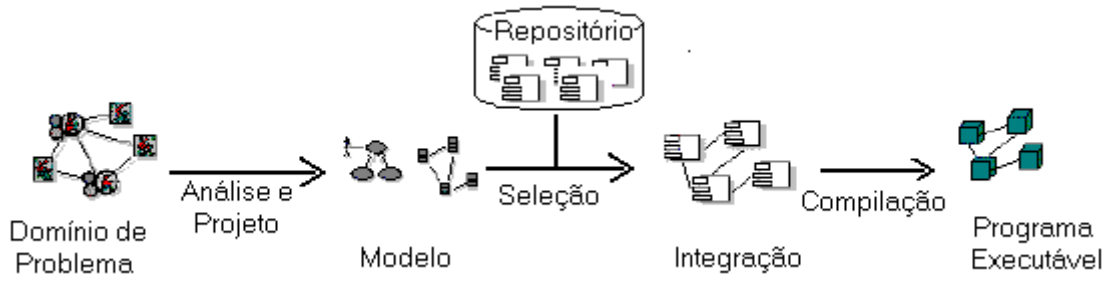

Figura 1. O processo de desenvolvimento baseado em componentes.

Os seguintes processos são particularmente enfatizados no desenvolvimento baseado em componentes: **Análise**: 1) captura os requisitos de sistema e define os seus limites; 2) define a arquitetura de sistema

que permita a colaboração entre componentes; 3) define os requisitos dos componentes. **Seleção**: A seleção é também uma busca pela mais alta coesão (*highly cohesive*) e pelo menor grau de acoplamento (*lowly coupled*), isto porque as dependências precisam ser minimizadas, de modo que falhas ou mudanças, fiquem localizadas, produzindo o mínimo de impacto possível no sistema (*robust*). **Integração**: A integração envolve a composição e adaptação dos componentes selecionados e a codificação do chamado *glue code*.

Na prática, a implementação de uma aplicação a partir de um framework orientado a objetos e/ou serviços, requer um significativo esforço técnico, pois a integração entre os componentes ocorre no nível de programação (*glue code*). Essa programação inclui e requer o mapeamento dos protocolos de comunicação na linguagem de implementação, conhecimento avançado de APIs e possivelmente a existência de algum tipo de *middleware* (EJB, CORBA, COM, etc.). Um exemplo típico dessa abordagem para construção de ambientes do tipo CSCL/CSCW é o framework Habanero [Chabert 1998]: sua instalação requer versões específicas de múltiplos pacotes de software (Jigsaw, Jini, object databases, etc.) e uma extensão/adaptação/composição requer a escrita de código Java, o chamado *glue- code*.

A arquitetura da Web sugere uma outra possibilidade: a composição de *hyperlinks* pode dar origem a um aplicação "portal". A Web faz composição de aplicações e não de serviços sinalizando uma orientação inovadora. A composição utiliza apenas o conhecimento declarativo (HTML) e não requer qualquer tipo de *glue-code*.

Entretanto, em situações reais, as tecnologias e paradigmas utilizadas no desenvolvimento de aplicações Web correntemente e a complexidade e necessidades do domínio (modelo de utilização e modelo de tarefa) dos ambientes de apoio ao trabalho cooperativo (CSCW), em particular daqueles voltados para o ensino e aprendizagem (CSCL), impossibilitam a construção de novos ambientes com a simples reutilização baseada no modelo de hipertexto da Web.

3. Em direção a um framework para o desenvolvimento cooperativo de ambientes do tipo CSCL/CSCW na Web.

Segundo [Krueger 1992] uma tecnologia para construção de software baseada em reuso pode reduzir a distância cognitiva (esforço intelectual): (1) reduzindo a distância entre as fases de concepção e especificação e (2) reduzindo a distância entre as fases de especificação e a produção do sistema executável.

