Universidade Federal do Amazonas
Instituto de Computação
Programa de Pós-Graduação em Informática

Leonardo Nascimento dos Santos

Um Arcabouço para Desenvolvimento de Ambientes Virtuais Flexíveis e Especificáveis

Tese de Doutorado em Informática

Manaus/AM
2012

Leonardo Nascimento dos Santos

Um Arcabouço para Desenvolvimento de Ambientes Virtuais Flexíveis e Especificáveis

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Informática, como requisito parcial para a obtenção do título de Doutor em Informática, área de concentração Inteligência Artificial.

Universidade Federal do Amazonas
Instituto de Computação
Programa de Pós-Graduação em Informática

Área de Concentração: Inteligência Artificial

Orientador: Prof. Dr. Alberto N. de Castro Júnior
Co-orientador: Prof. Dr. Crediné S. de Menezes (Universidade Federal do Rio Grande do Sul – UFRGS)

Manaus/AM
2012

RESUMO

Nos últimos dez anos, a Web tem se transformado numa grande plataforma de interação entre seus usuários. No entanto, observa-se que de suas ferramentas não são suficientemente adequadas para dar suporte a essas interações. O desenvolvimento dessas ferramentas não tem conseguido acompanhar de forma adequada a necessidade por interação dessas pessoas. De forma que se torna necessário o desenvolvimento de ferramentas flexíveis, que permitam que a interação aconteça de acordo com as necessidades de seus utilizadores. Ou seja, as ferramentas web de suporte devem ser condicionadas pela natureza evolutiva das pessoas e devem refletir a evolução de suas necessidades, adaptando seus requisitos de usabilidade e sociabilidade. Várias abordagens de desenvolvimento de ambientes virtuais os tratam como produtos prontos, que podem esperar pelo ciclo tradicional de desenvolvimento de software, ou seja, se uma demanda por alterações surgir, ela deve voltar para o a mesa de projeto para que ele sofra as modificações cabíveis. Para se resolver esse problema têm-se concebido um paradigma inovador de construção de ambientes virtuais chamado MOrFEu. Nesse trabalho, apresenta-se uma extensão desse paradigma que é capaz de descrever detalhadamente cada um dos componentes de um ambiente virtual, e a partir de uma "programação declarativa", esse ambiente virtual é implementado. Nesse arcabouço, foi possível modelar/descrever 20 tipos de ambientes virtuais distintos. É apresentado também como esses ambientes são flexíveis, suscetíveis a alterações em tempo de execução, bastando serem feitas alterações em suas descrições. Dessa forma, é possível facilmente construir ou adaptar ambientes virtuais que atendam às demandas de interação de seus usuários.

Palavras-chave: Desenvolvimento de Ferramentas de Comunicação; Ambientes Virtuais; Flexibilidade.

Abstract

BLABLABLA.

Keywords: HELLO.

Lista de Figuras

Lista de Tabelas

Sumário

1 Introdução

Nos últimos dez anos, a Web tem se transformado numa grande plataforma de interação entre seus usuários, através de várias de suas ferramentas. No entanto, observa-se que algumas dessas ferramentas não são suficientemente adequadas para dar suporte a essas interações. O desenvolvimento dessas ferramentas não tem conseguido acompanhar de forma adequada a necessidade por interação através da Web desses grupos de pessoas. De forma que se torna necessário o desenvolvimento de ferramentas flexíveis, que permitam que a interação aconteça de acordo com as necessidades de seus utilizadores. Ou seja, **as ferramentas web de suporte à interação entre as pessoas devem ser condicionadas pela natureza evolutiva das pessoas e que devem refletir a evolução de suas necessidades, adaptando seus requisitos de usabilidade e sociabilidade** (PAREDES, 2007).

A Web tem se transformado de uma "web de leitura" para uma "web de leitura e escrita" (ULLRICH, BORAU, *et al.*, 2008), ou seja, tem ido ao encontra das ideias de seu criador: "um sistema no qual compartilhar o que você sabe ou pensa, é tão fácil quanto aprender o que outra pessoa sabe" (BERNERS-LEE, 2000, p. 33). Nesse caminho, seus usuários passaram a ser mais participantes nesse ciberespaço, interagindo com outras pessoas através das máquinas, o que fez surgir uma necessidade maior por aplicações com interação do tipo humano-computador-humano (PAREDES, 2007) (CLUBB, 2007) – trata-se de um tipo de interação entre as pessoas que utiliza os computadores e os sistemas de rede de computadores como mídia para viabilizar essa interação.

Aplicações web típicas de interação humano-computador-humano são os **Ambientes Virtuais**, que são sistemas web que reúnem virtualmente várias pessoas em um mesmo lugar possibilitando a interação entre elas. Eles também podem ser chamados de **Espaços Virtuais**, não havendo diferenciação de semântica nesse caso.

Fuks *et al.* (2007) afirmam que, mesmo que um desenvolvedor seja capaz de desenvolver uma aplicação "ótima" para a interação de um grupo de pessoas, eventualmente ela se tornará inadequada devido a novas situações e problemas que certamente aparecerão. Paredes (2007) acredita que os ambientes virtuais devem ser condicionados pela natureza evolutiva das pessoas e que devem refletir a evolução de suas necessidades, adaptando seus requisitos de usabilidade e sociabilidade.

Este problema é mais agudo quando se trata de aplicações para domínios específicos, como o domínio complexo da educação, que demanda ferramentas mais flexíveis que possuem natureza maleável. Como é o caso das arquiteturas pedagógicas, seu caráter é pensar a aprendizagem como um trabalho artesanal (CARVALHO, NEVADO e MENEZES, 2005), onde o estudante é o ator principal e o professor um estimulador. Deve-se pensar a aprendizagem como um processo de criação de novidades, de descobertas e invenções, permitindo que os sujeitos realizem experimentações, simulações, em busca de soluções para questões significativas do ponto de vista do sujeito. Essas atividades demandam ambientes que devem ser flexíveis, de forma a favorecer o protagonismo e a autoria individual e coletiva, oferecendo formas diferenciadas de organizar as interações e produções, tendo como referência espaços de autoria reorganizáveis e flexíveis (MENEZES, NEVADO, *et al.*, 2008).

Ou seja, o caráter "artesanal" das arquiteturas pedagógicas implica que as atividades serão diferenciadas para cada grupo de estudantes, para cada vez que as atividades de uma arquitetura pedagógica são realizadas. Obviamente, as arquiteturas pedagógicas propõem atividades não tradicionais, isso implica o surgimento de novas ferramentas para apoiar essas atividades, demandando flexibilidade para criação

dessas ferramentas. Além disso, isso demanda flexibilidade para modificação ou adaptação de ambientes virtuais existentes, tendo em vista que cada grupo tem sua própria maneira de realizar a atividade.

Na tentativa de solucionar o problema da falta de flexibilidade das aplicações de interação humano-computador-humano, desenvolvedores de ambientes virtuais têm apresentado técnicas flexíveis de desenvolvimento de novos ambientes virtuais. Entre elas, destacam-se aquelas que são baseados em módulos (DOUGIAMAS e TAYLOR, 2003), em componentes (GEROSA, PIMENTEL, *et al.*, 2006) (WON, STIEMERLING e WULF, 2006) (BEDER, SILVA, *et al.*, 2007), em linhas de produção de software (GADELHA, NUNES, *et al.*, 2009), em *web-services* (PESSOA, NETTO e MENEZES, 2002) (MEDEIROS, 2005) e em agentes (VASSILEVA, DETERS, *et al.*, 2001).

No entanto, essas abordagens não são capazes de tratar os tipos de modificações dinâmicas que os usuários estão demandando (NEVADO, CARVALHO e MENEZES, 2007). Todas essas abordagens tratam o ambiente virtual como uma aplicação concluída, que não precisarão de modificações depois de serem entregues para o uso. Consideram que o processo de mudanças nas necessidades dos usuários pode esperar pela reformulação das características do ambiente por parte dos desenvolvedores. Ou seja, eventualmente a aplicação irá se tornar inadequada, e quando isso acontecer será necessário reformulá-la, recomeçando do projeto. Assim, essas abordagens ainda não são flexíveis o suficiente para lidar com as demandas em domínios altamente dinâmicos, como é o caso do domínio da educação e da nova Web.

Além das abordagens já citadas, destaca-se na literatura um paradigma inovador de modelar ambientes virtuais, o MOrFEu (MENEZES, NEVADO, *et al.*, 2008), no qual seus autores organizam a estrutura e o uso de ambientes virtuais de uma forma diferenciada.

Parte-se de estruturas simples de suporte à autoria individual e coletiva, e da socialização de suas produções, resultando em um documento compartilhado. Esse

documento resultante é a expressão das ferramentas de comunicação/interação nos ambientes virtuais. Ou seja, o ato de interagir através de um ambiente virtual é visto como o ato de criar coletivamente um documento compartilhado.

Baseado no MOrFEu, um primeiro arcabouço de construção de ambientes virtuais foi desenvolvido (SANTOS, CASTRO e MENEZES, 2010) (SANTOS, CASTRO e MENEZES, 2012). Nesse arcabouço, já era possível a implementação de diversos ambientes virtuais passíveis de modificações em tempo de execução.

No entanto, ele sofria de algumas limitações. Alguns tipos de ambientes virtuais não eram implementáveis diretamente, era necessário se fazer uso de alguns artifícios, por exemplo, o conceito de estado e mudanças de estado. Outra deficiência conceitual do MOrFEu era que ele não possuía um sistema de permissões, requisito forte de alguns tipos de ambientes virtuais.

Assim, analisando-se suas limitações e suas caraterísticas, juntamente com as características dos ambientes virtuais candidatos a serem implementados por tal arcabouço, propôs-se outro arcabouço para desenvolvimento de ambientes virtuais flexíveis. Esse arcabouço foi chamado de **MX**, ou MOrFEu eXtendido.

Além de corrigir deficiências do MOrFEu, as principais características do MX estão na modelagem formal de ambientes virtuais, possibilitando um mapeamento direto para uma implementação. A modificação de ambientes assim implementados pode ser feitas apenas modificação suas especificações, alterando assim suas funcionalidades. Além disso, o MX introduz o conceito de agentes que atuam em ambientes virtuais, realizando algumas funcionalidades comuns entre os ambientes virtuais atuais, como busca e funções de percepção.

Com o arcabouço MX, foi possível especificar formalmente 20 tipos diferentes de ambientes virtuais, desde os mais simples e comuns como fórum, blog e chat, passando por ambientes virtuais mais complexos, como aqueles que dão suporte às atividades complexas de arquiteturas pedagógicas (CARVALHO, NEVADO e MENEZES, 2005). Além disso, outros sistemas inusitados que podem ser considerados

como ambientes virtuais, como é o caso de um jogo de Dominó On-line, também foram descritos nesse arcabouço. Essa quantidade de ferramentas descritas com esse arcabouço tem o objetivo de mostrar a capacidade da proposta de descrever precisamente uma grande quantidade e diversidade de ambientes virtuais, buscando cobrir a maior gama possível de ambientes virtuais de interação humano-computador-humano.

1.1 Objetivo

O objetivo deste trabalho é conceber um arcabouço de desenvolvimento de ambientes virtuais de interação humano-computador-humano de forma que os ambientes construídos a partir desse arcabouço sejam flexíveis em sua criação e modificáveis em tempo de execução.

1.2 Contribuições

As contribuições deste trabalho são relativas à criação de ambientes virtuais que sejam alteráveis em tempo de execução e à formalização de conhecimento sobre ambientes virtuais de interação humano-computador-humano.

Os ambientes implementados no arcabouço desenvolvido são alteráveis em tempo de execução, mantendo os dados que já haviam sido incluídos. E para que essas alterações sejam feitas, basta alterar a especificação desses ambientes. Essas alterações incluem mudanças em suas estruturas e funcionalidades. Isso possibilita uma melhor adequação do propósito de um ambiente virtual, que é ele deve evoluir conforme as necessidades de seus usuários evoluem.

Dessa forma, espera-se uma nova era no desenvolvimento de ambientes virtuais, com cada grupo de usuários utilizando seus ambientes virtuais personalizados e adequados para suas necessidades de interação. Assim, espera-se combater o problema

identificado na literatura da "adequação das atividades aos recursos disponíveis". Portanto, uma nova geração de diversos ambientes virtuais pode surgir.

Além disso, apresenta-se uma maneira formal de se definir e descrever ambientes virtuais. Isso proporciona os seguintes desdobramentos:

- Proporciona uma linguagem franca de comunicação entre os desenvolvedores de ambientes virtuais, descrevendo de maneira exata um dado ambiente virtual, permitindo sua descrição precisa e comparação com outros ambientes;
- Permite raciocínio sobre o conhecimento formalizado de um ambiente virtual, possibilitando uma série de ações que podem ser realizadas por agentes inteligentes artificiais;
- Permite a criação de ambientes virtuais inovadores para atender às diversas demandas dos atuais cenários e usuários da Web, pois é possível a implementação imediata, no arcabouço desenvolvido, desses ambientes formalizados.

Além disso, o desenvolvimento desses ambientes será agilizado, dado que a "programação" de um ambiente virtual passa a ser a definição de suas estruturas e funcionalidades, feito de maneira declarativa, o que reduz o tempo de desenvolvimento, pois gera uma quantidade de código bem menor, permite cópia (reuso) de código de uma aplicação para outra, e as dificuldades técnicas de implementação de aplicações web são superadas.

1.3 Publicações

Dentro do contexto deste trabalho, foram publicados três artigos em eventos:

- "MOrFEU: Multi-Organizador Flexível de Espaços VirtUais para Apoiar a Inovação Pedagógica em EAD", Simpósio Brasileiro de Informática na Educação (SBIE 2008) (MENEZES, NEVADO, *et al.*, 2008);
- "MOrFEu: Criando Ambientes Virtuais Flexíveis na Web para Mediar a Colaboração", no Congreso Iberoamericano de Informática Educativa (IE 2010) (SANTOS, CASTRO e MENEZES, 2010);

- "Flexible Virtual Environments for Teaching and Learning", na conferência Frontiers in Education (FIE 2012) (SANTOS, CASTRO e MENEZES, 2012).

Ainda no período do desenvolvimento do curso de doutorado, o autor deste trabalho publicou os seguintes artigos:

- "Fleshing Out Clues on Group Programming Learning", na conferência International Conference on Enterprise Information Systems (ICEIS 2009) (CASTRO, FUKS, *et al.*, 2009);
- "Impacto da usabilidade na educação a distância: um estudo de caso no Moodle IFAM", no Simpósio Brasileiro de Fatores Humanos na Computação (IHC 2010) (MAGALHÃES, GOMES, *et al.*, 2010).

1.4 Método

Este trabalho de iniciou com uma **revisão bibliográfica**, quando se tomou conhecimento das dificuldades em se desenvolver ambientes virtuais para dar apoio às Arquiteturas Pedagógicas. Logo se percebeu que esse não era um problema isolado, que a literatura e a prática de alguns colegas subsidiavam a existência desse problema e de como ele vinha sendo tratado sem sucesso. No rumo de tentar solucionar o problema de flexibilidade em ambientes virtuais, encontrou-se a proposta do MOrFEu que parecia promissora para tratar esse problema.

Logo se percebeu que os **trabalhos relacionados** listados na literatura não tratavam o problema de maneira adequada. Todos tratavam o desenvolvimento de ambientes virtuais como um desenvolvimento de uma ferramenta de software tradicional. Não atentando para as características de seus usuários e para as características da tarefa desses ambientes. Esse problema era mais agudo com as Arquiteturas Pedagógicas.

O primeiro passo foi **testar a proposta do MOrFEu** e verificar se ela era adequada para modelar ou explicar as ferramentas atuais de comunicação/interação. Um ponto importante a ser ressaltado é que nas abordagens tradicionais uma

aplicação de comunicação/interação não é considerada isoladamente um ambiente virtual. No entanto, observou-se que por suas características e da teoria presente na literatura esse tipo de ferramenta poderia ser considerada sim um ambiente virtual.

Após a **análise** das principais ferramentas de comunicação/interação conhecidas e os ambientes virtuais tradicionais, percebeu-se que a proposta do MOrFEu se encaixa para explicar essas ferramentas, segundo os conceitos do MOrFEu.

Partindo dessa experiência positiva, propôs-se uma forma de se modelar essas ferramentas através de diagramas, segundo os conceitos do MOrFEu. Em mais uma experiência satisfatória, constatou-se que os **conceitos do MOrFEu eram adequados para se modelar esses ambientes virtuais**.

Como **prova de conceito**, foram implementadas algumas dessas ferramentas que tinham sido modeladas. E para agilizar o processo de implementação, foi desenvolvido um meta-ambiente virtual para o desenvolvimento desses ambientes segundo os conceitos do MOrFEu. Então, as ferramentas já implementadas foram reimplementadas nesse arcabouço e muitas outras que não tinham implementação conhecida também foram implementadas.

Alguns ambientes mais complexos também foram acomodados e implementados nesse estágio, ambientes para suporte a algumas Arquiteturas Pedagógicas. Eram ambientes um pouco mais complexos e muitas ainda não possuíam implementação conhecida. Todas essas implementações foram feitas de maneira muito rápida se comparadas a um processo tradicional de desenvolvimento, e com pouca quantidade de código.

Desejava-se **verificar a viabilidade** proposta em uma situação real de uso, foi feito um **estudo de caso** do uso de um ambiente virtual para suporte a uma arquitetura pedagógica chamada Debate de Teses. Os participantes eram professores da rede pública de ensino que lecionavam no interior do estado do Amazonas. Eles foram encorajados a aprender como seria a aplicação dessa atividade em suas escolas

de origem, e que fizessem o máximo de sugestões possíveis à melhoria do ambiente, ou personalização do mesmo para uso em sua realidade.

Um ponto positivo desse estudo de caso foi que muitas das solicitações de modificação já puderam ser realizadas em tempo de execução, e com pouco tempo de desenvolvimento. No entanto, **algumas dessas alterações não puderam ser atendidas prontamente**.

Analisando as modificações frustradas, puderam-se **identificar deficiências** principalmente no protocolo de interação do arcabouço proposto.

Nesse ponto, buscou-se **identificar situações parecidas àquelas levantadas no estudo de caso**. Situações que ocorreriam com o uso de outras ferramentas de comunicação e interação. Por exemplo, com ferramentas com alto grau de controle de *workflow* como em um jogo on-line, ou em ferramentas com um índice de modificações elevado, que está em constantes modificações de suas funcionalidades, como o Facebook[1]. Assim, foi possível identificar outras necessidades e deficiências para o arcabouço proposto. Ao mesmo tempo em que ficavam ainda mais claro os elementos comuns a todos esses ambientes virtuais.

Desenvolveu-se então um modelo formal dos componentes de ambientes virtuais segundo a perspectiva do MOrFEu. Assim foram **modeladas** todas as ferramentas que já tinham sido objeto de estudo com o arcabouço anterior e ainda mais algumas ferramentas que não tinham sido levadas em consideração. Isso foi feito como **prova de conceito** de que o modelo proposto cobria as mais diversas situações e características entre os ambientes virtuais conhecidos.

Então, foram implementados alguns ambientes virtuais que, na sua somatória, possuíam todas as características do modelo formal proposto. Essa implementação foi bem natural, apenas traduzindo o que já estava definido em lógica de primeira ordem no modelo formal proposto para o Prolog, uma linguagem de programação em lógica.

[1] http://www.facebook.com

Essa tradução foi feita com pouquíssima diferença na semântica utilizada na representação em lógica de primeira ordem.

Esses últimos protótipos demonstraram a **factibilidade da proposta** e que os ambientes virtuais se tornavam concretos apenas com a definição declarativa de suas funcionalidades.

Algumas **simulações de modificações** dos ambientes também foram desenvolvidas, visando mostrar sua capacidade em acomodar modificações em tempo de execução. Todas foram bem sucedidas.

Dessa forma, conclui-se que o modelo formal para ambientes virtuais baseado no MOrFEu se mostra factível para a modelagem e desenvolvimento de ambientes virtuais, de forma que os ambientes criados dessa forma são desenvolvidos de uma maneira mais rápida que nas abordagens tradicionais e com uma quantidade de código inferior, e que acomodam alterações em suas definições em tempo de execução.

1.5 Organização do texto

2 Contextualização

Atualmente, os usuários da Web apresentam complexas e numerosas demandas, e as aplicações têm aumentado em quantidade e em qualidade para atendê-los, o que transformou a Web numa plataforma de intenso desenvolvimento. Esses usuários não podem esperar pelo ciclo tradicional de desenvolvimento de software, gerando, assim, a necessidade por abordagens mais flexíveis para a criação de novas ferramentas de software.

A Web 2.0 – também chamada de Web Social – foi o nome dado ao modo diferenciado que a Web passou a ser usada nos últimos dez anos. Ela mudou de uma mídia para uma plataforma, de uma "web de leitura" para uma "web de leitura e escrita" (ULLRICH, BORAU, *et al.*, 2008). Isso, associado com uma maior oferta de acesso à Internet, fez crescer seu número de usuários. Essas pessoas passaram a ser incentivadas a produzir mais conteúdo (autoria) e interagirem (socializarem) através da Web. Isso mudou a forma como eles pensam a rede, de simples espectadores passaram a ser protagonistas, produtores de conteúdos e artefatos.

Para atender a esses novos usuários, os desenvolvedores passaram a cuidar de problemas relacionados com a Interface Humano-Computador (IHC) e a construírem aplicações cada vez mais parecidas com aquelas feitas para *Desktop*. A Web passou a ser, então, a primeira opção de plataforma para se projetar um sistema, com auxílio de vários *frameworks* e *kits* de desenvolvimento (TAIVALSAARI, MIKKONEN, *et al.*, 2008).

Do mesmo modo, a diversidade de aplicações na Web e o número crescente de usuários fizeram surgir demandas por outras aplicações personalizadas às atividades dessas pessoas. Os *mashups* são um exemplo de solução desse problema, uma vez que

unificam diversas aplicações e geram uma nova aplicação que muitas vezes possui funcionalidades diferentes das originais, vistas isoladamente. Outra característica é que os próprios usuários são criadores desses artefatos, sendo então considerados usuários-finais programadores[2] (*end-user programmers*) (AHMADI, JAZAYERI, *et al.*, 2009).

As práticas educativas realizadas nesses ambientes virtuais deixam ainda mais evidente as necessidades de novos tipos de aplicações. Por exemplo, a pesquisa em Arquiteturas Pedagógicas (FAGUNDES, NEVADO, *et al.*, 2005) fez surgir atividades que necessitam de ferramentas de software específicas para lhes dar suporte. A ideia que norteia essas arquiteturas é "pensar a aprendizagem como um trabalho artesanal" (CARVALHO, NEVADO e MENEZES, 2005). Segundo essa postura, cada atividade demanda um suporte diferenciado, personalizado, com pequenas modificações nas aplicações. Mais exemplos de atividades podem ser encontrados nos trabalhos de Nevado, Carvalho e Menezes (2007) e Nevado, Dalpiaz e Menezes (2009).

Na seção a seguir, é apresentado um breve levantamento histórico no desenvolvimento de ferramentas de software para a Web. E na Seção 0, considera-se um domínio complexo, o da educação, para evidenciar a necessidade por flexibilidade nas aplicações web.

2.1 Histórico da Evolução das Tecnologias para Web

A "World Wide Web" (WWW) tem passado por intensas mudanças tecnológicas desde a sua criação, em 1990, por Berners-Lee e seus colegas do CERN (BERNERS-LEE, 2000). No entanto, até hoje, o conceito que Berners-Lee propôs não foi modificado. E, atualmente, a Web está convergindo para uma de suas ideias

[2] Myers, Ko e Burnett (2006) definem usuários-finais programadores como "pessoas que escrevem programas, mas *não* como sua ocupação principal. Em vez disso, eles têm que escrever programas para alcançar seu objetivo principal".

principais: “um sistema no qual compartilhar o que você sabe ou pensa seja tão fácil quanto aprender o que outra pessoa sabe” (p. 33).

Nesta seção, apresenta-se um breve histórico das características e tecnologias relacionadas à Web, bem como suas estratégias de desenvolvimento.

2.1.1 A Primeira Geração

A Web começou quando Berners-Lee (2000) criou um sistema de hipertexto distribuído, em 1990. Preparada para ser utilizada em qualquer sistema operacional que tivesse um browser (um software cliente), a Web foi aos poucos se tornando uma mídia de alcance mundial. Nessa primeira geração, pouco havia de interação entre usuários usando a Web como meio. Em sua maioria havia sites que mostravam propagandas de empresas e proviam serviços.

Berners-Lee acreditava que havia um poder em organizar as ideias de uma maneira não forçada, em forma de teia (*web*) (2000, p. 3) e que o objetivo final da Web era dar suporte e melhorar nossa existência em forma de teia neste mundo (p. 123). Por isso, apostava que os sistemas de hipertexto seriam a forma ideal de se organizar as ideias. “Hipertexto é uma maneira de ligar e acessar informações de vários tipos como uma teia de nós na qual o usuário pode acessar conforme sua vontade” (BERNERS-LEE e CAILLIAU, 1990).

Ele queria criar um espaço de informação no qual qualquer um teria acesso imediato e intuitivo (2000, p. 157), e que compartilhar o que você sabe ou pensa fosse tão simples quanto aprender algo que outra pessoa sabe (p. 33). Por fim, Berners-Lee conclui que a Web é mais uma criação social que tecnológica, para ajudar as pessoas a trabalharem juntas (p. 123).

Berners-Lee vai mais além, ele deseja que um dia as máquinas sejam capazes de poder analisar todos os dados presentes na Web – o que ele chamou de “Web Semântica” (2000, p. 157) –, de modo a ajudar os usuários a utilizarem a Web.

Em 1990, Berners-Lee e seus colegas do CERN implementaram o primeiro servidor web e o primeiro browser (cliente). Eles continham várias limitações, se comparados às ideias originais de Berners-Lee.

No entanto, desde o início, a Web é composta de cinco componentes básicos: URL, HTTP, HTML, o servidor e o cliente (o browser). A URL[3] (BERNERS-LEE, MASINTER e MCCAHILL, 1994) é o endereço de um site ou servidor na Web, com ele o usuário é capaz de "chegar" aonde deseja. HTTP[4] (BERNERS-LEE, FIELDING e FRYSTYK, 1996) é o protocolo que governa a troca de informações entre o cliente e o servidor web. HTML[5] (BERNERS-LEE e CONNOLLY, 1993) é a linguagem para escrever hipertextos. O servidor web é o programa responsável por entregar hipertextos conforme for solicitado pelo cliente, sendo que essa requisição é feita através do protocolo HTTP. O cliente é o browser, um programa que recebe arquivos em HTML e os processa para mostrá-los ao usuário. O usuário interage com esse programa para navegar nas páginas através dos links. Esse clique nos links é processado pelo browser e se torna em uma nova requisição HTTP.

Esses cinco elementos sofreram algumas mudanças em suas especificações no decorrer do tempo. A URL passou, por exemplo, de simples acesso a documentos únicos para acessos com passagem de parâmetros, que são interpretados pelo servidor (BERNERS-LEE, FIELDING e MASINTER, 2005). No HTTP, foi melhorada a questão de transmissão na Internet, causando um menor impacto no fluxo de dados pela rede e se tornando um protocolo "bem comportado" (GETTYS, 1996). O HTML sofreu diversas modificações, a maioria visando uma padronização de recursos disponíveis entre os diversos browsers (RAGGETT, HORS e JACOBS, 1999).

Tratando-se do servidor, a geração das páginas web passou de simples entrega de conteúdos de arquivos para geração dinâmica de páginas, gerando páginas HTML principalmente a partir de dados presentes em banco de dados e parâmetros passados

[3] Uniform Resource Locators
[4] Hypertext Transfer Protocol
[5] Hypertext Markup Language

pela URL (método GET) ou por método POST – quando é passada uma quantidade maior de dados que trafega na requisição HTTP feita pelo cliente. Esse processo se iniciou com o CGI[6], em 1993, que é uma interface simples para executar programas externos ao servidor web de uma maneira que seja independente de plataforma (ROBINSON e COAR, 2004). Depois, surgiram linguagens de programação que executavam diretamente no servidor web. Entre elas, pode-se citar: PHP, JSP[7] e ASP[8]. Essas linguagens ganharam destaque no desenvolvimento de aplicações para a Web por terem funcionalidades específicas para esse domínio, misturando códigos-fonte com conteúdo HTML num mesmo arquivo.

Por sua vez, o cliente Web foi o que mais sofreu melhorias. Como surgimento do CSS (LIE e BOS, 2008) e da linguagem JavaScript (RAGGETT, 1997), em 1998, o Browser passou a ter que processar um conjunto mais complexo de elementos. As páginas ficaram melhor organizadas visualmente e mais interativas. Outras pequenas aplicações passaram também a executar no cliente através da instalação de plug-ins, como Adobe Flash e Java Applet. Naquele momento, os documentos da Web, que em sua maioria eram estáticos, tinham se transformado em documentos com pequenos pedaços embutidos de interatividade (RAMAN, 2009).

Em seguida, surgiram os frameworks de desenvolvimento para a Web. Esses conjuntos de códigos para implementação de funcionalidades que comumente existem em aplicações web – por exemplo, registro de usuários, *login*, rotinas de manipulação de banco de dados e *templates* de telas – auxiliam no desenvolvimento rápido, padronizado e com maior segurança. Exemplos desses recursos são: CakePHP, Drupal, Joomla, Yii, Zend, Apache Struts, Django e Ruby on Rails.

Por volta do ano 2000, começou um processo de desenvolvimento de aplicações "sociais", fenômeno que foi chamado de "Web 2.0". E o surgimento de novas tecnologias impulsionou esse processo.

[6] Common Gateway Interface
[7] Java Server Pages
[8] Microsoft Active Server Pages

2.1.2 Web 2.0

A nova forma como a Web tem sido utilizada nos últimos dez anos, que foi chamada de "Web 2.0" ou "Web Social", tem atraído um número cada vez maior de usuários, que passaram a ser mais participativos e produtores de conteúdos. Esse já era o objetivo inicial de Berners-Lee (2000) e esse fenômeno foi observado e primeiro relatado por O'Reilly (2007), que enumerou algumas características e tendências, organizando-as como elementos da Web 2.0. A Web tornou-se uma plataforma de desenvolvimento de aplicações com projeto preocupado com usabilidade e em atrair o maior número de pessoas possível, visto que eles agora são os grandes produtores de conteúdo, o que tem se tornado cada vez mais precioso. Com mais usuários participando do processo de criação de conteúdo e do uso de novas aplicações, surgiu a necessidade por aplicações personalizadas ao uso que esses usuários gostariam de dar a elas.

No início do desenvolvimento da Web, Berners-Lee e seus colaboradores tiveram que fazer uma escolha técnica que perduraria por vários anos. O browser seria apenas um leitor de páginas em hipertexto em vez de também editar esses arquivos. Isso "deixou várias pessoas pensando a Web como um meio no qual poucos publicavam e muitos navegavam". A visão dele era de "um sistema no qual compartilhar o que você sabe ou pensa, fosse tão fácil quanto aprender o que outra pessoa sabe" (BERNERS-LEE, 2000, p. 33). Essa situação perdurou há até pouco mais de dez anos, quando as aplicações passaram a permitir que os usuários gerassem conteúdo e o postassem na rede de maneira mais fácil. Um exemplo desse tipo de aplicação são os blogs. Por ter raízes na origem da Web, Taivalsaari *et al.* (2008) defendem que "a Web 2.0 não é uma ideia nova, é mais um termo de marketing, rodeado por muito modismo".

O'Reilly (2007) enumerou várias características dessa "nova" Web, e foi o primeiro a fazê-lo, seguido por diversos outros autores, que fomentaram a discussão. Entres eles estão Anderson (2007) e Ullrich *et al.* (2008), que, de maneira mais

sistematizada, descreveram essas características, as quais, segundo o contexto desta investigação, podem ser descritas como:

Produção Individual e Conteúdo Gerado por Usuário. Os usuários têm sido incentivados a produzir mais conteúdo. Diferentemente da situação na primeira geração, onde apenas os criadores de sites tinham capacidade de gerar conteúdo na Web, hoje existem sites que possibilitam a qualquer pessoa criar uma página e começar a postar na Web qualquer conteúdo que desejar. Berners-Lee (2000, p. 157), quando criou a Web já tinha esse intuito: "Eu sempre imaginei o espaço de informação como algo em que qualquer um teria acesso imediato e intuitivo, e não apenas navegar, mas criar". Sites para criar páginas pessoais, blogs e wikis são exemplos de aplicações que permitem que os usuários criem facilmente um espaço onde podem postar suas opiniões e informações.

O Poder das Massas. Ullrich *et al.* (2008) disseram: "Os serviços da Web 2.0 são caracterizados pelo fato de seus valores aumentarem conforme o número de pessoas que os estão usando". Os usuários proveem dois tipos de informação para um site da Web 2.0 quando o estão usando, implícito e explícito. Tomando como exemplo um site de uma loja virtual, os usuários criam conteúdo explícito quando escrevem opiniões sobre os produtos e fornecem informações implícitas quando compram mais de um produto, informando à loja que esses dois produtos têm relação e que provavelmente outros usuários também gostariam de comprar esses produtos juntos. As redes sociais também possuem muitas informações sobre seus usuários, perfis, preferências, amizades, etc. Essas informações podem ser usadas de diversos modos, como por exemplo, propaganda individual para cada usuário, o que aumenta o poder de venda daquela propaganda.

Dados em Escala Épica. Com tantas pessoas participando e produzindo conteúdo na Web, encontra-se disponível uma imensa massa de dados relativos aos mais diversos tópicos, e esses dados estão em múltiplos formatos. Para aproveitar e processar essas informações Berners-Lee também antecipou o conceito de "Web

Semântica": "as máquinas se tronarem capazes de analisar todos os dados na Web" (BERNERS-LEE, 2000, p. 157). No entanto, por causa da variedade de dados e dos dados não estarem rotulados, atualmente isso é impossível. Mas muito tem sido feito e pesquisado em domínios restritos.

Potencializando a Cauda Longa. O conceito estatístico da "cauda longa" (*long tail*) foi usado por Anderson (2006) para explicar um fenômeno econômico, e hoje é usado para explicar uma mudança de paradigma com a Web 2.0. Trata-se do fato de a Web estar possibilitando um suporte melhor a assuntos (ou aplicações, informações, produtos, etc.) que não são de interesse da maior parte das pessoas, mas apenas de nichos. Anderson (2007) exemplifica com o fato de que atualmente a Wikipedia (WIKIMEDIA FOUNDATION, 2011) tem milhões de páginas e a maioria é sobre assuntos que não são *hot topics* (temas de grande interesse). Isso acontece porque as poucas pessoas interessadas nesses assuntos têm possibilidade de produzirem esses conteúdos.

O Beta Perpétuo. "Beta" é o nome dado à versão de um software que está pronto para uso, mas ainda precisa passar por testes com usuários reais. Os serviços da Web 2.0 adotaram essa estratégia de desenvolvimento para proporem sempre novas funcionalidades e mudanças. Essa prática faz com que os seus usuários pensem essas aplicações como aplicações dinâmicas, esperando novas funcionalidades e mudanças das existentes. Deixando de pensar essas aplicações como produtos finais, essas pessoas passam a questionar as funcionalidades de outras aplicações, passando a imaginar o que mais um dado serviço poderia oferecer. Isso pode mudar o modo como as pessoas geralmente pensam aplicações de computador de "como eu posso melhorar minhas atividades com o uso dessa aplicação" para "como essa aplicação pode melhorar para me auxiliar melhor em minhas atividades".

Assim, é fácil observar que, no estágio atual da Web, ela é mais uma criação social que uma criação técnica, para ajudar pessoas a trabalharem juntas e não como um brinquedo tecnológico, como foi dito por Berners-Lee (2000, p. 123).

Taivalsaari *et al.* (2008) fizeram uma distinção importante das características da Web 2.0. Eles as dividiram em características de "colaboração" e características de "interação".

"Colaboração" refere-se ao aspecto "social", que "permite que um vasto número de pessoas colabore e compartilhe os mesmo dados, aplicações e serviços na Web" (TAIVALSAARI, MIKKONEN, *et al.*, 2008). Isso já foi discutido neste trabalho quando se expôs as características apontadas por O'Reilly (2007).

O aspecto de "interação" foi permitido graças às chamadas "tecnologias da Web 2.0". Refere-se ao fato de que Berners-Lee (2000, p. 157) chamou de "intuitivo". Fazer os Sites "se comportarem mais como aplicações para desktop, por exemplo, possibilitando páginas web atualizar um elemento da interface do usuário por vez, ao invés de ter que atualizar a página inteira toda vez que alguma coisa mudar" (TAIVALSAARI, MIKKONEN, *et al.*, 2008). Essa tecnologia ficou conhecida como Ajax, mais precisamente XML HTTP Request (KESTEREN e JACKSON, 2010), que surgiu em 2006, que permite serem feitas requisições assíncronas para o servidor Web através de Java Script e que apenas algumas partes das páginas sejam atualizadas sem que seja necessário um novo carregamento da página inteira.

Com aplicações cada vez mais parecidas com aplicações de desktop – com janelas, barras, botões e ícones – foi possível melhorar a usabilidade dessas aplicações, pois seus usuários estão mais acostumados com esse tipo de interface, o que os deixa numa posição mais confortável. Isso habilitou um número maior de pessoas a utilizarem essas aplicações.