Uma paráfrase dessa constatação é que uma tecnologia de construção de software reduz o esforço cognitivo (no sentido de conhecimento especializado) quanto aproxima o integrador de componentes (construtor de aplicações) ao usuário não especializado (autor de documentos). Isto é, quando o modelo de tarefa "programação de aplicação" se aproxima do modelo de tarefa "autoria de documentos", ou mais diretamente quando uma aplicação se aproxima de um documento, conforme figura 2.

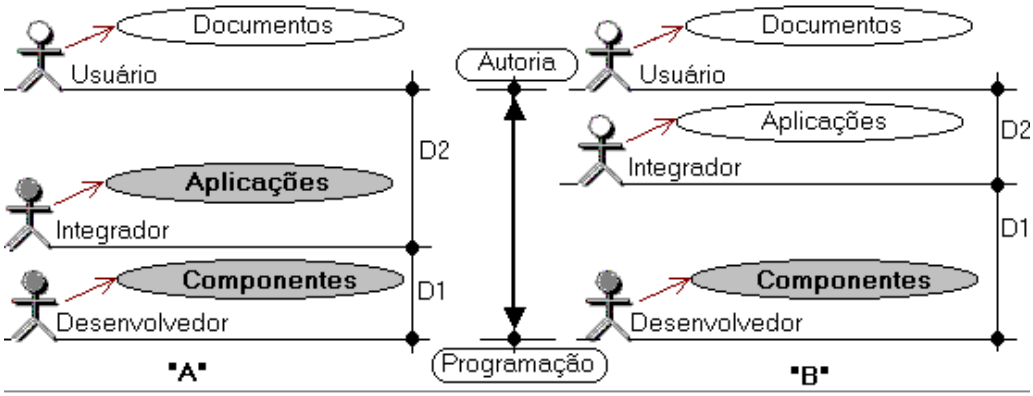

Figura 2. Programação X Autoria

A figura 2 faz apelo para a noção intuitiva da distância, em termos de conhecimento especializado, para produzir três tipos de produtos de software: componente, aplicação e documento. O lado esquerdo (A) da figura 2 apresenta um cenário em que o integrador (construtor de aplicações) está próximo do

programador (desenvolvedor de componentes), em conseqüência o produto aplicação está mais próximo do produto componente e mais distante do produto documento. O lado direito (B) da figura 2, apresenta uma situação diferente: o produto aplicação se aproxima do produto documento e em conseqüência um integrador (construtor de aplicações) está bem mais próximo do usuário não especializado (autor de documentos).

O lado esquerdo (A) da figura 2 apresenta uma visão coincidente com os atuais paradigmas orientados a componentes em que a construção de aplicações (do mesmo modo que os componentes), requer um esforço (conhecimento especializado) de "programação". O lado direito (B) da figura 2 apresenta um paradigma coincidente com aquele utilizado pela Web. A construção de aplicações requer apenas a "composição" em uma linguagem de formatação (HTML/XML), nos mesmos moldes dos documentos. No contexto do desenvolvimento baseado em componentes, como apresentado na figura 1 da seção 2, o processo fica mais reduzido com a eliminação das fases de integração e compilação. Isto implica que a fase de seleção dos componentes deve ser suficiente para produzir uma aplicação executável. De fato, essa constatação serve de motivação para a proposta apresentada aqui neste trabalho.

3.1. Visão Conceitual

A figura 3 dá uma visão conceitual das principais atividades relacionadas à construção e utilização de ambientes do tipo CSCL no escopo desse artigo. Cada camada representa diferentes papéis de atuação. Os três atores e seus respectivos "casos de uso": comunidade (utilização do ambientes), autor (autoria de ambientes) e desenvolvedor (registro de aplicações/componentes), possuem cada um deles o seu próprio modelo de tarefa: modelo de utilização, modelo pedagógico e modelo de desenvolvimento.