Hoje, são vários os *frameworks* e *kits* de desenvolvimento para aplicações desse tipo, também chamadas de *Rich Internet Applications* (aplicações ricas da Internet). Alguns exemplos dessas tecnologias são: Adobe Integrated Runtime (AIR), Google Web Toolkit (GWT), JavaFX, Microsoft Silverlight, Ruby on Rails e Morfik.

Mais usuários e produtores de conteúdo; serviços fornecidos a grupos pequenos de usuários; aplicações dinâmicas; tudo tem contribuído para uma nova mentalidade

dos usuários da Web 2.0. Isso também tem gerado uma demanda crescente por software especializado, adequado às atividades de pequenos grupos de usuários. No entanto, esses usuários não podem esperar o ciclo tradicional de desenvolvimento de software.

Técnicas de *tailoring* têm sido propostas para lidar com esses problemas (AHMADI, JAZAYERI, *et al.*, 2009), ou seja, permitir que o usuário modifique as aplicações conforme ele a usa ao invés de isso ser feito no momento do projeto daquela aplicação (MACLEAN, CARTER, *et al.*, 1990). Isso é discutido dentro do contexto de programação por usuário-final (*end-user programming*), quando as aplicações são construídas por usuários-finais programadores (*end-users programmers*), "pessoas que escrevem programas, mas *não* como sua ocupação principal. Em vez disso, eles têm que escrever programas para alcançar seu objetivo principal" (MYERS, KO e BURNETT, 2006). Ahmadi, Jazayeri *et al.* (2009) expõem um modelo para elaboração colaborativa de ferramentas por usuários-finais programadores, que também tem como atores programadores, que criam linguagens de programação para usuário-finais e ambientes de simulação, e programadores de domínio específico. Essa infraestrutura foi organizada por eles como exposto na Figura 1.

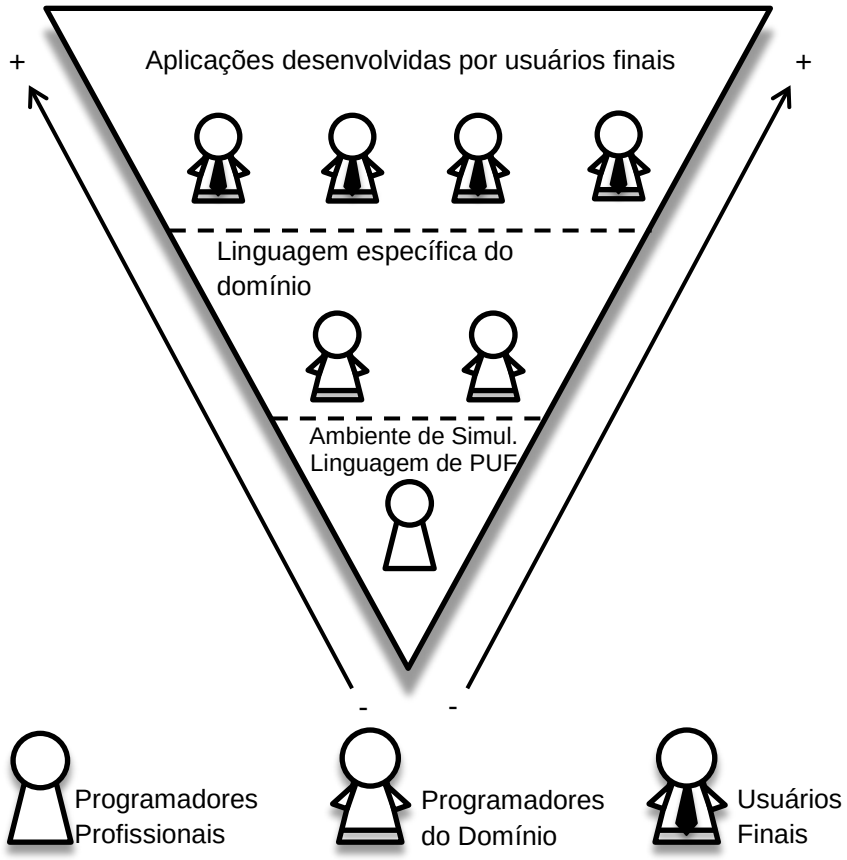


Figura 1. Infraestrutura para construção de aplicações por usuários-finais. Fonte: (AHMADI, JAZAYERI, *et al.*, 2009).

Os *mashups* são outro exemplo de solução, unificando diversas aplicações, gerando assim uma nova aplicação que, muitas vezes, possui funcionalidades diferentes das originais, vistas isoladamente. Seus criadores, muitas vezes, não são programadores profissionais, mas usuários-finais programadores. Esses *mashups* podem ter as seguintes funções (RAMAN, 2009):

- **Agregação**: novos artefatos da Web podem ser criados agregando elementos já existentes na Web. Ou seja, integrar dados de múltiplas fontes em uma única visão.
- **Projeção**: uma informação disponível na Web pode ser filtrada para se adequar ao contexto de navegação do usuário. Ou seja, podem ser feitas múltiplas visões de um mesmo pedaço de dados. Por exemplo, pode ser feita uma representação visual que mostra dados históricos como uma tabela de números ou como um histograma.

Diante do exposto, observa-se que os usuários se tornaram os atores principais na Web. Eles tanto produzem conteúdo quanto têm a necessidade de produzir também aplicações. Na chamada Web 1.0, não havia tanta quantidade de informação disponível, pois os criadores dessas informações se concentravam apenas em *hot topics* (temas de grande interesse). O mesmo acontece no atual estágio de desenvolvimento de aplicações, os criadores de aplicações não são capazes de desenvolver todas as aplicações que os usuários demandam, apenas as de maior interesse. Portanto, o poder de desenvolvê-las deve ser passado aos usuários, os maiores interessados em seu uso.

2.2 Ambientes Virtuais de Interação Humano-Computador-Humano

Ambientes Virtuais de Interação Humano-Computador-Humano Baseados na Web são sistemas web que reúnem virtualmente várias pessoas em um mesmo lugar possibilitando a interação entre elas. Clubb (2007) e Paredes (2007) são os primeiros a utilizar o termo "Interação Humano-Computador-Humano" (*Humano-to-Computer-to-Human Interaction* – HCHI). Trata-se de um tipo de interação entre as pessoas que utiliza os computadores e os sistemas de rede de computadores e a Internet como mídia para viabilizar essa interação. Para simplificar, neste trabalho fará referência apenas a "ambientes virtuais" quando se estiver falando de ambientes virtuais de interação humano-computador-humano, especialmente aqueles baseados na Web. O termo "espaço virtual" também é visto como um sinônimo para "ambiente virtual", sem diferenciação de semântica neste trabalho.

Há ainda outra definição para Ambiente Virtual, relacionada com Realidade Virtual, que não é o foco deste trabalho. Os ambientes de realidade virtual são diferentes dos ambientes virtuais estudados neste trabalho. Aqueles simulam a realidade da melhor maneira possível, priorizando a imagem e outras sensações

sensoriais humanas. Os ambientes virtuais tratados nesse trabalho têm caráter mais abstrato.

Os *groupware*, por sua vez, possuem características muito semelhantes aos ambientes virtuais tratados aqui. *Groupware* são sistemas baseados em tecnologias de computação e comunicação que ajudam grupos de participantes, e ajudam a dar suporte a um ambiente compartilhado (ELLIS e WAINER, 2000). Eles também propiciam a interação entre as pessoas, mas não se restringem apenas à Web. Assim, aqueles *groupware* que são baseados na Web também são ambientes virtuais. Desta forma, algumas características de *groupware* também podem ser atribuídas a esses ambientes.

Os ambientes virtuais baseados na web são mais restritos, limitando-se às tecnologias da Web, como HTML e Javascript. No entanto, na Web, os ambientes propiciam formas mais ricas de interação entre as pessoas e geralmente possuem artefatos diferentes e mais flexíveis que os artefatos presentes ao nosso redor, por se trataram de artefatos abstratos.

Alguns *groupware* e os ambientes virtuais de aprendizagem (AVAs) são ambientes virtuais baseados na web, da forma como foi colocado neste trabalho. Mas é preciso ressaltar que os *groupware* estão intimamente relacionados com a área de pesquisa *Computer Supported Collaborative Work* (CSCW), e os AVAs com *Computer Supported Collaborative Learning* (CSCL). Ambos são de especial interesse deste trabalho, pois são a maioria dos ambientes virtuais baseados na web.

A diferença entre CSCW e *groupware* é que CSCW descreve a pesquisa e *groupware* descreve a tecnologia (SCHMIDT e BANNON, 1992). CSCW é a área de pesquisa que estuda como as pessoas utilizam a tecnologia, com relação a hardware e software, para trabalhar juntas em tempo e espaços compartilhados (RAMA e BISHOP, 2006).

Os *groupware* podem ser definidos como um processo intencional de grupos de pessoas mais o software para dar suporte a eles (JOHNSON-LENZ e JOHNSON-

LENZ, 1981 *apud* PENICHET, MARIN, *et al.*, 2007). Ellis *et al.* (1991) definem os *groupware* como sistemas de computação que dão suporte a grupos de pessoas que tomam parte em uma tarefa em comum e que provêm uma interface para um ambiente compartilhado. Várias tecnologias de comunicação podem ser utilizadas para implementar tais sistemas, mas quando o *groupware* é um sistema web, ele é considerado um ambiente virtual.

Várias ferramentas são consideradas *groupware*. Pode-se citar: fax, e-mail, sistema de telefone por IP, chat, gerenciador de documentos, fórum, mecanismos de aprovação, calendários de grupo, planejamento compartilhado, workflow, gerenciamento de eventos, agenda, quadro branco compartilhado, sistemas de notificação, etc. (PENICHET, MARIN, *et al.*, 2007). Mas aqueles que são baseados na web são os de interesse deste trabalho.

O termo CSCL é derivado do termo CSCW, e uma das tecnologias estudadas por essa área de pesquisa são os AVAs. Watson e Watson (2007) definem os AVAs *Learning Management System* (LMS), *Course Management System* (CMS) e *Learning Content Management System* (LCMS). LMS é um *framework* que cuida de todos os aspectos do processo de aprendizagem, não se limitando apenas ao ambiente virtual, mas também cuida das atividades e processos realizados nesse ambiente. LMS também é um termo utilizado para se referenciar a AVAs em geral. E um CMS é um ambiente que provê o instrutor com um conjunto de ferramentas e um *framework* que permite a criação relativamente fácil de conteúdos de cursos on-line e o subsequente ensino e gerência do curso, incluindo as interações dos estudantes do curso (EDUCAUSE EVOLVING TECHNOLOGIES COMMITTEE, 2003 *apud* WATSON e WATSON, 2007). Por fim, um LCMS é um sistema usado para criar, armazenar, montar e entregar conteúdo personalizado de *e-Learning*[9] na forma de objetos de aprendizagem (OAKES, 2002 *apud* WATSON e WATSON, 2007).

[9] Entende-se por *e-Learning* (aprendizagem eletrônica), como a aprendizagem realizada através de tecnologias digitais.

Os LMS e os CMS são ambientes virtuais. Pois as atividades realizadas em um LMS necessitam de interação entre as pessoas e um espaço compartilhado. E os CMS também cuidam das interações entre os estudantes dos cursos. No entanto, um LCMS lida exclusivamente com objetos de aprendizagem, porém não necessariamente possui interações entre as pessoas e um espaço compartilhado. Assim, os LCMS não podem ser classificados como ambientes virtuais.

Itmazi (2005) afirma que o mercado de AVAs no ano de 2005 possuía pelo menos 200 produtos. Destacam-se no mercado nacional AulaNet (FUKS, 2000), e-Proinfo (MINISTÉRIO DA EDUCAÇÃO, 2000), TelEduc (ROCHA, 2003) e Amadeus[10].No entanto, o ambiente virtual de maior sucesso atualmente no Brasil e também no resto no mundo é o Moodle[11] (DOUGIAMAS e TAYLOR, 2003). Esses e outros ambientes virtuais de aprendizagem são muito semelhantes, as comunidades são organizadas em cursos, um espaço compartilhado onde estão disponíveis arquivos, links e atividades para serem desenvolvidas, possuem sistema de notas e ferramentas para comunicação direta ou ampla entre professores e alunos. A principal diferença entre esses e os outros disponíveis no mercado está no uso de diferentes tecnologias que podem facilitar seu uso, customização e atualização.

Neste trabalho, interessa-se por uma classe específica de aplicações web, a de *groupware* e ambientes virtuais de aprendizagem (AVA). Mesmo trazendo algumas vantagens importantes, se comparados a aplicações web tradicionais, a forma como são projetados dificulta a implementação de estratégias flexíveis para o desenvolvimento de aplicações personalizadas a pequenos grupos de usuários.

Os ambientes virtuais na Web foram criados para dar suporte a atividades de caráter colaborativo – quando pessoas realizam atividades em conjunto –, seja para trabalho ou aprendizagem. Eles reúnem pessoas em um mesmo "lugar" virtual, possibilitando a interação entre elas. Para isso, esses sistemas contam com um

[10] http://amadeus.cin.ufpe.br/
[11] http://moodle.org/

conjunto de ferramentas de comunicação, coordenação e cooperação (FUKS, RAPOSO, *et al.*, 2007).

Superficialmente, eles são apenas um conjunto dessas ferramentas disponíveis em um mesmo lugar. Mas os ambientes podem ser vistos como um micro-mundo da Web, com várias aplicações disponíveis e possibilidade de interação entre diversas pessoas.

Há funcionalidades de natureza pragmática que contribuem para o sucesso desses ambientes. Por exemplo, ferramentas de percepção (*awareness*); registro de atividades (*log*) e análise de atividades dos usuários; organização para realização de atividades (*workflow*); e sistemas de notas (no caso de AVAs).

A principal desvantagem em relação a aplicações web tradicionais é a centralização dos artefatos, que fica em desacordo com a ideia de aplicações distribuídas da Web.

No entanto, a relação custo-benefício é razoável e os ambientes virtuais na web são bem sucedidos para suporte a atividades colaborativas. Itmazi (2005) afirma que o mercado de AVAs no ano de 2005 possuía pelo menos 200 produtos, um número significativo.

2.3 Arquiteturas Pedagógicas e suas Demandas

Neste trabalho, se está interessado também pelo melhoramento no suporte a atividades de ensino-aprendizagem mediadas por ambientes virtuais, por se tratar de um domínio de grande demanda por aplicações flexíveis e rico em situações-problema. Neste contexto, tem se destacado recentemente na literatura as "Arquiteturas Pedagógicas" como atividade que demanda um alto grau de flexibilidade das ferramentas de suporte. Assim, nesta seção, buscam-se mais evidências da necessidade por flexibilidade em ambientes virtuais na Web que dão suporte a essas atividades.

Carvalho *et al.* (2005) definem arquiteturas pedagógicas como

> Estruturas de aprendizagem realizadas a partir da confluência de diferentes perspectivas: abordagem pedagógica, software, internet, inteligência artificial, educação a distância, concepção de tempo e espaço. O caráter destas arquiteturas pedagógicas é pensar a aprendizagem como um trabalho artesanal.

Nessa prática pedagógica, o estudante é o ator principal e o professor um estimulador, um orientador. Deve-se pensar a aprendizagem como um processo de criação de novidades, de descobertas e invenções, permitindo que os sujeitos realizem experimentações, simulações em busca de soluções para questões significativas do ponto de vista do sujeito. E demandam ambientes que sustentem a aprendizagem em rede, em comunidades de aprendizagem. Esses ambientes devem ser flexíveis, de forma a favorecer o protagonismo e a autoria individual e coletiva, oferecendo formas diferenciadas de organizar as interações e produções, tendo como referência espaços de autoria reorganizáveis e flexíveis (MENEZES, NEVADO, *et al.*, 2008).

Ou seja, por ter um caráter "artesanal", as atividades dessas arquiteturas serão diferenciadas para cada grupo de estudantes. Para isso, é necessário que o ambiente virtual que dá suporte a essas atividades seja flexível ao ponto de ser modificado para cada atividade particular. Não há como prever parâmetros de configurações para todas as modificações possíveis, o conhecimento será construído de forma personalizada, demandando suporte computacional também personalizado para cada caso.

Além disso, as atividades concebidas para as arquiteturas pedagógicas são diferenciadas daquelas que os ambientes virtuais tradicionais geralmente dão suporte. Portanto, é necessário que o ambiente possibilite a criação de novas ferramentas para dar suporte a essas atividades. É importante ressaltar que essas atividades ainda possuem caráter artesanal e, por isso, são sujeitas a modificações.

A seguir, apresentam-se a descrição de algumas arquiteturas pedagógicas.

2.3.1 Projetos de Aprendizagem

Na arquitetura pedagógica Projetos de Aprendizagem (FAGUNDES, NEVADO, *et al.*, 2005) os estudantes constroem conhecimento a partir da busca por respostas às suas indagações. A base para o desenvolvimento de um projeto de aprendizagem é o conhecimento anterior, inventariado através de certezas provisórias e dúvidas temporárias. Durante o processo, os protagonistas vão esclarecendo dúvidas, validando certezas e assim construindo conhecimento para responder à dúvida central, denominada de "Questão de Investigação". Há interações entre os autores do projeto, os colegas de classe, os professores e colaboradores externos.

Para concretizar essas interações e o desenvolvimento do projeto, o ambiente virtual deve dar suporte a facilidades para debates, avisos, escrita cooperativa, diário de bordo, livro de visitas, etc.

No entanto, os ambientes virtuais tradicionais impõem dificuldades que obrigam o uso de ferramentas adicionais de apoio. Uma análise dessas dificuldades conduziu à concepção de um ambiente virtual específico, chamado AMADIS.

2.3.2 Júri Simulado

Trata-se de "uma ferramenta a contribuir para a construção do conhecimento por meio do desenvolvimento da argumentação, das possibilidades de cooperação, criatividade e ludicidade" (REAL e MENEZES, 2007). Existe um "Réu", que é o assunto a ser discutido; um "Juiz" que é o professor; a "Defesa" e a "Acusação"; e os "Jurados", que votam por um veredicto.

Um mesmo espaço de interação é divido em subespaços destinados a fóruns de discussão dos grupos de defesa e acusação. Todos os participantes têm acesso a todas as postagens, porém somente é permitida a postagem no espaço da "Defesa" para grupo de defesa; e no espaço de "Acusação", para grupo de acusação.

Após um período determinado de tempo, e que tenha se desenrolado a discussão, o "Juiz" determina que os "Jurados" façam uma votação, para determinar quem ganhou a discussão. Assim, o juiz dá a sentença final sobre o assunto.

Esta arquitetura pedagógica não chegou a ser experimentada, mas foi proposto que o ambiente de suporte fosse implementado através de uma ferramenta wiki.

2.3.3 Construindo Conceituações

Esta arquitetura pedagógica foi desenvolvida por Nevado *et al.* (2009) no contexto de um curso de formação continuada de tutores. "Partindo de um arcabouço inicial, as etapas, produtos e interações foram sendo definidas ao final de cada etapa, sendo inclusive necessária a inserção de novas etapas". Como resultado, foi elaborada uma atividade com cinco etapas.

Na primeira etapa, foi solicitado aos participantes que registrassem suas ideias acerca do tema em uma ferramenta de escrita coletiva de forma livre, sem censuras, visando o mapeamento do conhecimento atual do grupo. A seguir, o professor sintetizou e sistematizou em teses as principais ideias dos participantes.

Para cada participante, foi elaborado um quadro composto pelas teses levantadas e um espaço para que ele pudesse apresentar um posicionamento (concordância ou discordância) acerca de cada tese, devidamente justificado.

Na terceira etapa, cada participante devia revisar o posicionamento de outros participantes (nos quadros deles) acerca das teses, em uma espécie de revisão por pares. Assim, cada participante devia receber duas ou mais revisões para cada posicionamento seu.

Os participantes retornaram ao seu quadro de discussão e deviam fazer uma réplica das revisões feitas por seus colegas.

A atividade terminou com cada participante apresentando uma revisão do seu próprio posicionamento, fortalecendo ou modificando o inicial.

Esta atividade foi implementada com auxílio de uma ferramenta wiki, fazendo necessária a cópia de páginas modelos para a formação de quadros para cada participante.

2.3.4 Diário Virtual

Essa arquitetura pedagógica foi desenvolvida por Serres e Basso (2009) a partir da experiência com atividades de matemática usando ambientes virtuais com estudantes do terceiro ano do ensino médio.

Foram definidos problemas-desafio para cada estudante, que deveria resolvê-lo descrevendo a solução em seu diário, um espaço para elaboração de texto.

Além disso, foi pedido aos estudantes que comentassem a solução de dois outros colegas, visando à interação do ponto de vista da troca de ideias em relação a conhecimentos matemáticas.

O suporte a esta arquitetura foi implementado através de uma ferramenta wiki. Foi criada uma página para cada participante, que seria seu diário. Ao final de cada página, existia um espaço destinado à postagem de mensagens, tanto de revisores quanto de réplicas dos autores.

2.3.5 Dificuldades na Implementação das Arquiteturas

Na Tabela 1, é apresentado um resumo das principais características das arquiteturas pedagógicas citadas. Em algumas atividades foi adotada uma ferramenta Wiki para implementar o ambiente mediador que se desejava, como é o caso do quadro de discussões do “Construindo Conceituações” e o “Diário Virtual”. Um Wiki possibilita a criação de páginas e estruturas de tabelas que podem ser preenchidas com texto posteriormente. Por essa flexibilidade e facilidade na criação de estruturas para textos, o Wiki torna-se um ferramenta facilitadora de desenvolvimento de novas práticas pedagógicas utilizando a Web. No entanto, é necessário um esforço inicial de

criar esses esqueletos de texto e copiá-los para as páginas de cada um dos participantes. E no caso de modificações, é necessário que cada página de participante sofra a modificação.

As regras de interação e o cronograma para a criação do documento foram disponibilizados de forma verbal, e esperava-se que cada participante respeitasse e se responsabilizasse pelo cumprimento do estabelecido.

Observa-se ainda, pelos relatos, que mesmo os ambientes que foram implementados tornaram-se, em geral, inadequados para o trabalho com outros grupos. É difícil predizer como um grupo particular irá colaborar, e cada grupo tem características e objetivos distintos (GUTWIN e GREENBERG, 2000), o que dificulta o reuso do ambiente. Isso recai no problema de adequar a proposta de trabalho aos recursos e possibilidades do software disponível.

Tabela 1. Comparação das dificuldades encontradas nas arquiteturas pedagógicas.

	Projeto de Aprendizagem	Júri Simulado	Construindo Conceituações	Diário Virtual
Complexidade da Atividade	Complexa	Complexa	Complexa	Simples
Posto em Prática	Sim	Não	Sim	Sim
Ferramenta de Apoio	Ambiente Virtual Específico	Sugeriu Wiki	Wiki	Wiki
Protocolo de Interação	Implementado no Ambiente Virtual	Verbal	Verbal	Verbal
Novas Instâncias	Fácil	Médio	Difícil	Difícil
Modificações	Difícil	Fácil	Difícil	Difícil

2.4 Conclusão do Capítulo

A partir do levantamento realizado foi possível obter requisitos para aplicações web mais flexíveis. Para isso, foi analisado o contexto da Web 2.0, do desenvolvimento de ambientes virtuais e das demandas geradas pelas arquiteturas pedagógicas.

Observa-se na Web 2.0 o aumento da quantidade de informação disponível na rede, que se deve ao fato de os usuários serem os produtores de imenso conteúdo.

Logo, para se ter também um aumento na quantidade de aplicações disponíveis, é preciso que a habilidade de construir aplicações seja possibilitada aos usuários.

Com mais produtores de aplicações, haverá mais software produzidos que não serão *hot topic*, ou seja, aplicações que não são de grande interesse do público geral. Esse fenômeno é conhecido como o efeito da cauda longa. Isto quer dizer que nichos de usuários terão a possibilidade de construir suas próprias ferramentas para dar suporte a suas atividades, fazendo surgir uma Web de aplicações (AHMADI, JAZAYERI, *et al.*, 2009). No entanto, isso não acontece hoje em dia. Hoje, os produtores de software estão interessados em aplicações que atraiam um grande número de pessoas.

A cultura do "beta perpétuo" reforça que as aplicações dificilmente estão completas, que não precisam mais de nenhuma alteração. O mesmo acontece com as necessidades dos usuários, pois é natural que sempre aparecem mais necessidades ou algumas delas são modificadas. Assim, é preciso permitir que os usuários alterem suas aplicações conforme suas necessidades.

Além disso, o desenvolvimento de *mashups* mostrou que, ao ser construído, ele se torna mais um elemento acessível na Web, logo, pode servir de fonte para outro mashup. Raman (2009) demonstrou que esse número de aplicações possíveis é da ordem de dois elevado ao número de aplicações existentes hoje na Web ($2^{|W|}$), um número consideravelmente grande. O mesmo deve acontecer com as novas aplicações, elas devem poder utilizar, assim como os mashups, dados de outras aplicações. E o mesmo fenômeno dos "mashups de mashups" poderá ocorrer com as aplicações com dados de outras aplicações. Esse cenário ilustra uma tendência para a intensa criação de aplicações pelos próprios usuários, especialmente em domínios complexos e de forte interação como é o caso da educação, como situado pelas arquiteturas pedagógicas.

Ambas as gerações de conteúdo e de aplicações são produções intelectuais. E elas devem ser o ponto de partida das atividades na Web. Berners-Lee (2000, p. 160) compartilha de uma preocupação semelhante:

> Cada vez que eu escrevo alguma coisa com o computador, eu tenho que escolher se abro a aplicação de "correio eletrônico", a aplicação "net news", ou a aplicação "editor Web". [...] Na verdade, é pedido para que eu selecione qual protocolo usar. O computador deveria descobrir isso por ele mesmo.

Ou seja, o ato de produzir conteúdo ou aplicação não deve estar atrelado a uma plataforma. Esta produção intelectual deve ter existência independente de onde ela for divulgada. Isso implica que o usuário deve ser capaz de produzir conteúdo independente de onde deseja utilizá-lo. Ou até mesmo poderá utilizá-lo em mais de um local.

O mesmo é válido para as aplicações, o que recai na preocupação dos desenvolvedores de linguagens de programação de um sistema ser portável para várias plataformas. Isso já é um fato, quando se trata de aplicações para Web, pois o browser possui comportamento semelhante nas diversas plataformas de computação (TAIVALSAARI, MIKKONEN, *et al.*, 2008). Logo, as aplicações web são acessíveis em diversas plataformas.

Outra preocupação de Berners-Lee (2000, p. 157) é de as máquinas serem capazes de analisar todos os dados presentes na Web. Portanto, as aplicações devem conter conteúdo semântico sobre os dados que carregam, e deve haver a possibilidade da disponibilização desses para outras ferramentas de software em um formato padronizado.

Todos esses requisitos são válidos para os ambientes virtuais, que internamente funcionam como micro-mundos da Web.

E ao analisarmos as necessidades por ferramentas flexíveis das arquiteturas pedagógicas, pode-se observar que elas compartilham dos requisitos aqui citados.

A diversidade das atividades propostas pelas arquiteturas, evidentemente, não são *hot topic*, pois não utilizam apenas as ferramentas tradicionais disponíveis nos ambientes virtuais, como fórum, chat e wiki.

O caráter artesanal das atividades das arquiteturas pedagógicas requer que os ambientes virtuais sejam passíveis de mudanças, que pequenos ajustes sejam feitos,

de modo a adequar o software à atividade realizada. E não o inverso, quando atividade a ser realizada é adaptada às ferramentas disponíveis, como acontece atualmente.

Assim, podem-se resumir as necessidades evidenciadas na lista de requisitos para um ambiente virtual flexível, a seguir:

R1. Os usuários devem ser capazes de construir suas próprias ferramentas;
R2. Os usuários devem ser capazes de fazer alterações nas aplicações conforme sua necessidade e em tempo de execução;
R3. As aplicações devem ter possibilidade de serem compostas por dados de outras aplicações;
R4. A produção de conteúdo deve ser independente de onde ele vai ser utilizado, ou disponibilizado;
R5. As máquinas devem ser capazes de analisar os dados contidos nessas aplicações;

No próximo capítulo, são apresentados trabalhos relacionados que buscam a soluções para o problema apresentado.

3 Trabalhos Relacionados

Alguns trabalhos têm sido propostos para o desenvolvimento de ambientes virtuais mais flexíveis, com ferramentas mais adequadas às necessidades específicas de cada grupo de usuário. Entre elas, destacam-se abordagens baseadas em componentes, linhas de produção de software e *web-services*. Essas soluções partem da visão do desenvolvedor, que enxergam o produto ambiente virtual como um software finalizado, preocupando-se assim com a criação de um bom produto para um determinado grupo de usuários. No entanto, não consideram que as necessidades dos usuários e as funcionalidades requeridas podem mudar no decorrer do uso.

3.1 Aspectos de Groupware

Ellis e Wainer (2000) discutem aspectos de *groupware* que são interessantes para entender as ideias do MOrFEu, mais adiante no texto. Eles discorrem sobre três importantes tipos de *groupware*, que chamaram de aspectos: **o Guardião do Artefato, o Coordenador e o Comunicador**.

Algumas vezes a colaboração entre um grupo de pessoas é centrada no acesso e modificação de um conjunto de dados compartilhados, que foi chamado de artefato. **O Guardião do Artefato** é o conjunto de funcionalidades relacionadas com a manipulação desse artefato. Algumas funcionalidades do guardião são:

- Controle de acesso ao artefato;
- Controle de acesso simultâneo ao artefato;
- Versionamento do artefato;
- Armazenamento de informações do autor e data de modificação;

Algumas vezes colaboração ocorre com cada participante do grupo executa alguma atividade, em uma ordem previamente definida. **O Coordenador** das atividades é o conjunto de funcionalidades relacionadas a essa evolução temporal do sistema, permitindo que uma atividade seja feita depois que todas as atividades anteriores estejam terminadas. As funcionalidades do guardião são:

- Permitir que uma atividade seja feita uma vez que todas as atividade que a antecedem estejam terminadas;
- Notificação aos usuários que eles podem iniciar uma dada atividade;
- Inspecionar o corrente estado do processo;
- Alteração dinâmica da descrição do processo (o plano) para lidar com surpresas;
- Ajudar os participantes a gerenciar seus trabalhos.

Comunicação é o aspecto básico de qualquer tentativa de colaboração. Numa aplicação principalmente guardiã, há comunicação (implícita) quando um participante muda o artefato e isso é passado aos outros participantes. Numa aplicação principalmente coordenadora, há comunicação (implícita) quando um participante acaba uma atividade e habilita outro participante a começar a próxima atividade. O aspecto de **Comunicador** agrupa as funcionalidades que permitem diferentes usuários a se comunicarem explicitamente. Algumas funcionalidades do Comunicador são:

- Enviar e receber mensagem;
- Entrar ou sair de uma conferência;

Ellis e Wainer deixam claro que esses aspectos apontados por eles são apenas de caráter classificatório. As verdadeiras aplicações *groupware* possuem muitas vezes funcionalidades de dois ou mais aspectos.

O paradigma MOrFEu de desenvolvimento de ambientes virtuais (ver o Capítulo 4) se baseia no aspecto de Guardião de Artefato para propor uma organização diferente para ambientes virtuais. Ou seja, os outros dois aspectos, o Coordenador e o Comunicador, são concebidos como Guardiões de Artefatos. Por

exemplo, o processo de comunicação explícita entre dois participantes, pode ser visto como a criação de um artefato compartilhado, resultante da interação entre esses dois participantes. Ou seja, uma aplicação tradicionalmente comunicadora como um Chat pode ser vista com um guardião de artefato, onde o artefato (ou documento) é a sessão de conversa, que foi criado coletivamente pelos participantes do chat.

Ainda é interessante notar, no trabalho de Ellis e Wainer (2000), que um de seus desejos é a concepção de Guardiões especificáveis:

> Nos Guardiões está embutido (*embeded*) um modelo ontológico que normalmente é fixo e definido a priori pelo desenvolvedor do groupware. Mas seria interessante se os próprios usuários pudessem definir ou adaptar a ontologia do Guardião. Da mesma maneira que os sistemas workflow são Coordenadores especificáveis, há uma necessidade por Guardiões especificáveis.

Essa é exatamente a ideia central deste trabalho, fazer aplicações Guardiães de Artefatos especificáveis, partindo da ideia do MOrFEu que visualiza as aplicações Coordenadoras e Comunicadoras também como Guardiães. Ou seja, com um modelo formal de especificação desse tipo de aplicação é possível torná-los concretos e modificáveis em tempo de execução.

Além disso, essa especificação de ambientes permite que eles sejam descritos e comparados (ELLIS e WAINER, 1994). Além disso, é possível existir uma evolução dessas ferramentas, impulsionada por essas comparações e facilidades no seu desenvolvimento.

Ellis e Wainer (1994) ainda visualizam três componentes de *groupware*: modelo ontológico, modelo de coordenação e modelo de interface do usuário.

3.2 Buscando-se desenvolver Ambientes Virtuais

Dentre as técnicas atuais de desenvolvimento flexível de *groupware* e ambientes virtuais de aprendizagem (AVAs) estão aquelas baseadas em módulos, em componentes, em linha de produção de software e em *web-services*. Essas técnicas de desenvolvimento partem do ponto de vista do projetista de ambientes virtuais, que de

uma maneira mais flexível e ágil gera um novo ambiente, seguindo os requisitos de um grupo em particular. No entanto, seus projetistas sabem que:

> Mesmo que um desenvolvedor de groupware seja capaz de desenvolver uma aplicação 'ótima' para um grupo, ela [a aplicação] irá eventualmente se tornar inadequada devido a novas situações e problemas que certamente aparecerão (FUKS, RAPOSO, *et al.*, 2007).

Mas eles lidam com um problema mais imediato, que os motiva a adotarem tais soluções:

> Em face das dificuldades de construção e manutenção, o desenvolvedor de groupware gasta mais tempo lidando com dificuldades técnicas que moderando e provendo suporte para as interações entre os usuários. Tais problemas levam à necessidade de criar uma forma mais rápida e efetiva de se desenvolver groupware (FUKS, RAPOSO, *et al.*, 2007).

Um dos ambientes virtuais mais populares é o Moodle (DOUGIAMAS e TAYLOR, 2003), com mais de 50 mil sites registrados em 210 países (MOODLE.ORG, 2011). Esse AVA possui **desenvolvimento modular**. O núcleo de serviços básicos, configurações e cursos é indivisível. E suas ferramentas são instaladas como módulos. Várias ferramentas estão disponíveis por padrão, e a comunidade de desenvolvimento disponibiliza tantas outras. No entanto, a modificação de funcionalidades de alguma ferramenta é feita de maneira trabalhosa e requer acesso ao código-fonte e privilégios como administrador do sistema. E se for realizada uma modificação como essa, ela será válida para todo o ambiente. Ou seja, grupos pequenos não podem produzir ou alterar suas ferramentas sem que isso afete todos os usuários do ambiente.

Outra técnica utilizada é o desenvolvimento **baseado em componentes**. "A ideia principal dessas abordagens é encapsular muitas das dificuldades técnicas" (GADELHA, NUNES, *et al.*, 2009). A montagem desses componentes de acordo com as necessidades de um dado grupo determina o desenvolvimento do ambiente virtual para dar suporte a esse grupo. Os trabalhos de Gerosa *et al.* (2006), Won *et al.* (2006) e Beder *et al.* (2007) são exemplos de utilização dessa técnica.

Outro caso semelhante é o desenvolvimento **baseado em linha de produção** de software, no trabalho de Gadelha *et al.* (2009). Diferencia-se da abordagem baseada em componentes por tratarem o processo de desenvolvimento de uma maneira mais sistemática, diferente da maneira *ad hoc* da baseada em componentes. O processo cobre todas as etapas de desenvolvimento do *groupware*, desde o levantamento de requisitos.

Outra abordagem é a baseada nas tecnologias da própria Web, os ***web-services***, que são serviços (ou funções) hospedados na Web que encapsulam seu funcionamento interno, mas provêem uma descrição de suas entradas e da saída esperada de sua função, geralmente esta descrição é em WSDL (CHRISTENSEN, CURBERA, *et al.*, 2001). São exemplos de aplicação dessa abordagem os trabalhos de Pessoa *et al.* (2002) e Medeiros (2005). Eles geram ambientes virtuais compondo *web-services* disponíveis na Web, como *mashups*.

Vassileva *et al.* (2001) propuseram um ambiente virtual baseado em **agentes** que se assemelha à abordagem de *web-services*. Nesse ambiente, serviços ficam disponíveis num *pool* de serviços. Cada serviço – com funcionamento muito semelhante aos *web-services* – é gerenciado por um agente que se comunica com outros agentes formando uma comunidade. Um agente também cuida de cada usuário, provendo-lhe uma interface para o ambiente. Os agentes de usuários são orientados a objetivos, e emprestam os recursos gerenciados por outros agentes, negociando com eles. Os recursos são estáticos, e os agentes que gerenciam esses recursos precisam ser configurados para tal. Portanto, a tarefa de alterar as ferramentas consiste em encontrar (ou produzir) na Web outras ferramentas com as funcionalidades desejadas e programar um agente que gerencie este recurso.