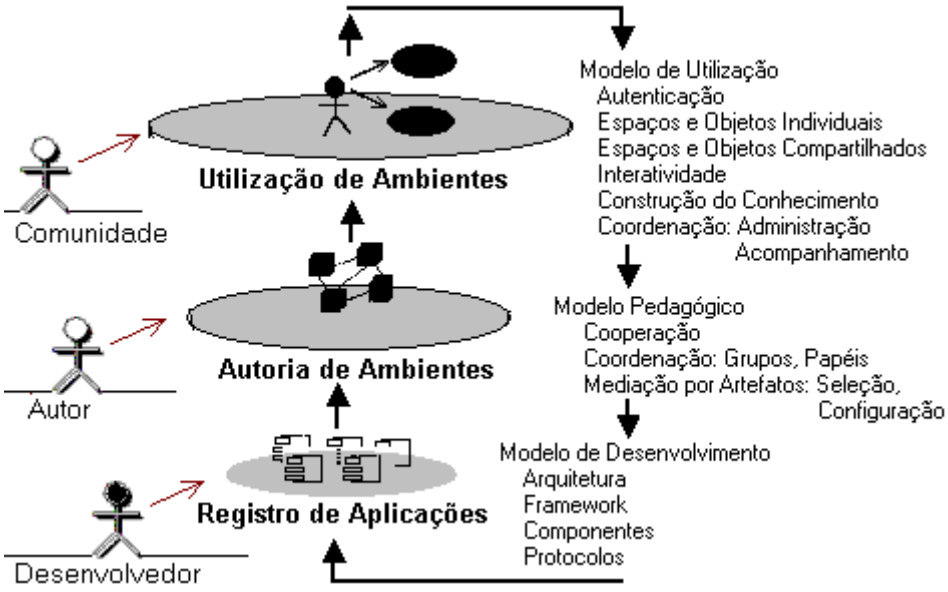

Figura 3. Visão Conceitual: Atores e Modelos de Tarefas.

Na figura 3 um "desenvolvedor" é um especialista em programação, produtor de aplicações básicas, cujo modelo de desenvolvimento requer uma arquitetura que inclui um modelo de componentes e protocolos de comunicação e composição entre aplicações, isto é um framework. O "autor" é um típico profissional do ensino que possui como modelo de tarefa a aprendizagem cooperativa e utiliza o framework proposto como ponto de partida para a autoria de ambientes de aprendizagem segundo as suas necessidades pedagógicas. Na camada superior, o ator "comunidade" representa o conjunto dos usuários que possuem um modelo de utilização comum aos ambientes virtuais de aprendizagem. As setas indicam o caminho percorrido pelos requisitos. Por exemplo, uma aplicação registrada pelo "desenvolvedor" deve atender ao "Modelo de Desenvolvimento" especificado pelo framework que por sua vez deve expressar aspectos do domínio do "Modelo Pedagógico" capturados pelos casos de uso do "Modelo de Utilização".

O objetivo deste trabalho é propor um modelo de desenvolvimento cooperativo que na primeira camada potencializa os esforços dos desenvolvedores (programadores) e na segunda camada torna a construção de ambientes virtuais que atende o modelo de aprendizagem cooperativa uma atividade de autoria.

3.2. Aspectos dos requisitos dos CSCL/CSCW

A Web permite-nos pensar em um *framework* orientado a aplicações, indo além do paradigma orientado a reuso de código, *middleware* ou serviços. Porém, assim como na modelagem orientada a objetos, permanece a complexidade em relação a partição do domínio. Uma questão típica é como lidar com aspectos não funcionais de comunicação, cooperação, e coordenação, em geral embutidos em uma estrutura/arquitetura de componentes. Freqüentemente observa-se que algumas propriedades do sistema "atravessam" (*cross cuting*) em direção a vários componentes. Em *AOP* (*Aspect Oriented Programming*) tais propriedades são chamadas de "aspectos" [Grundy 1999]. Em ambientes CSCL/CSCW os seguintes aspectos podem ser observados:

Portal: o termo "portal" designa um ponto compreensível de acesso à informação (categorização e busca), aplicações (desktop integrado) e pessoas (colaboração) em uma comunidade virtual na Web. **Grupo**: os seguintes termos também são entendidos com o mesmo significado: projeto, turma, disciplina, evento, etc. **Papel**: o conjunto de permissões de um indivíduo em relação ao sistema e que corresponde à sua responsabilidade na organização. **Autenticação**: procedimento de identificação padrão em ambientes multi-usuários (Aspectos relacionados a organização de comunidades virtuais).

Notificação: visa manter as aplicações e seus usuários cientes de fatos, preferências, acontecimentos e eventos que transcorrem no ambiente. **Log**: [Stahl 2002] argumenta que é fundamental que os ambientes virtuais de aprendizagem guardem o histórico das atividades de cada indivíduo para monitoramento das interações, coordenação e análises. **Composição**: um dos objetivos da proposta aqui relatada é que uma aplicação desenvolvida para um ambiente/portal possa ser utilizada (*plug-and-play*) em um outro, de modo transparente, apenas apontando para um *hyperlink*. A composição diz respeito ao acoplamento entre uma aplicação e outra aplicação e ainda entre a aplicação e o framework (Aspectos relacionados a interoperabilidade entre aplicações).

3.3 Arquitetura da Fábrica de Ambientes

A figura 4 salienta os módulos de colaboração da comunidade de desenvolvedores (implementadores e registradores de componentes) e autores (criadores e administradores de portais).

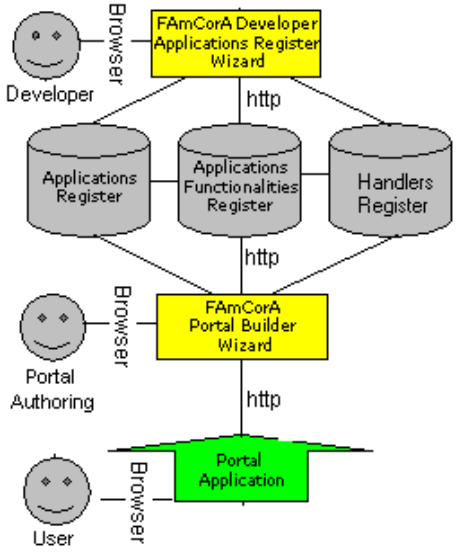

Figura 4. Arquitetura da Fábrica.

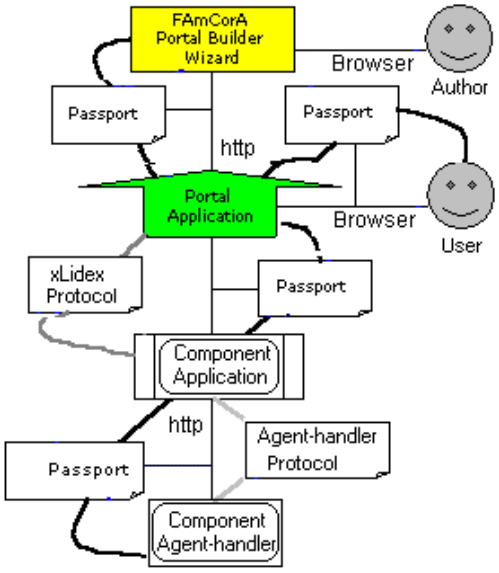

Figura 5. Arquitetura de Comunicação

Um desenvolvedor utiliza o módulo Developer Applications Register Wizard (gerenciador do catálogo) para: 1) Registrar Aplicações; 2) Registrar Funcionalidades; 3) Registrar *Handlers* (tratadores de

funcionalidades). Um autor utiliza o módulo *Portal Builder Wizard* para a geração (autoria) de um ambiente virtual de apoio à aprendizagem cooperativa. O módulo *Portal Application* é o resultado da atividade de autoria. Isto é, uma aplicação do tipo CSCL/CSCW pronta para ser utilizada.

3.4 Arquitetura do framework

Segundo [Garlan 2000], o reuso é uma característica de qualidade agregada à definição de uma arquitetura. Um dos objetivos da arquitetura é promover a reutilização. Uma arquitetura de software envolve um modelo de componentes/aplicações; os padrões que guiam a composição e interação entre componentes (protocolos) e as restrições nesses padrões (estilo arquitetônico).