Contudo, essas abordagens tratam o ambiente virtual como uma aplicação concluída, que o processo de mudanças nas necessidades dos usuários pode esperar pela reformulação das características do ambiente por parte dos desenvolvedores. Ou

seja, eventualmente a aplicação irá se tornar inadequada, e quando isso acontecer será necessário reformulá-la, recomeçando do projeto.

Paredes (2007) partilha da mesma preocupação deste trabalho: os ambientes virtuais devem ser flexíveis para atenderem aos requisitos de seus usuários. Como ele explica em sua tese de doutorado (p. 4):

> Os ambientes de interação virtual [ou apenas ambientes virtuais] também são condicionados pela natureza evolutiva das pessoas. As necessidades e anseios das pessoas evoluem com as sociedades, sendo o resultado dessa evolução refletida nos requisitos dos ambientes virtuais de interação que participam. Deste modo, os ambientes virtuais de interação evoluem com as necessidades de seus utilizadores, adaptando os seus requisitos de usabilidade e sociabilidade. [...] A regulação do ambiente deve refletir as necessidades de interação dos utilizadores a cada momento, adaptando-se de acordo com a evolução dos interesses e necessidades de seus utilizadores.

No entanto, Paredes está principalmente interessado nos requisitos de sociabilidade, lidando com a **regulação da interação** e a adaptabilidade: "As políticas sociais englobam a definição dos comportamentos dos utilizadores no ambiente, política de privacidade, segurança e liberdade de expressão". E para isso desenvolveu o modelo dos Teatros Sociais para regulação de ambientes virtuais.

Apesar de estar nitidamente preocupado com a flexibilidade dos ambientes virtuais, Paredes não trata de questões do dinamismo do ambiente e de suas ferramentas de comunicação/interação, apenas com seu modo de operação, ou seja, com as questões de regulação e coordenação.

3.2.1 O modelo dos Teatros Sociais

O modelo dos Teatros Sociais (PAREDES, 2007; PAREDES e MARTINS, 2010) é um modelo para regular interação social e controle de ambientes virtuais. Dentro desses ambientes, usuários se tornam atores, interpretando papéis bem definidos previamente em um contexto de interação virtual.

Metáforas de uma peça de teatro são usadas nesse modelo. Em uma peça, o roteiro define os papéis e as ações por eles realizadas, mas não fala diretamente do

ator que interpretará este papel. Quando um indivíduo interpreta um papel, ele é chamado de ator, e pode interpretar muito papéis em uma peça.

Os espaços onde ocorrem as interações foram chamados de Espaços Sociais e são equivalentes aos Veículos de Comunicação do MOrFEu. Logo, os Teatros Sociais podem ser definidos como um meta-ambiente para interação social, que vai ser instanciado em diferentes palcos, os Espaços Sociais, onde contextos reais de interação são virtualmente reproduzidos.

Os elementos básicos do modelo são os papéis, o fluxo de interação e as regras de interação.

Os **papéis** são interpretados por usuários (que passam a se chamar atores) em um Espaço Social. Um papel é caracterizado definindo-se: (1) sua identificação; (2) o número mínimo e o máximo de diferentes instâncias em permitidas em um Espaço Social; (3) uma escolha se o papel é designado antes ou em tempo de execução; (4) uma escolha se ele é estático ou pode ser modificado em tempo de execução; (5) se é obrigatório ou não; e (5) opcionalmente, um conjunto de sub-papéis que o usuário pode assumir em tempo de execução.

O **fluxo de interação** é equivalente ao roteiro de uma peça, ou seja, nele é definida a seqüências das ações a serem realizadas pelos atores. Ele pode ser definido por: (1) um conjunto de ações; (2) um conjunto de transações, guardadas por condições; e (3) um conjunto opcional de fluxo de dados.

E as **regras** são mecanismos para assegurar a regulação individual, relacionando papéis e ações definidas no fluxo de interação, especificando qual ator é permitido realizar tais ações.

3.3 Conclusão do Capítulo

Essas abordagens tratam o ambiente virtual como uma aplicação concluída, que o processo de mudanças nas necessidades dos usuários pode esperar pela reformulação

das características do ambiente por parte dos desenvolvedores. Ou seja, eventualmente a aplicação irá se tornar inadequada, e quando isso acontecer será necessário reformulá-la, recomeçando do projeto.

Essa é uma questão ainda mais crítica em domínios onde prevalecem ações dinâmicas fortemente baseadas na socialização de produção intelectual, como o domínio da educação, discutido no capítulo anterior.

No entanto, pontos positivos podem ser extraídos dessas abordagens.

Em ambientes virtuais centralizados, que funcionam em apenas um site, trazem algumas vantagens inerentes: centralização dos dados; possibilidade de processamento dos dados transversalmente às comunidades, quando há participantes em mais de uma comunidade; ferramentas de percepção (*awareness*) possuem maior poder de atuação; as diversas ferramentas de interação do ambiente possuem um mesmo padrão de interface, o que deixa seus usuários em uma situação mais confortável; a atualização para novas versões do software é mais fácil e pode ocorrer no sistema todo; etc.

Em um ambiente com desenvolvimento modular, seus módulos permitem a instalação fácil de novos tipos de ferramentas, assim como a atualização das existentes.

Uma característica interessante do desenvolvimento com componentes e linhas de produção de software é o reuso de código, que permite o desenvolvimento mais rápido de outras ferramentas semelhantes as já existentes.

Os ambientes baseados em *web-services* e agentes são distribuídos, ou seja, cada elemento do ambiente possui certo grau de independência e isolamento, o que é oposto aos ambientes centralizados. Sua característica mais interessante é sua independência, isso quer dizer que cada elemento possui sua própria estrutura e funcionamento diferenciado, as alterações feitas em um elemento não refletem no sistema como um todo, apenas das comunidades que fazem uso deste elemento. Assim, traz a possibilidade de modificações dos elementos de para grupos específicos,

ou até mesmo criação de elementos específicos para certas comunidades de interesse. Além disso, nos ambientes baseados em agentes há um infra-estrutura para suporte de software inteligente.

Dessa forma, algumas características desses ambientes virtuais são interessantes e respondem a alguns requisitos daqueles levantados no capítulo anterior.

R1. **Os usuários devem ser capazes de construir suas próprias ferramentas**: somente as abordagens de *web-services* deixam essa funcionalidade mais próxima do usuário, no entanto ainda é necessário apoio especializado para se fazer essas construções;

R2. **Os usuários devem ser capazes de fazer alterações nas aplicações conforme sua necessidade e em tempo de execução**: somente a abordagens dos teatros sociais explora esse requisito, através de suas definições declarativas de suas regras de negócio;

R3. **As aplicações devem ter possibilidade de serem compostas por dados de outras aplicações**: as abordagens de *web-services* são notadamente voltadas para esse atender a esse requisito, e as outras abordagens não foram projetadas para lidar com esse problema;

R4. **A produção de conteúdo deve ser independente de onde ele vai ser utilizado, ou disponibilizado**: nenhuma das abordagens citadas cumpre esse requisito;

R5. **As máquinas devem ser capazes de analisar os dados contidos nessas aplicações**: as abordagens de agentes são projetadas para dar suporte às funcionalidades que atenda a esse requisito;

Na Tabela 2, apresenta-se um resumo comparativo das abordagens de desenvolvimento de ambientes virtuais flexíveis. Adiantadamente, compara-se as abordagens relatadas nesse capítulo com o paradigma inovador MOrFEu e com a proposta deste trabalho, o MX. Nota-se que a proposta deste trabalho, o MX, trata de várias deficiências das abordagens antecessoras.

Tabela 2. Comparativo entre as abordagens de desenvolvimento de ambientes virtuais flexíveis.

	Módulos	Componentes	Web-Services / Agentes	MOrFEu	**MX**

Construção de ambientes personalizados	Não	Sim	Limitado	Sim	Sim
Construção de ambientes com protocolo complexo	Não	Sim	Não	Limitado	Sim
Inclusão de nova funcionalidade	Limitado	Não	Limitado	Sim	Sim
Alteração de funcionalidade	Não	Não	Sim	Sim	Sim
Especificação de funcionamento	Não	Não	Não	Não	Sim
Suporte a serviços externos	Não	Possivelmente	Sim	Não	Sim (com agentes)
Prevê o uso de agentes	Não	Possivelmente	Possivelmente / Sim	Não	Sim

4 MOrFEu: um Paradigma Inovador de Ambientes Virtuais

Menezes *et al.* (2008) propuseram uma nova maneira de se pensar ambientes virtuais, a essa proposta foi dado o nome de MOrFEu, um acrônimo para "Multi-Organizador Flexível para Espaços Virtuais". A construção desses ambientes está orientada às produções individuais e coletivas de seus usuários. A busca por essa nova abordagem está pautada nos seguintes princípios: plasticidades, ergonomia, redução do trabalho repetitivo e redução da sobrecarga cognitiva. Trata-se de um projeto multi-institucional com várias frentes de trabalho. Neste trabalho, parte-se da abordagem do MOrFEu para se criar uma maneira de se especificar ambientes virtuais.

"Morfeu", a palavra se refere ao deus romano dos sonhos, que tinha a habilidade de se "moldar" em diferentes formas nos sonhos das pessoas (DICT.ORG). Essa habilidade de "se moldar" também é a principal característica dos ambientes criados com o MOrFEu. Ou seja, esses ambientes virtuais podem ser moldados de diferentes formas para que atendam às necessidades de seus usuários, tanto com respeito às interações quanto à organização de suas produções intelectuais. Isso quer dizer que os espaços virtuais podem ser criados de acordo com a necessidade de seus usuários e que eles podem ser modificados em tempo de execução, de acordo com a mudança das necessidades de seus usuários.

Centrado nas produções de seus usuários, os ambientes virtuais MOrFEu são criados e organizados a partir dessas produções, em contraste com os ambientes tradicionais que organizam essas produções de seus usuários de acordo com suas ferramentas de comunicação/interação que possuem. Ou seja, os ambientes virtuais

MOrFEu são organizadores e compartilhadores das produções individuais de seus utilizadores. A essas produções deu-se o nome de **Unidades de Produção Intelectual (UPI)**.

Essa ideia é bastante semelhante à ideia do "Guardião do Artefato" (*artifact keeper*) de Ellis e Wainer (2000). Ocorre quando a colaboração entre um grupo de pessoas é centrada no acesso e modificação de um conjunto de dados compartilhados, que foi chamado de artefato. O Guardião do Artefato é o conjunto de funcionalidades relacionadas com a manipulação desse artefato (veja a Seção 3.1 para maiores detalhes). Esse artefato é chamado no MOrFEu de **documento**.

Dessa forma, um ambientes virtual MOrFEu possui um **documento** onde são organizadas essas produções de seus utilizadores. Esse documento é um artefato criado coletivamente, formado pelas produções de seus usuários. Por exemplo, uma aplicação tradicionalmente comunicadora como um Chat pode ser vista com um guardião de artefato, onde o artefato (ou documento) é a sessão de conversa, que foi criado coletivamente pelos participantes do chat.

O ato de incluir uma produção nesse documento foi chamado de **publicar**, ou seja, tornar público ao grupo de usuários do ambiente.

São necessárias regras de como esse documento deve ser composto. Ao conjunto de regras de como esse documento pode ser gerado foi chamado de **protocolo de interação**.

A visualização desse documento merece especial atenção, pois na maioria das vezes não é possível ver a totalidade do documento de uma vez só. Para isso, existem os chamados **templates**, que geram visualizações de partes do documento do ambiente virtual.

Na Figura 2, está representado o funcionamento de um ambiente virtual no paradigma MOrFEu, chamado de Veículo de Comunicação (VCom). Um VCom é usado para interação entre pessoas. Ele organiza UPIs que são publicadas através de um protocolo de interação. E esse VCom é acessado através de templates.

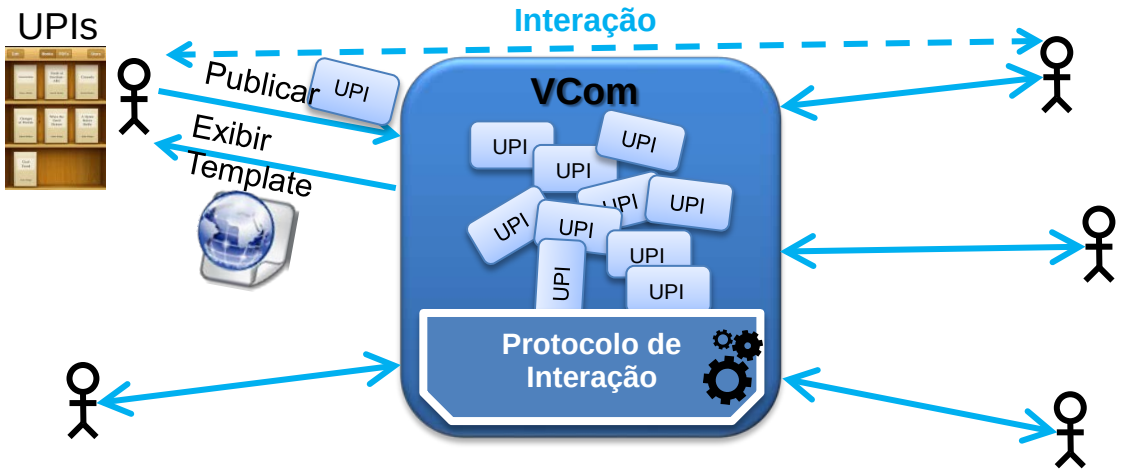


Figura 2. Funcionamento de um VCom, dando suporte à interação de seus usuários.

O paradigma MOrFEu não busca apenas a criação de ambientes inovadores construídos a partir de suas ideias. Ele defende que também é possível pensar os ambientes existentes atualmente da forma como é proposto.

Os diversos trabalhos no âmbito do projeto MOrFEu exploram cada um de seus elementos e tentam desenvolver a ideia como um todo.

Este trabalho encontra-se no âmbito do projeto MOrFEu, porém não busca desenvolver suas ideias como principal objetivo. A partir das ideias desse projeto, busca-se o desenvolvimento de uma solução para o problema proposto. Que é o desenvolvimento de ambientes virtuais flexíveis que atendam às necessidades por interação de seus usuários.

4.1 Autoria

Uma concepção relevante do sistema é o suporte à autoria (individual e coletiva), à publicação e à socialização das produções intelectuais, de acordo com contextos e estruturas já existentes ou estabelecidas *a posteriori*, rompendo com a organização convencional dos ambientes virtuais tradicionais que organizam os espaços de trabalho a partir das ferramentas de comunicação/interação que serão utilizadas.

Isso se assemelha às ideias de Berners-Lee (2000, p. 160) quando escreve: "Cada vez que eu escrevo alguma coisa com o computador, eu tenho que escolher se abro a aplicação de 'correio eletrônico', a aplicação 'net news', ou a aplicação 'editor Web'".

No contexto de ambientes virtuais, a situação é parecida, toda a produção[12] intelectual do indivíduo está atrelada ao uso de uma ferramenta, específica. Primeiro a pessoa deve escolher onde gostaria de socializar uma dada produção para depois produzi-la. Por exemplo, se uma pessoa deseja escrever uma resenha sobre um artigo, ela deve saber com antecipação onde esta resenha será publicada, ou seja, em qual ferramenta ela ficará "armazenada", se em um fórum sobre aquele artigo, em uma mensagem para outra pessoa, em um blog, etc.

Assim, é apresentado o conceito de **Unidade de Produção Intelectual** (UPI) que é o artefato básico de suporte à autoria, destinado a registrar as produções individuais. O tipo básico de uma UPI são textos escritos em HTML e, portanto, podem usar diferentes mídias (imagens, sons, vídeos, etc.) e referenciar, através de links, outras UPIs ou URLs. Outros tipos possíveis são: imagens, textos sem formatação, vídeos, documentos XML, partituras musicais, códigos-fonte em alguma linguagem de programação, etc.

Dessa forma, as UPIs existem independentemente dos lugares onde elas são socializadas (ou publicadas). Ou seja, uma mesma UPI pode ser publicada em mais de um lugar (ou ferramenta ou ambiente virtual).

Assim, a pessoa é proprietária daquilo que produz e possui o controle sobre isso, podendo organizar da forma que desejar na sua "prateleira" de UPIs (Figura 2). Contrastando com o paradigma atual: quando uma pessoa utiliza uma ferramenta de comunicação/interação, sua produção fica atrelada a essa ferramenta (Figura 3).

[12] Entende-se por produção qualquer documento escrito por um usuário, seja uma mensagem trocada em um email, uma mensagem instantânea escrita em um chat, uma resposta em um fórum, um comentário em um blog, um artigo curto em um blog etc.

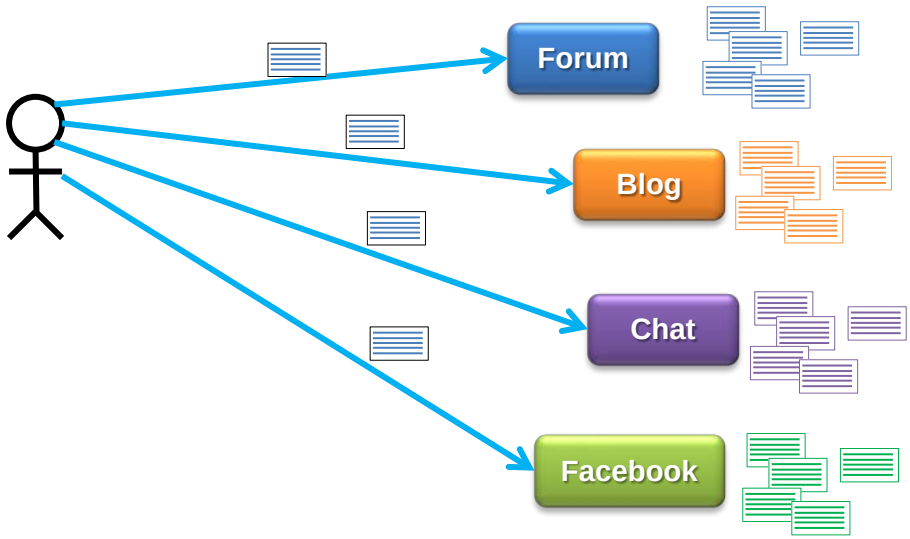


Figura 3. Forma tradicional em que as produções são gerenciadas nos ambientes virtuais tradicionais.

4.2 Publicação, Espaços Virtuais e Documentos

Publicação é o ato de socializar (ou postar) uma produção em um espaço virtual. Um elemento central para a modelagem dos ambientes virtuais com o MOrFEu é o que foi chamado de **Veículo de Comunicação** (VCom). Qualquer espaço de trabalho, voltado para comunicação, interação e produção, é modelado a partir desse conceito.