A figura 5 apresenta a arquitetura de comunicação entre os módulos. O módulo *Portal Builder Wizard* é o responsável pelo catálogo de portais (portal aspecto). O módulo *Portal Application* é o responsável pelo catálogo de grupos (aspecto grupo), usuários (aspecto autenticação) e papéis (aspecto papel). Todos esses aspectos, somados a aqueles relativos à interoperabilidade (composição) entre aplicações são resolvidos no nível da arquitetura de componentes e do protocolo *Passport*. O *log* das atividades (aspecto *log*) é mediado pelo protocolo *xLidex* (*Extensible Learning Interactions Data Exchange*). A interação entre uma aplicação e um agent-handler (aspecto notificação) é mediada por um protocolo específico (protocolo Agent-handler).

O Protocolo *Passport* é o mediador básico da arquitetura do framework, principalmente para a modelagem não funcional: organização, coordenação, autenticação (portal, grupo, usuário), configuração e composição (reuso). O *passport* do portal é a sua configuração básica. As configurações particulares para cada um dos seus usuários são mantidas pelo *passport* do indivíduo, isto é, cada usuário recebe o seu próprio *passport* ao fazer "*login*" em um portal.

3.4.1 Modelo de Componentes

Os componentes, aqui chamados de *famcoralets,* são os módulos básicos desenvolvidos por colaboradores especialistas em programação e são registrados em um repositório (catálogo) do framework. Um componente publica métodos, propriedades, eventos e receptáculos, conforme figura 6. Uma publicação é um compromisso de configuração. O termo receptáculo está relacionado com a composição da interface gráfica (GUI). Isto é, um componente (*famcoralet*) que publica um receptáculo passa a ser também um contêiner GUI.

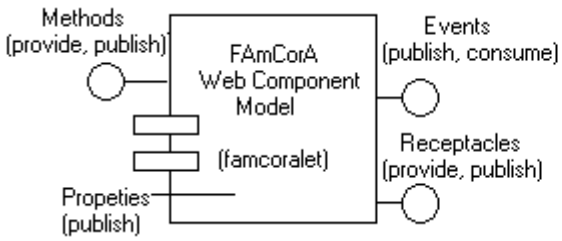

Figura 6. Modelo de Componente.

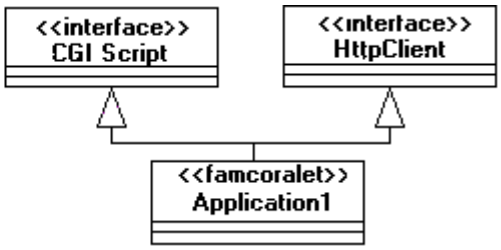

Figura 7. Classe de Aplicação.

Na proposta aqui relatada um componente é qualquer aplicação cujo front-end é um cliente HTTP (o browser ou outro componente), podendo ser implementado em qualquer tecnologia de geração dinâmica de páginas (scripts) em conformidade com o protocolo CGI (exemplos: cgi-bin, ASP, ASP.NET, Java servelet, PHP, ISAPI DLL, Apache DSO, etc.) com as seguintes responsabilidades: participar do processo de registro ao framework e estabelecer uma comunicação com outras aplicações conforme os protocolos estabelecidos pelo framework. O diagrama de classe da figura 7 mostra a hierarquia geral das classes de aplicação/componente. A figura 7 mostra que um componente/aplicação (famcoralet) compatível com o framework é um tipo de aplicação *script* CGI que implementa as interfaces de um cliente HTTP.

3.4.2 Arquitetura de Composição

A composição de componentes (seja de serviços ou GUI), segue o paradigma da autoria. O framework define um protocolo baseado em XML, chamado Passport, como mediador básico dos aspectos não

funcionais, tais como: organização, coordenação, autenticação (portal, grupo, usuário), configuração e composição e propõe uma arquitetura no estilo *blackboard* para mediar a comunicação entre componentes. Esta decisão não restringe a comunicação direta entre as aplicações (como acontece no estilo *Broker*) e tem entre os seus objetivos: facilitar o desenvolvimento de componentes, evitando que cada desenvolvedor tenha que implementar um parser/Interpretador para o protocolo, mediar a interação e composição de componentes e ainda certificação, evitando chamadas não autorizadas entre aplicações e o chamado "uso malicioso". A figura 8 ilustra a arquitetura de composição do framework centrada no *Passport Provider* que é o guardião do *passport*.