Um VCom é definido como um ambiente virtual para produção cooperativa de um dado grupo de usuários. Os VCom são as estruturas responsáveis por realizar qualquer necessidade de interação entre os usuários do ambiente. Na prática, o termo VCom é atribuído a qualquer ambiente virtual criado com o MOrFEu.

Pode-se modelar um espaço colaborativo a partir do conjunto de interações que resultará do processo de colaboração. Esse espaço pode ser representado por um **documento** hipermidiático que agrupa o produto dessas interações. Parte-se então do documento resultante dessa cooperação. O documento possui uma estrutura organizacional. Cada componente dessa estrutura possui uma equipe de produção. E o desenvolvimento ocorre segundo um conjunto de diretrizes de produção, o

Protocolo de Interação. O artefato resultante é um VCom, e é composto por Unidades de Produção Intelectual (UPIs). Portanto, os VComs são os espaços destinados à autoria coletiva.

As principais ferramentas de comunicação/interação hoje utilizadas na Web podem ser descritos como VComs. A seguir, são apresentados alguns exemplos:

- **Chat**: uma conversa por chat resultará em um documento final produzido pelos participantes de uma sessão de conversa. Neste documento, em geral, as postagens são realizadas uma após a outra, sem qualquer ordem predefinida de intervenção, feitas de forma síncrona.
- **Blog**: o autor publica várias postagens (UPIs) organizadas pela data de publicação. A cada uma dessas postagens cabem comentários (UPIs) de outras pessoas, que se tornam co-autores do documento. As postagens em um blog são, em geral, longas. As mensagens podem também ser curtas (micro-blog[13]) e os comentários podem virar um debate em forma hierárquica, às vezes.
- **Fórum**: as produções (UPIs) estão organizadas em forma de árvore, onde uma UPI "responde" a outra UPI ou inicia um novo ramo de publicação, de forma hierárquica. Logo, o documento desse VCom é o conjunto de todas as produções organizadas em forma de árvore. É possível ter fóruns com limitação do nível de profundidade da discussão, ou imaginar um fórum realizado sincronamente.
- **Mural**: um mural é uma coleção de postagens (UPIs), sem réplica, com prazo de expiração, postados de forma assíncrona.
- **Wiki**: as pessoas criam páginas (UPIs) que podem ser editadas a qualquer momento. Estas páginas possuem links para outras páginas, que são a única forma de publicar novas páginas. As páginas mais os links entre elas formam o documento deste VCom.

4.3 O que o MOrFEu não é

O MOrFEu é um paradigma para desenvolvimento de ambientes virtuais para a Web, ele não trata de outras classes de ambientes virtuais, como os ambientes de

[13] Exemplo: http://twitter.com

realidade virtual. Também não se propõe a representar outros sistemas de informação que não tenham as características de ambientes virtuais: sistemas na Web que tem propósito de apoiar a interação entre seus usuários.

4.4 Arquitetura de Desenvolvimento de Veículos de Comunicação

Foi desenvolvido um meta-ambiente virtual que gerencia outros ambientes virtuais nele contidos, os Veículos de Comunicação (VCom). Neste capítulo, é proposta um arquitetura para os VComs de maneira que eles atendam aos requisitos de flexibilidade.

Model-View-Controller (MVC) é um padrão de arquitetura de software destinado a aplicações com alto nível de interatividade, cujo objetivo é melhorar a usabilidade da aplicação, pois permite que a interface se ajuste (possivelmente em tempo de execução) independemente do seu núcleo de funcionalidades (BUSCHMANN, MEUNIER, *et al.*, 1996). Como os ambientes virtuais são aplicações web interativas (no sentido aplicação-usuário), decidiu-se utilizar tal tipo de arquitetura para se projetar o protótipo do meta-ambiente MOrFEu. Naturalmente, nessa arquitetura pode-se notar certo grau de flexibilidade no quesito interface, pois o MVC é destinado a aplicações capazes de serem modificadas suas interfaces facilmente.

No entanto, ainda há a necessidade de flexibilidade com respeito ao model e ao controller, onde fica o núcleo das funcionalidades. Isso será abordado nas próximas seções, a seguir.

Primeiramente, são expostos os conceitos gerais desses três componentes desse padrão de arquitetura (BURBECK, 1987):

- **Model**: gerencia o comportamento e os dados do domínio da aplicação, responde a solicitações de informações sobre seu estado (geralmente pela view), e responde a instruções de mudança de estado.

- **View**: gerencia a saída gráfica/textual para a porção da tela alocada para a aplicação. No caso de aplicações web, trata-se daquilo que é visto no browser do usuário.
- **Controller**: interpreta as entradas de dados do usuário pelo mouse e teclado, comandando modificações ao model e/ou à view conforme o caso.

Na Figura 4, é apresentado um diagrama que resume as funções desses três componentes. O usuário visualiza o conteúdo gerado pela view e gera entrada de dados que são processados pelo controller. O controller, por sua vez, processa e decide o que fazer com essa entrada: atualiza a view, atualiza dados no model e requisita a visualização de outra view.

É importante salientar que para um mesmo conjunto de dados é permitido existir inúmeras views, o que permite visualizações diferentes desses dados. Por exemplo, um mesmo conjunto de dados presentes em uma planilha pode ser visualizado de diferentes formas: em uma tabela, em um gráfico de linhas, em um gráfico de pizza, um histograma, etc.

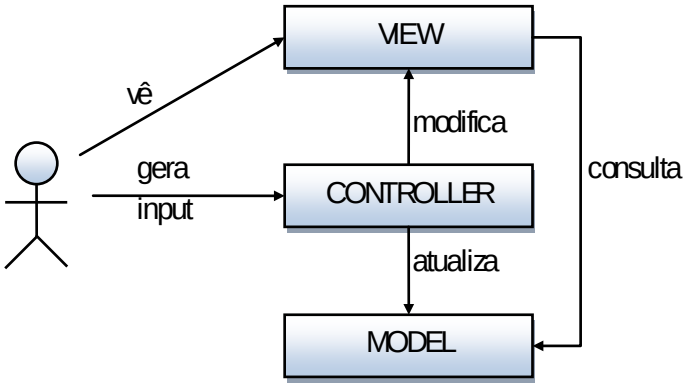


Figura 4. Esquema do padrão de arquitetura MVC.

A seguir, é descrito qual o papel de cada um desses três componentes na arquitetura de Veículos de Comunicação do MOrFEu.

4.4.1 Model

Um dos princípios do MOrFEu é a flexibilidade, assim, seus componentes devem ser flexíveis. Portanto, o model de um VCom precisa ser maleável, ou seja, seu

esquema de dados não pode ser fixo. De modo que seja possível adicionar novos (ou alterar antigos) tipos de dados no model.

É óbvio que o tipo de banco de dados tradicional, o relacional, não é muito adequado para essa tarefa, pois as modificações de esquema não são eficientes, mesmo que em alguns casos seja suportada. Assim, os bancos de dados chamados NoSQL são os mais recomendados, mais especificamente aqueles que são *schema-free* (com esquema de dados livre). Alguns exemplos são: aqueles baseados em XML, como eXist[14] e Sedna[15]; e os baseados em documentos, como CouchDB[16], MongoDB[17] e SimpleDB[18].

4.4.2 View

As views geram aquilo que é visto pelo usuário. Tratando-se de ambientes virtuais na web, as views de um VCom devem ser programas/scripts que gerem páginas HTML, usando os dados presentes no model. Diversas views podem atuar sobre um mesmo conjunto de dados. Por exemplo, um mesmo conjunto de dados presentes em uma planilha pode ser visualizado de diferentes formas: em uma tabela, em um gráfico de linhas, em um gráfico de pizza, histograma, etc.

É óbvio que cada tipo de model possui sua maneira diferente de acessar os dados, assim, as views devem ser compatíveis com o model escolhido. Por exemplo, se o model é implementado com MongoDB, pode-se escolher a linguagem PHP para se implementar as views, já que ela possui um suporte nato para HTML e o MongoDB possui *drivers* para essa linguagem. No entanto, a escolha que foi feita para implementação do protótipo foi um bancos de dados XML com processador de views em XSLT.

[14] http://exist.sourceforge.net/
[15] http://www.sedna.org/
[16] http://couchdb.apache.org/
[17] http://www.mongodb.org/
[18] http://aws.amazon.com/simpledb/

4.4.3 Controller

O controller é elemento responsável por gerenciar as regras de negócio da aplicação, altera os dados e aplica uma série de restrições próprias da aplicação. Em aplicações web colaborativas, além das tradicionais ações de manipulação dos dados, são necessárias algumas funções de regulação social (PAREDES e MARTINS, 2010), também chamadas de funções de coordenação (FUKS, RAPOSO, *et al.*, 2007).

Ações de manipulação de dados podem ser resumidas pelo acrônimo em inglês CRUD (*Create, Read, Update and Delete*), que significa criar, ler, atualizar e apagar. Essas operações são realizadas tanto no model quanto no banco de UPIs, e incluem também ações de publicação de UPIs no model.

Além disso, há ações de publicação. Publicar quer dizer referenciar UPIs em VComs, e pode ocorrer de duas formas, no momento da publicação pode ser pedido ao usuário criar uma nova UPI ou escolher uma UPI sua já existente. Dessa forma, pode-se resumir as ações de manipulação de dados de um VCom através da sigla CRUDP ("P" para *Publish*).

4.4.4 UPIs

Além dos três elementos tradicionais da arquitetura MVC, há o banco de Unidades de Produção Intelectuais (UPIs). As UPIs encontram-se separadas do model do VCom devido à proposta do MOrFEu de que as UPIs existem independentemente de onde elas serão publicadas, devendo estar disponíveis para publicação em todos os VComs.

Assim, existe um banco de UPIs que é gerenciado pelo sistema MOrFEu. E no model do VCom são feitas referências (publicações) às UPIs que o compõem.

4.4.5 Espaços Individuais

Como em qualquer ambiente virtual, o MOrFEu possui espaços individuais e coletivos. Os espaços coletivos são os Veículos de Comunicação, e sua arquitetura básica foi descrita nas seções anteriores. Porém há ainda espaços pertencentes a cada usuário.

Nesses espaços, o usuário pode gerenciar suas produções, quais UPIs produziu e onde as publicou. Além disso, operações básicas também devem estar disponíveis, como alteração de perfil e senha, e consulta de logs de acesso próprios.

4.4.6 Resumo da Arquitetura

Foi descrita uma arquitetura MVC para desenvolvimento de veículos de comunicação. Um dos pontos a serem ressaltados é que o model deve suportar esquema variável em tempo de execução e as views devem ser compatíveis com o model escolhido. O controller, por sua vez, trata de ações de manipulação dos dados e de regras de interação social. Além disso, as UPIs são elementos transversais aos VComs, podendo coexistir em vários ambientes. Na Figura 5, é apresentado um diagrama com o resumo das principais funcionalidades de cada um dos componentes da arquitetura e as relações entre eles.

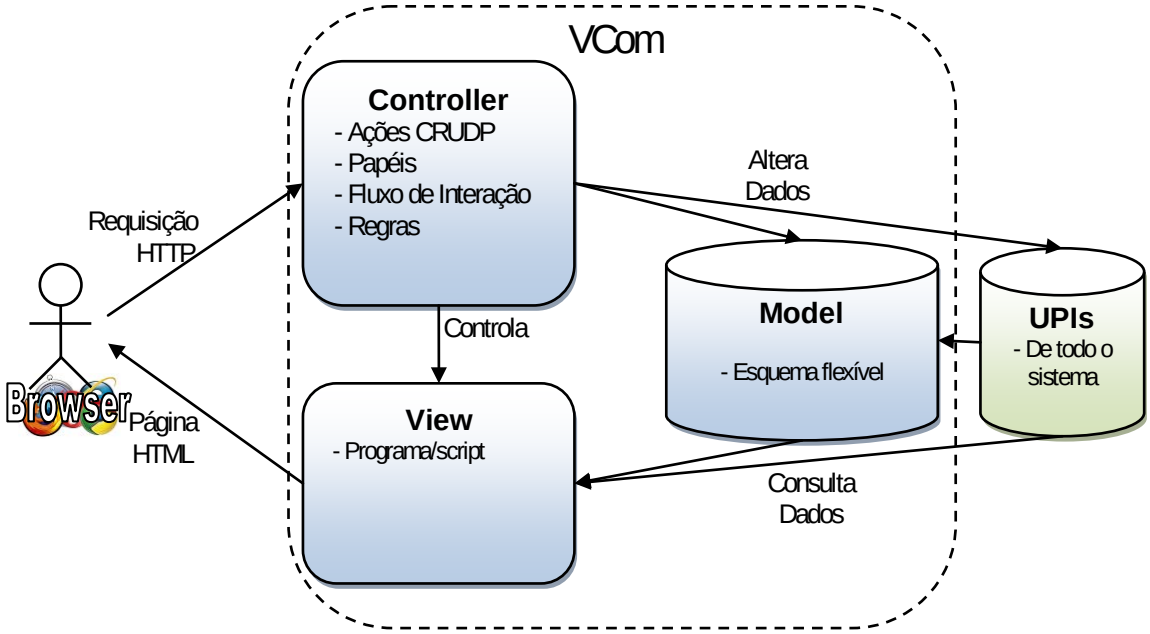


Figura 5. Arquitetura MVC dos Veículos de Comunicação do MOrFEu.

4.5 Meta-ambiente para Desenvolvimento de Veículos de Comunicação

Foi implementado um meta-ambiente virtual para criação de veículos de comunicação (VCom). Suas principais funcionalidades são: criação e gerenciamento de UPIs; criação e edição de VComs; uso de um VCom, incluindo publicações de UPIs.

O esquema de dados do sistema inclui usuários, UPIs, versões de UPIs e VComs. Na Figura 6, é apresentado o esquema de dados desse protótipo. Nota-se sua simplicidade, com apenas quatro elementos. O gerenciador de banco de dados utilizado foi o MySQL.

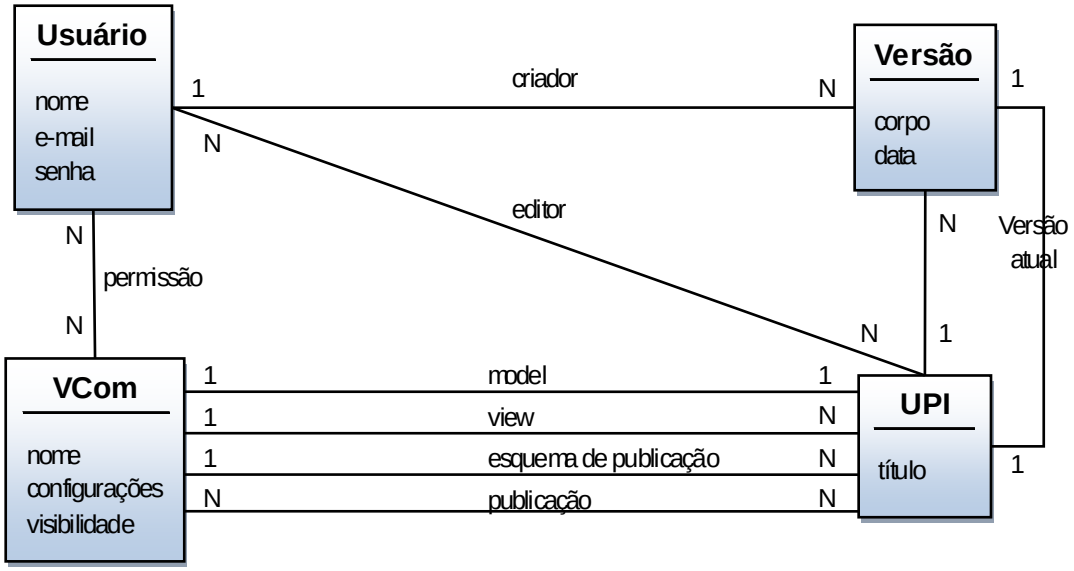


Figura 6. Esquema de dados do protótipo do MOrFEu.

Cada UPI é composta por versões, que guardam seu conteúdo. Sempre que ocorre uma edição de uma UPI é criada uma nova versão, e as anteriores ficam armazenadas. Vários usuários podem editar uma UPI, mas apenas um usuário é autor de cada versão.

Um VCom é uma composição de UPIs. O seu model é um arquivo XML armazenado em uma UPI. E ele pode possuir várias views, que são arquivos XSLT, também armazenados como UPIs. Para se realizar a publicação são usados esquemas de publicação, que são fragmentos do XML que será inserido, e as UPIs publicadas.

Além disso, há VComs padrões. A criação (ou instanciação) de um VCom se dá pela cópia de um dos VComs padrões pré-definidos. Esses modelos para cópia possuem o model inicial (geralmente vazio de dados, mas com estrutura), as views, e os esquemas de publicação.

Para o desenvolvimento rápido de um protótipo, foi escolhido o framework de desenvolvimento Yii (Yii, 2011). Ele é um framework PHP com arquitetura MVC que gera códigos-fontes bases a partir do esquema do banco de dados passado. Ele possui diversas funcionalidades já implementadas como acesso ao banco de dados e tratamento de login de usuários, o que agiliza o desenvolvimento de aplicações web.

A escolha desse framework não tem relação direta com a implementação do MOrFEu, foi apenas uma escolha técnica para agilizar o desenvolvimento.

Para implementação do model foram utilizados arquivos no formato XML. Essa escolha foi feita pelo fato do XML ser uma linguagem de marcação de dados que permite o armazenamento de dados de maneira semi-estruturada e com esquema flexível. Haviam outras alternativas, mas o XML se mostrou mais simples de implementar e de ser entendido por humanos. Esses arquivos XML também são produções e foram tratados como UPIs, de um tipo especial.

Para as views foram utilizadas folhas de estilo XSLT. Como os models são arquivos XML, é natural processá-los com XSLT para exibição destes em outro formato, no caso, em HTML. A linguagem XSLT possui repetições, condicionais, variáveis e funções, e é uma linguagem Turing Completa (KEPSER, 2004). As folhas de estilo tomam o documento em XML, o processam e o exibem em outro formato, podendo computar dados e omitir parte dos dados presentes no XML original.

No entanto, o controller não foi totalmente implementado. Somente foram implementadas as funções básicas de manipulação dos dados do model e de publicação (CRUDP) e manipulação de views. As funcionalidades implementadas até agora são resumidas na Figura 7.

Os papéis básicos agora existentes são os autores e os leitores de um VCom. Ao autores são aqueles a quem é permitido gerenciar o VCom e suas estruturas internas. Aos leitores não é permitida essa alteração de parâmetros do VCom, daí o nome. No entanto, pode ser permitido a eles publicar em VComs normalmente.

O controller faz o processamento e execução das views. Na Figura 8, é apresentado o processo realizado para exibir uma view. São carregados o model e a view, e é executado o processador de XSLT do PHP. Algumas vezes a view pode requisitar informações de UPIs e usuários, que estão contidas no banco de dados.

Outra responsabilidade do controller é o processamento de publicações, representado na Figura 9. No momento de publicação de uma nova UPI, o usuário

preenche um formulário de UPI com título e corpo de texto, e essas informações são passadas para o processador de publicação, juntamente com o local dessa publicação, definido no link de publicação na view. A nova UPI, então, é inserida no banco de dados e o model é atualizado.

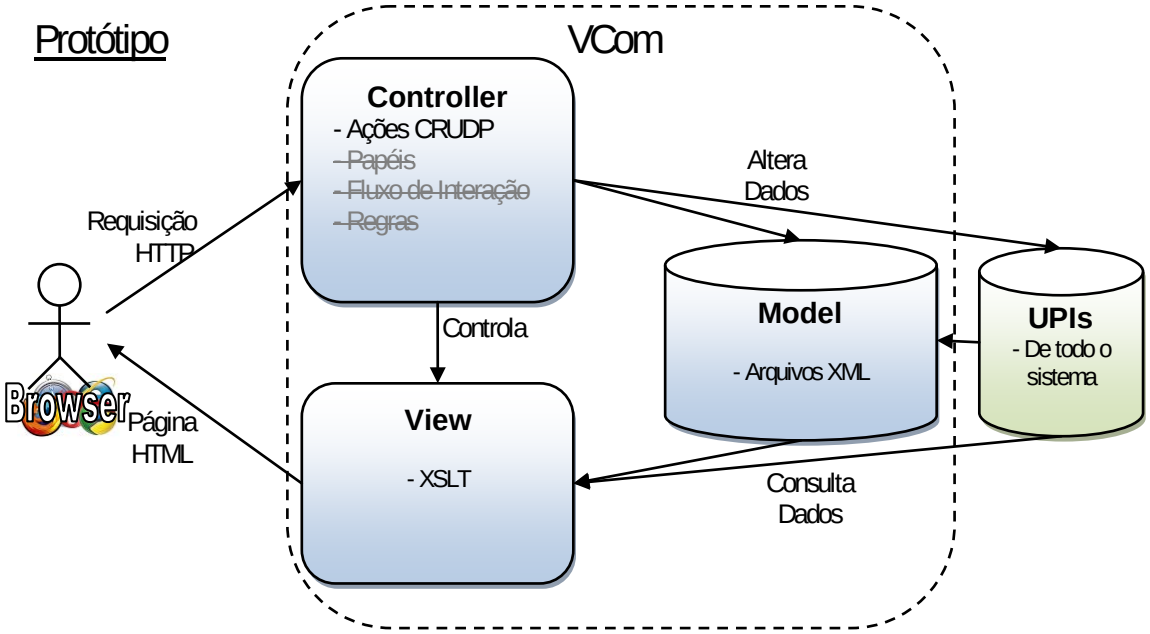


Figura 7. Arquitetura MVC do Protótipo do MOrFEu.

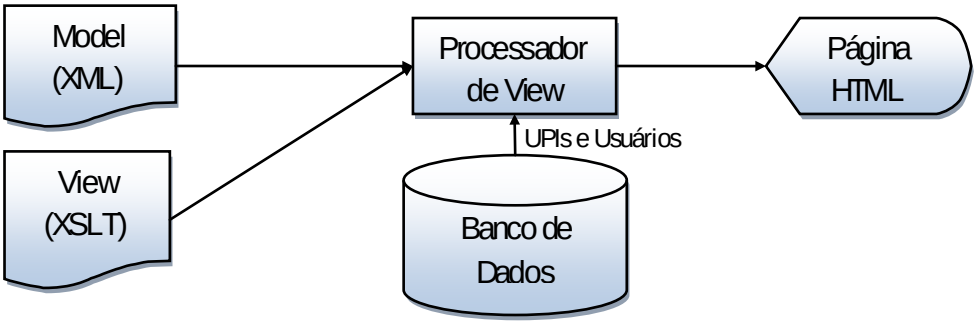


Figura 8. Processamento de uma view.

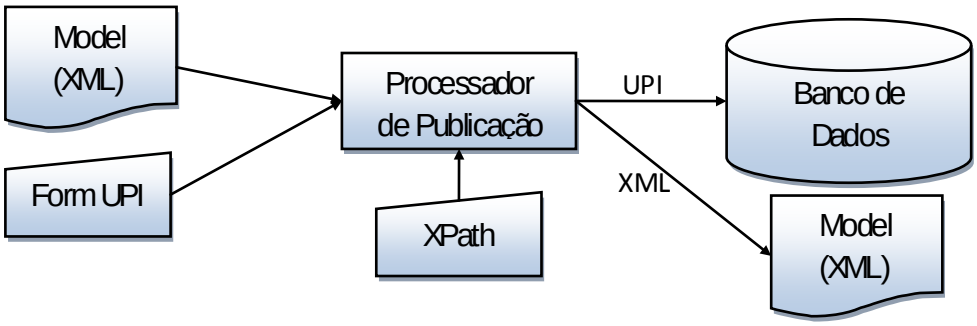


Figura 9. Processamento de uma publicação.

4.5.1 Ilustrando o Ciclo de Desenvolvimento de um Veículo de Comunicação

Nesse meta-ambiente, o desenvolvimento de um Veículo de Comunicação (VCom) começa pela modelagem dos dados, seguido de uma modelagem das páginas e da navegação. O objetivo da modelagem é nortear o desenvolvimento da aplicação, pois a aplicação pode ser desenvolvida incrementalmente, à medida que as necessidades surgem. Ou seja, este processo sugere um desenvolvimento em espiral.

Será utilizado como exemplo o desenvolvimento de uma ferramenta de apoio à arquitetura pedagógica "Construindo Conceituações" (NEVADO, DALPIAZ e MENEZES, 2009) (ver Seção 2.3.3). Esta ferramenta possui muitas das funcionalidades que esse primeiro protótipo do MOrFEu disponibiliza. Exemplos de outras ferramentas serão mostrados para completar a descrição de suas funcionalidades.

4.5.1.1 Modelagem dos Dados (Model)

Na implementação do MOrFEu, foi feita a escolha da XML para representar e armazenar os dados, pois é uma linguagem que pode modelar dados com esquema flexível. Este tipo de flexibilidade é desejável pelo MOrFEu, porque além de poder se criado diversos esquemas de dados, pode-se acrescentar e/ou alterar esses esquemas com facilidade.

Jackson (1975) propôs um diagrama de caixas e setas para representar algoritmos e programas, a linguagem JSP – Jackson Structured Programming. Diagramas semelhantes podem ser feitos para descrever dados hierárquicos como aqueles que são guardados por XML. Uma seta de A para B significa "A é composto de B"; asterisco, "zero ou mais"; e interrogação, "zero ou um". Este diagrama pode, assim, facilmente ser traduzido para DTD ou XML Schema, que definem qual o

formato aceitável para arquivos XML. Na Figura 10, é apresentado um diagrama no estilo JSP e uma instância de documento XML equivalente ao diagrama.

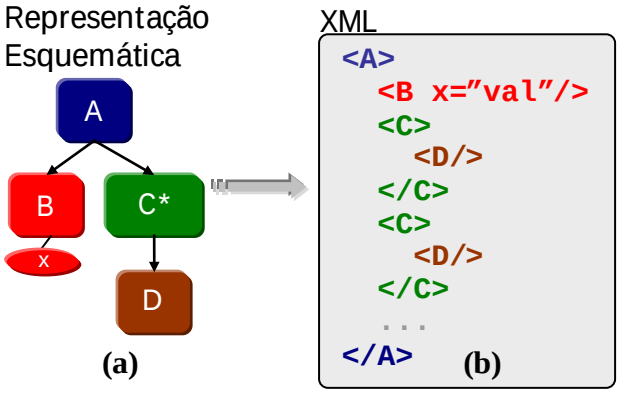


Figura 10. (a) Representação Gráfica da Estrutura de um VCom. (b) Tradução da linguagem gráfica para XML.

Na Figura 11, é apresentado o diagrama em JSP do esquema de dados da ferramenta de apoio à arquitetura pedagógica construindo conceituações, chamado aqui de "quadro de discussão". Houve diferenciação de nomes, pois trata-se apenas de uma ferramenta que apoia a arquitetura pedagógica, e essa ferramenta disponibiliza os quadros onde são feitas as discussões.

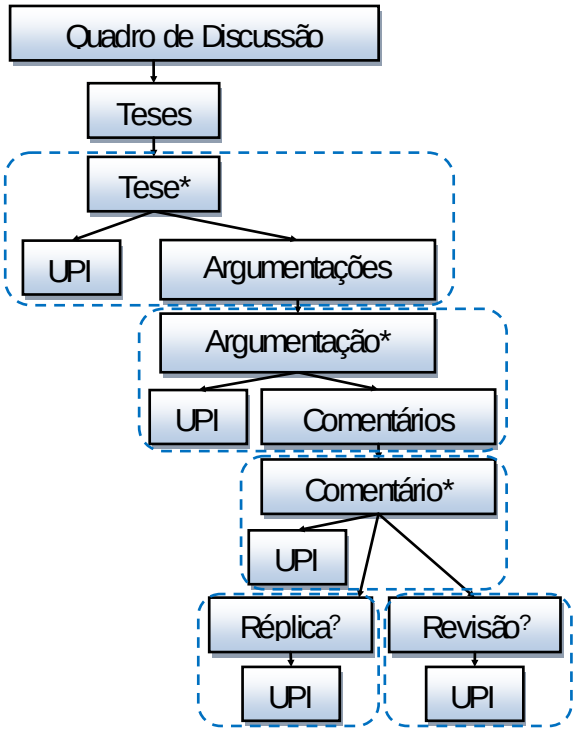


Figura 11. Diagrama do esquema de dados da ferramenta de apoio à arquitetura pedagógica Construindo Conceituações usando diagrama JSP.

Além disso, é possível representar publicações. Uma publicação significa: incluir um pedaço de dados XML em um arquivo XML existente. Esses pedaços foram chamados de "esquemas de publicação". Assim os dados são inseridos dentro do banco de dados do VCom, que é o arquivo XML em questão. No diagrama da Figura 11, esse esquema é representado com linhas pontilhadas delimitando cada esquema de publicação.

No quadro de discussão (ver Figura 11), observa-se que ele é composto por teses. Cada tese é composta por uma UPI que guarda o texto dessa tese e por argumentações inicias, referente à etapa 2 da arquitetura pedagógica. E assim por diante, com comentários para cada argumentação, seguidos nas etapas seguintes de réplicas e da revisão do posicionamento inicial.

As caixas que não são contornadas por linhas pontilhadas representam os "dados iniciais" do VCom. Assim, quando um quadro de discussão é criado ele possui o seguinte arquivo XML: `<quadrodiscussao><teses/></quadrodiscussao>`. E o esquema de publicação de tese é: `<tese><upi/><argumentações/></tese>`.

Uma publicação sempre é feita através de um esquema de publicação. E se esse esquema possuir a tag `<upi/>`, o sistema cria uma nova UPI e solicita ao usuário preenchê-la. O que fica salvo no XML é apenas um identificador daquela UPI através de um atributo da tag. As UPIs ficam armazenadas num banco de dados tradicional e estão sujeitas ao controle de versões, sempre que uma alteração é feita em alguma UPI é gerada uma nova versão e a antiga é mantida inalterada.

Um fórum de discussão possui um recurso proveitoso, a definição das postagens é feita de forma recursiva: uma postagem é feita em resposta a outra postagem. Na Figura 12, é apresentado o esquema de dados de uma ferramenta de fórum, a caixa com linha dupla é uma chamada recursiva ao elemento postagens definido anteriormente. Assim, uma postagem é formado por uma UPI e por respostas, que também são postagens.

Ainda é possível, no momento da publicação, atribuir dados adicionais além das UPIs, em forma de atributos das tags. Por exemplo, em um wiki foi atribuído um atributo com o nome da página, que é a chave para o link com outras páginas (Figura 13). Relações entre entidades que, em bancos de dados entidade-relacionamento, normalmente são N-para-N podem ser implementados utilizando-se esses atributos.

No VCom para apoio básico a projetos de aprendizagem (Figura 14) que foi construído, há uma característica interessante, os sub-VComs:

- "Desenvolvimento do Projeto" é um sub-VCom wiki simples;
- "Diário de bordo" é um blog;
- "Fórum de orientação"; e
- "Livro de Visitas" é um mural.

Com esse elementos é construído o esquema de dados de um VCom: tags, atributos, relações hierárquicas entre tags, repetições, chamadas recursivas, links entre itens e esquemas de publicação.

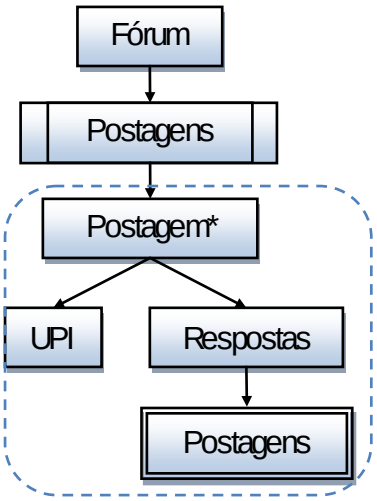


Figura 12. Diagrama do esquema de dados de uma ferramenta de fórum.

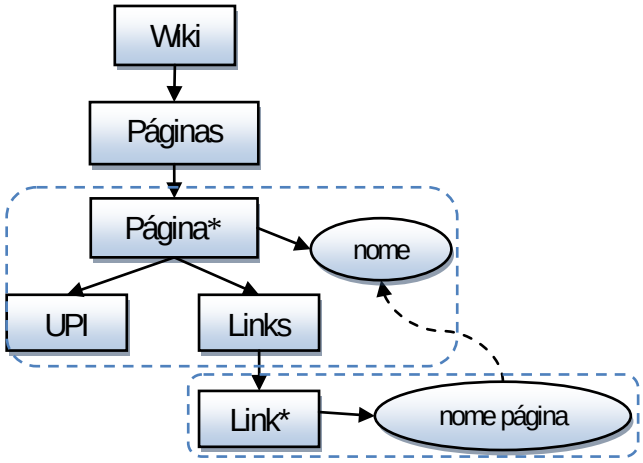


Figura 13. Diagrama do esquema de dados de uma ferramenta de wiki.

4.5.1.2 Modelagem da Navegação

Como se trata de uma aplicação web, é interessante modelar a navegação entre as páginas dessa aplicação. O quadro de discussão necessita de poucas páginas em sua versão original: uma página para o quadro de discussão de algum usuário; uma lista de participantes que possui links para os quadros desses participantes; e uma lista de quadros que o usuário está interagindo. Uma representação simples é exibida no diagrama da Figura 15. Assim, sabe-se que é preciso criar três views, um para cada página nesta modelagem.

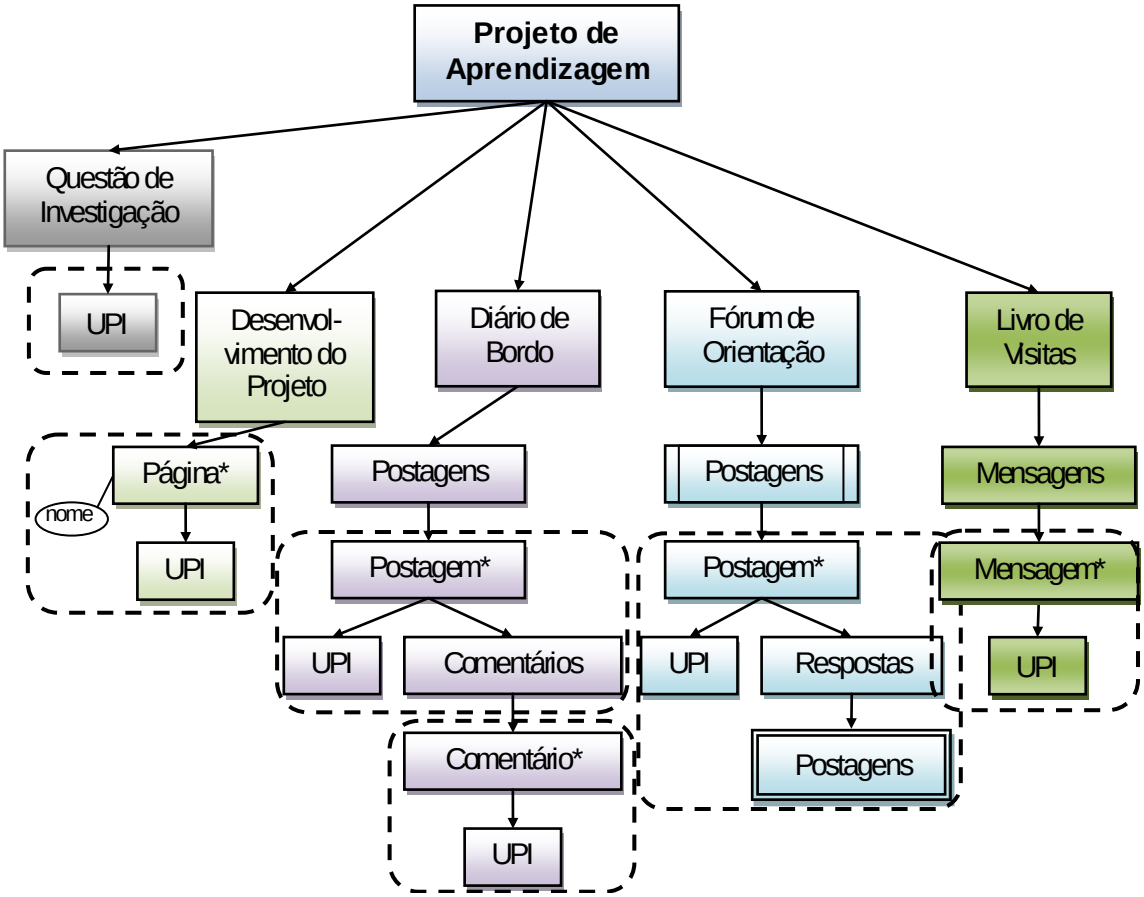


Figura 14. Diagrama do esquema de dados de uma ferramenta de apoio a projetos de aprendizagem, destacando os sub-VComs.

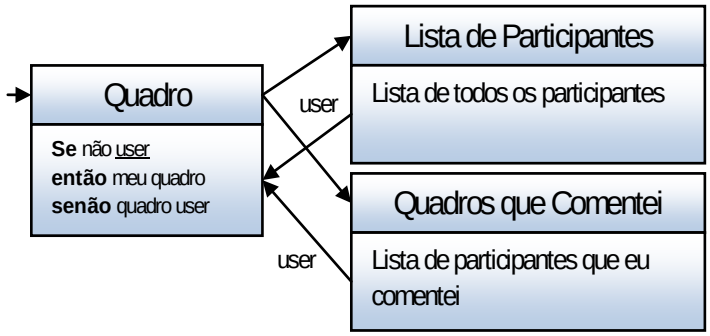


Figura 15. Diagrama de navegação do quadro de discussão.