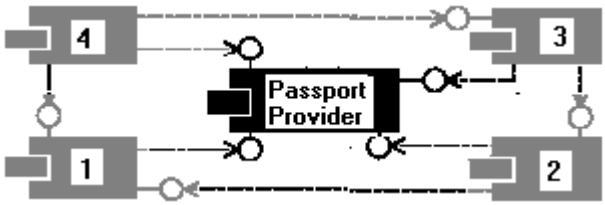

Figura 8. Arquitetura de composição.

Um *passport* descreve aspectos não funcionais oferecidos pelo framework, tais como o portal e o grupo que originou a chamada, a autenticação do usuário naquela chamada, etc. Na prática, algumas dessas informações, por exemplo, o *host* provedor do *passport* e a referência GUID do *passport*, precisam ser antecipadas, cabendo então ao componente que solicita um serviço de um outro passar esses parâmetros na mensagem de chamada GET do http.

De fato existe uma ontologia para essa requisição. O pseudocódigo da figura 10 exemplifica essa situação para o caso hipotético em que o Componente "1" chama o Componente "2".

```
Component1.httpRedirect("http://component2host/component2/init?
                         portal=FAmCorA&
                         group=system&
                         passporthost=http://200.137.66.74&
                         GUID=P17032003171414")
```

Figura 10. Ontologia de uma requisição entre componentes.

3.4.3 Protocolo xLidex

O ***Ex****tensible* ***L****earning* ***I****nteractions* ***D****ata* ***Ex****change* (*xLidex*) é um protocolo baseado em XML para a troca de dados relativa as atividades do usuário (*log*) entre aplicações, oferecendo mecanismos para a coordenação e o monitoramento do trabalho individual e do grupo. O *xLidex* é extensível e aberto, permitindo retratar/reportar qualquer tipo de atividade proposta em uma ferramenta. As figura 11 e 12 retratam trechos da sintaxe do protocolo para um solicitação de dados e uma resposta a essa solicitação

```
<xLide communication-id="GUID_01">
  <head>
    <datetime>.....<datetime>
    <from-agent>.....</from-agent >
    <about communication-id="">
    </about>
  </head>
<body  type="request-user-interaction">

    <request-user-interaction>
        <portal>.....</portal>
        <group>..... </group>
        <email>.. .. ...</email>
        <init-date>.....</init-date>
        <end-date>.....</end-date>
  </request-user-interaction>

</body>
</xLidex>
```

Figura 11. xLidex Solicitação

```
<xLidex communication-id="GUID_02">
  <head>
     <datetime >.....</datetime>
     <from-agent>.....</from-agent>
     <about communication-id="GUID_01">
     </about>
  </head>
  <body type="response-user-interaction">

       <response-user-interaction>
       <report  type="document-handling">
           <document-handling>
             <event  datetime="..... ">
             </event>
             <content>.....</content>
           </document-handling>
       </report>
    </response-user-interaction>

  </body>
</xLidex>
```

Figura 12. xLidex Resposta

Em *xLidex* um repórter pode ser de vários tipos: dependentes apenas de uma ontologia para <content> e <event>.

3.4.4 Protocolo Agent-handler

O protocolo Agent-handler governa a comunicacão/interação entre uma aplicação e um agent-handler em tempo de execução. Ao se acoplar a uma aplicação um agent-handler se anuncia como sendo do tipo *redirect* ou *interactive*. Um agente do tipo *redirect* desvia a execução de uma funcionalidade da aplicação para outra localidade, sem nada retornar. Por outro lado, um agente do tipo *interactive* retorna uma das seguintes diretivas: *redirect* (em que o agente pede que a aplicação seja redirecionada para outra localidade), *response* (em que o agente formata a informação que deve ser retornada ao usuário da aplicação) ou *continue* (em que o agente solicita que a aplicação siga o curso normal do processamento). Quando invocado, o agent-handler recebe as mesmas informações da aplicação, podendo, inclusive, modificá-las.

4. O Portal da Fábrica de Ambientes (FAmCorA)

O *site* do FAmCorA materializa as idéias apresentada neste trabalho tanto no sentido de oferecer um suporte aos progamadores fazendo o catálogo da aplicações quanto por oferecer um ambiente visual de autoria para a construção de portais de apoio a aprendizagem na Web no mais genuíno estilo *click and play* voltado para usuários não especialistas.

O portal FAmCorA, como qualquer portal gerado pelo framework proposto, pode conter qualquer quantidade de grupos (que podem conter subgrupos). Um grupo/subgrupo pode conter qualquer quantidade de papéis. Todo usuário inscrito em um grupo tem um papel. Para cada papel pode ser disponibilizado um conjunto de ferramentas (famcoralets), e cada ferramenta pode ser configurada, de modo transparente (*click and play*!), para operar segundo um papel. Um portal criado com o framework do FAmCorA é sempre uma obra "em construção". A qualquer momento o autor pode cria novos grupos e papéis, configurar e reconfigurar as ferramentas, sem nenhum impacto sobre o ambiente, porque o trabalho de autoria é uma "simples" geração de conhecimento declarativo, isto é, uma configuração de (re)uso.

4.1 Ambiente do desenvolvedor

Um desenvolvedor é um colaborador do FAmCorA que desenvolve aplicações (*famcoralets*) e está inscrito no grupo "Developers". A figura 13 mostra que o desenvolvedor utiliza uma ferramenta de nome Register (gerenciador das "páginas amarelas" ou catálogo) para: 1) Registrar Aplicação; 2) Registrar Funcionalidade; 3) Registrar Handler; 4) Registrar Temas. A figura 13 apresenta ainda a atividade de registrar um handler (um handler é um termo genérico no FAmCorA, podendo significar deste um tratador de eventos até mesmo um valor atribuído a uma propriedade). O desenvolvedor escolhe a

aplicação "NewsBoard" (lista de seleção) e pode assim, para cada funcionalidade da aplicação, ou seja, "New" (*method*), "OnNew" (*event*), SPONSOR_1 (*receptacle*) e "BackGround" (*property*) propor um handler. No exemplo, o desenvolvedor registra no catálogo um agent-handler, do tipo "*interactive*", nomeado "*NotifyUser*", que pode ser utilizado para interceptar o evento "OnNew" da aplicação. O *handler* nesse caso é um serviço (script CGI).

Figura 13. Ambiente do Desenvolvedor.

Figura 14. Ambiente de Autoria.

4.2 Ambiente de Autoria de Portais

Segundo [Garlan 2000], com o rápido crescimento do Internet, muitos usuários estão em condições de "recortar" e "colar" serviços. Tais usuários mesmo que possuindo uma perícia técnica limitada, ainda assim exigirão garantias de que as partes coladas irão funcionar corretamente, segundo as suas expectativas.

A autoria de um portal educacional com o framework FAmCorA e assistida por um *wizard*, seguindo o paradigma RAD. Um *click* de *mouse* gera o núcleo básico do sistema, prontamente instalado e funcionando. Isto é, um Web portal. Ao logar no portal, o autor do recebe a ferramenta básica de adminstração e configuração (*GroupManager*) para: 1) criar grupos e os seus papéis, 2) incluir novos usuários em um grupo. 3) escolher e configurar as ferramentas que estarão à disposição dos usuários, em função do papel assumido por cada um deles dentro de um grupo. Na figura 14 o autor escolhe no catálogo a ferramenta "NewsBoard" e a paleta de eventos para configura um agent-handler (*NotifyUser*) como ouvinte (*listener*) do evento "OnNew" para todo usuário do grupo inscrito com o papel "Participant".

5. Considerações Finais

O framework aqui relatado está baseado em padrões da Web, com ênfase no reuso de aplicações executáveis, e não de código fonte (módulos, bibliotecas de classes, etc) ou componentes binários, dependentes de um midleware (EJB, CORBA, .Net, etc.). Isto tem um significado que julgamos importante: uma aplicação desenvolvida e catalogada no *framework* pode ser utilizada em vários ambientes, apenas apontando para um *hyperlink*. Além disso, a proposta possibilita a construção de ambientes virtuais de aprendizagem nos mesmos moldes da composição de documentos, seja porque o conhecimento fica expresso em uma linguagem de formatação (XML), seja pela mediação do ambiente visual de autoria.

Na seqüência da pesquisa estamos trabalhando para disponibilizar o *site* do FAmCorA para todo a comunidade de pesquisadores em informática na educação e esperamos com isso contribuir para congregar os nossos esforços de desenvolvimento.

6. Referências

Behar, P.; Kist, S.O.; Bittencourt, J. V.. A caminho de um Ambiente para a Educação à Distância: ROODA Rede cOOperativa De Aprendizagem. XII SBIE, Vitória/ES, 2001.

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

Crespo S. P.; M. F. Fontoura, Lucena C. J. Construção de Cursos Baseados na Web Usando o Ambiente AulaNet. IX SBIE, Fortaleza/CE, 1998.

Crespo, S. P.; Schlemmer E.; Santos C. T.; Cláudia C.; Rheinheimer P. L. AVA: Um Ambiente Virtual Baseado em Comunidades. XIII SBIE, São Leopoldo/RS, 2002

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.

Eberspächer, H. F.; Vasconcelos, C. D.; Jamur, J. H.; Eleuterio, M. A. Eureka: um ambiente de aprendizagem cooperativa baseado na Web para Educação à Distância. X SBIE, Curitiba/PR, 1999.

Farias C. R. G.; Pires, L. F.; Sinderen, M. A. Component-Based Groupware Development Methodology. In 4th International Enterprise Distributed Object Computing Conference, Japan, 2000.

Gambhir, S.; Muchmore, M. W.; Web services: Revolution in the making. PC Magazine, November/2001.

Garlan, David. Software Architecture: a Roadmap. The Future of Software Engineering, A. Finkelstein, ed., ACM Press, 2000.

Grundy, J., Aspect-oriented Requirements Engineering for Component- Based Software Systems, 4th IEEE International Symposium on Requirements Engineering, 1999.

Jermann, Patrick; Soller, Amy; Muehlenbrock, Martin . From Mirroring to Guiding: A Review of State of the Art Technology for Supporting Collaborative Learning. In Euro-CSCL 2001.

Jia, Yuqing Gu Yu. The Representation of Component Semantics: A Feature-Oriented Approach. 9th IEEE Conference and Workshops on Engineering of Computer-Based Systems, Sweden, April, 2002.

Krueger, Charles W.; Software Reuse. ACM Computing Surveys, Vol. 24, No. 2, June 1992

Menezes, C.; Pessoa, J.M., Netto, H. V.; et all; Educação a distância no Ensino Superior - Uma proposta baseada em Comunidades de Aprendizagem, XIII SBIE, São Leopoldo/RS, 2002

Rees, Michael J.; Herring,Charles. A Component-BasedGroupware Architecture Model (COGAM) http://comet.it.bond.edu.au/borg

Roseman, M., Greenberg, S. Building Real-Time Groupware with GroupKit, A Groupware Toolkit. ACM Transactions on Computer-Human Interaction 3, (1), 1996.

Santoro, F.M.; Borges, M.R.S.; Santos, N., Um Framework para Estudo de Ambientes de Aprendizagem Cooperativa Apoiados por Computadores. IX SBIE, Fortaleza, Ceará, 1998.

[Stahl 2002] Stahl, Gerry. Contributions to a Theoretical Framework for CSCL. In Euro CSCL 2002, http://orgwis.gmd.de/~gerry/publications/conferences/2002/cscl2002/index.html