4.5.1.3 Views

As views são os geradoras das páginas de um VCom. Foi escolhida a linguagem XSLT para se desenvolver essas views. XSLT é uma linguagem destinada a fazer transformações em XML. Nesse protótipo, as views "extraem" dados dos arquivos XML para formar suas páginas HTML.

A XSLT possui a funcionalidade de `<xsl:template match="XPATH">`, que é executado quando o arquivo XML combina com o padrão definido por um XPath. Também possui funcionalidades de seleção, repetição e variáveis.

Para se adquirir informações do sistema, como nome de usuário, foto ou papel de um usuário, utiliza-se uma funcionalidade especial do processador de XSLT/XML na linguagem PHP: `php:functionString('f1',parametros)`. Isso faz com que seja executado a função de nome "`f1`" do sistema.

O processo de publicação é feito através de links e é gerenciado pelo controller. Para gerar esses links, utiliza-se uma função do sistema que tem como parâmetros diversas informações, tais como: local (XPath) onde vai ser publicado; qual esquema de publicação; o texto do link; se haverá inserção de nova UPI ou de uma já existente.

Links para acessar outras páginas (ou views) também possuem uma ideia similar de geração dos links de publicação. Contudo, podem-se passar parâmetros que serão argumentos de entrada para as outras páginas acessadas.

A exibição de sub-VComs (Figura 14) foi feita utilizando-se cópias de views das ferramentas originais, que funcionam isoladamente. Para tal é preciso passar o XPath do sub-VCom que se deseja trabalhar. Ou seja, funciona como uma ferramenta separada, mas ainda estão todos associados ao VCom principal.

Na Figura 16, apresenta-se um trecho de código no editor de XSLT do meta-ambiente. Trata-se da view da página de "Lista de Participantes" do quadro de discussão. Nesse trecho podem-se observar alguns dos elementos citados anteriormente:

- Da linha 6 à linha 10 é um cabeçalho da página, com links para dois outras views, utilizando a função "generate_show_link";
- Da linha 14 à linha 26 é gerada uma lista com todos os usuários que postaram alguma coisa em seus quadros neste VCom;
- Na linha 22 é gerado um link para visualizar a view que gera a página do quadro de discussão, passando como parâmetro o id do usuário.

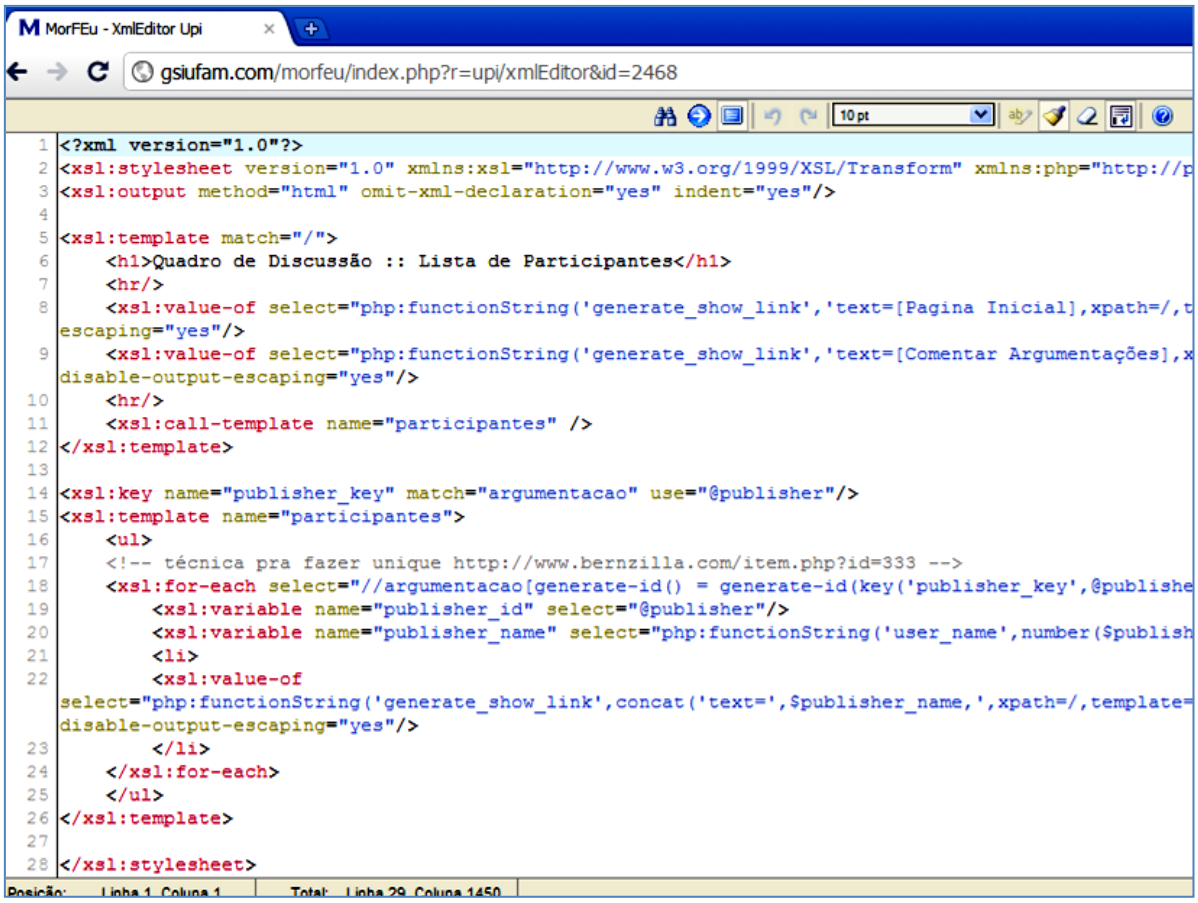


Figura 16. View em XSLT da página Lista de Participantes do VCom para Construindo Conceituações.

4.5.1.4 Alterando o Veículo de Comunicação

Nevado, Dalpiaz e Menezes (2009) descreveram uma versão básica para o quadro de discussão. Suas funcionalidades foram implementadas e foram descritas nas seções anteriores. No entanto, melhorias ainda puderam ser realizadas.

Como foi argumentado anteriormente, a execução dessa atividade tem caráter artesanal. Ou seja, cada situação vai demandar diferentes modificações. No entanto, isso não foi possível de ser feito primeiro no estudo de caso de Nevado, Dalpiaz e Menezes devido à carência de apoio tecnológico adequado.

Observa-se que, na montagem inicial da arquitetura pedagógica construindo conceituações, o papel do professor ficou obscuro. Não há espaço para sua interação no quadro de discussão. Um recurso simples que esse quadro deve conter é

possibilidade de interação direta entre o professor e um aluno, dentro do contexto de uma argumentação em particular. Assim, o professor pode auxiliar esse aluno e orientá-lo na direção desejada.

Dessa forma, foi criado mais um esquema de publicação, chamado de "nota". Essa nota é um comentário que o professor faz a algum dos outros elementos do quadro, como argumentação, comentário e réplica, do esquema de dados. Este novo esquema de publicação é composto por `<nota><upi/></nota>`, e pode estar associado a qualquer uma das tags citadas. Na Figura 17, é apresentado o novo esquema de dados com as alterações em destaque.

Na view da página do quadro, foi preciso acrescentar um pouco mais de código para realizar a seguinte funcionalidade: *Para um usuário professor (ou se já existe uma nota publicada) aparecerá um link para publicar uma nota*. E outro pedaço de código para listar todas as notas de cada tag. Na Figura 18, pode-se ver uma tela do sistema funcionando, um quadro de discussões, com dados de testes.

Essas alterações foram realizadas de forma relativamente fácil e rápida, cerca de duas horas de trabalho, com apenas um programador desenvolvendo o trabalho. Num sistema tradicional, no entanto, isso poderia acarretar uma mudança na estrutura do banco de dados e uma reformulação em diversos arquivos de códigos.

Um ponto importante a ressaltar é que essas alterações foram feitas com permissões de usuário comum. O sistema permite que o dono do VCom altere o model, as views e os esquemas de publicação. Todas as alterações são gerenciadas pelo sistema de versões, que guarda a versão anterior à alteração, permitindo que o usuário faça alterações sem preocupações de inutilizar a aplicação, pois pode revertê-las quando quiser.

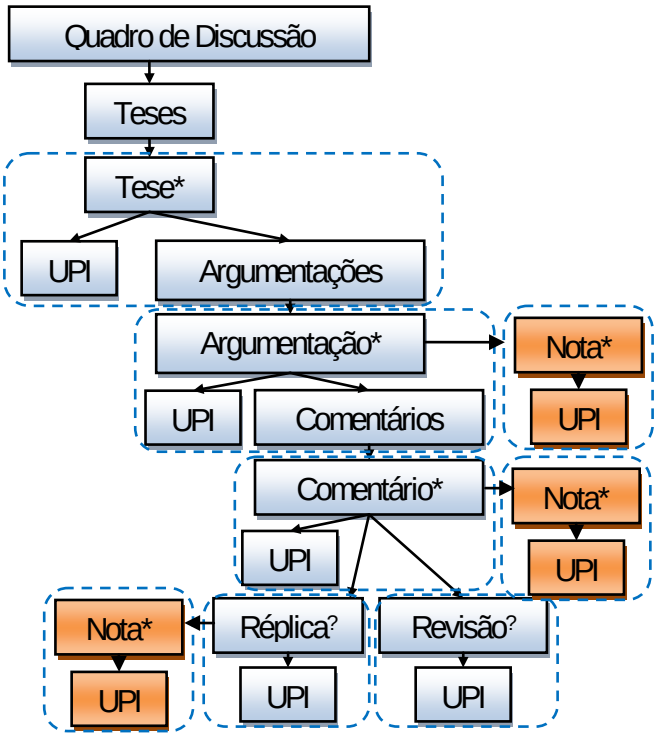


Figura 17. Diagrama do esquema de dados da ferramenta de apoio à arquitetura pedagógica Construindo Conceituações estendido para contemplar notas de texto do professor.

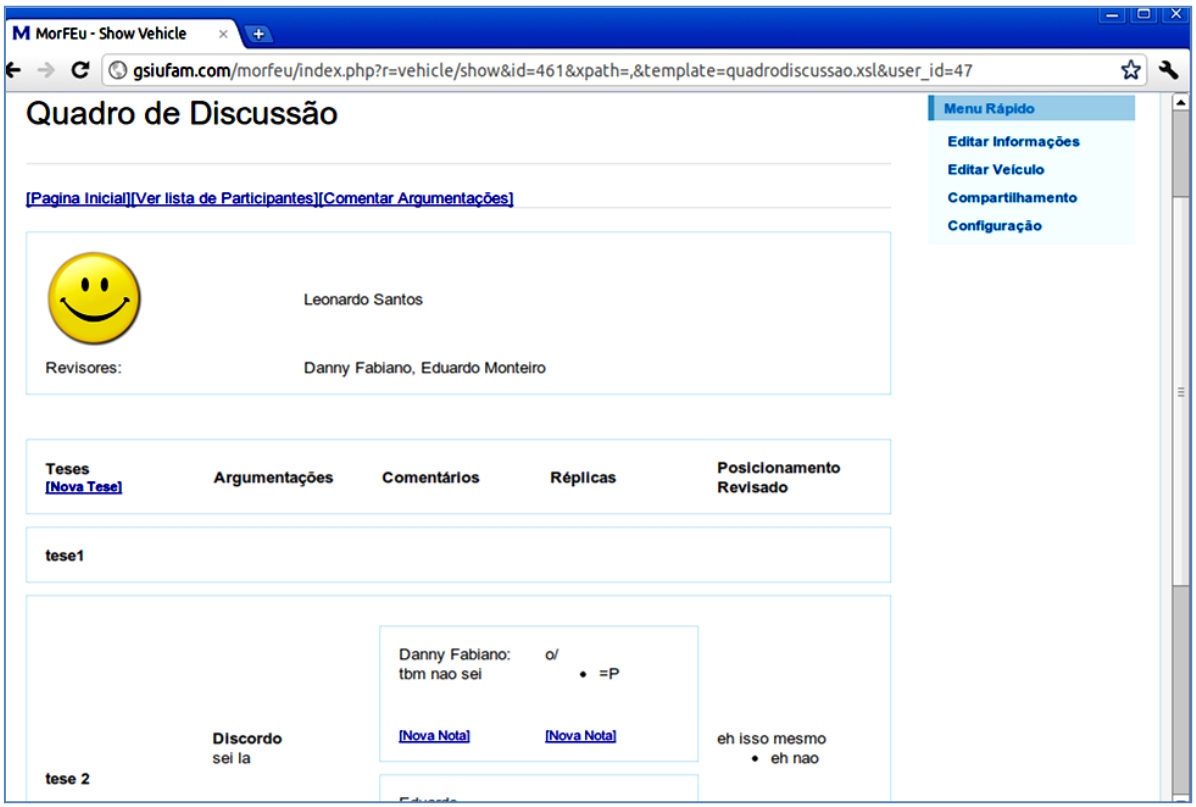


Figura 18. Tela do protótipo funcionando, um quadro de discussão da arquitetura pedagógica construindo conceituações.

4.6 Etapas de Avaliação da Proposta MOrFEu

A proposta do MOrFEu foi concretizada no protótipo apresentado nas seções anteriores. No entanto, é necessário avaliar se o MOrFEu é capaz atender aos requisitos levantados.

Primeiramente, é necessário saber se o MOrFEu realmente consegue generalizar as ferramentas de comunicação/interação existentes. Para isso, foram realizadas implementações de diversas dessas ferramentas, de diferentes contextos de aplicação e de diferentes funcionalidades, buscando uma expressividade das situações reais de uso.

Deseja-se avaliar também as questões da flexibilidade dos VComs. Para isso, se realizou um estudo de casos com usuários reais. Um estudo de caso é a metodologia mais adequada para isso devido ao fato do fenômeno das demandas por flexibilidade e a capacidade do MOrFEu de atender essas demandas estar muito ligadas ao contexto que elas surgem, não havendo limites claramente evidentes.

4.6.1 Prova de Conceito

Como prova de conceito foram implementados 14 ferramentas já existentes na Web ou que sua implementação não foi concretizada, de diferentes contextos de aplicação e de diferentes funcionalidades, buscando uma expressividade das situações reais de uso. Suas principais características e funcionalidades estão listadas nos itens a seguir.

4.6.1.1 Ferramentas comuns de interação

Ferramentas tradicionais de comunicação/interação na Web: **fórum, blog, wiki, enquete, glossário e mural**. Essas ferramentas são mais simples, pois não possuem muitas opções de regulação, com poucos papéis.

4.6.1.2 Fórum da Controvérsia Acadêmica

Um fórum modificado para apoio à prática pedagógica **Controvérsia Acadêmica** realizada em ambientes virtuais (MENDONÇA, CASTRO, *et al.*, 2003), onde o segundo nível do fórum é composto pelas postagens "Concordo", "Discordo" e "Depende", cada uma referente a um etapa da atividade. A atividade era composta por dois grupos de pessoas, onde na primeira etapa um grupo deveria discutir a favor do assunto a ser debatido e na segunda etapa contra, e vice-versa par ao outro grupo. A última etapa todos deveriam discutir no tópico "Depende". Para a implementação desses grupos, foi utilizado um artifício: o participante deveria escolher o grupo a que gostaria de pertencer, e ao fazer essa escolha a pessoa publicava em uma área destinada ao grupo escolhido, assim os participantes daquele grupo são as pessoas que publicaram nessa área do model. A mudança de etapas também foi resolvido de maneira similar, se existir uma dada publicação, então quer dizer que a atividade se encontra em uma certa etapa.

4.6.1.3 CARTOLA

Uma ferramenta web para o **objeto de aprendizagem CARTOLA** (ALVES, NUNES, *et al.*). Nesse jogo, o participante deve escrever um texto com três cartas sorteadas. No jogo original, as cartas são palavras e figuras, no entanto, foi feita uma simplificação permitindo apenas palavras, já que o protótipo ainda não dá suporte a UPIs que sejam imagens.

4.6.1.4 Júri Simulado

Um quadro de discussões para a arquitetura pedagógica **Júri Simulado** (REAL e MENEZES, 2007). Nessa arquitetura pedagógica, os participantes são divididos em três grupos, Acusação, Defesa e Júri, mais o Juiz. O caso a ser julgado é um assunto

geralmente polêmico que divide opiniões. O papel do Juiz é apenas intermediar o debate entre Acusação e Defesa, e o do Júri é votar a favor ou contra, quando chegar à etapa determinada. A escolha de grupos e mudanças de etapas foi feita de maneira similar ao que foi feito para a Controvérsia Acadêmica.

4.6.1.5 Construindo Conceituações

Um quadro para a arquitetura pedagógica **Construindo Conceituações** (NEVADO, DALPIAZ e MENEZES, 2009) descrito anteriormente na Seção 2.3.3 e exemplificado na Seção 4.5.1.

4.6.1.6 Projetos de Aprendizagem

Um ambiente para apoio básico a **Projetos de Aprendizagem** (FAGUNDES, NEVADO, *et al.*, 2005), descrito anteriormente na seção 4.5.1.

4.6.1.7 Diário Virtual de Aprendizagem

O **Diário Virtual** para aprendizagem de matemática (SERRES e BASSO, 2009). Onde os estudantes deveriam descrever sua solução para um determinado problema e comentar a solução de pelo menos dois outros colegas. Semelhante ao "Construindo Conceituações", com revisão por pares, mas numa versão mais simples.

4.6.1.8 Curso

Uma versão básica do **"Curso" do Moodle** (Moodle, 2011). Trata-se da página inicial de um curso, divida em tópicos, onde cada tópico são dispostas diversas ferramentas e recursos. As ferramentas e os recursos são sub-VComs, implementados

de maneira similar àqueles da ferramenta de apoio a Projetos de Aprendizagem (Figura 14).

4.6.1.9 Bolão da Copa

Uma ferramenta para apoio a um **bolão da copa do mundo de futebol**. Os primeiros jogos já estavam definidos e os jogos seguintes ocorriam de acordo com o resultado de jogos anteriores. Os participantes do bolão deveriam publicar seus palpites antes do início do jogo. Os participantes ganhavam pontos de acordo com os acertos nos placares dos jogos.

4.6.1.10 Conclusão

Esses VCom descritos acima foram implementados de maneira rápida. Eles possuem de uma a quatro views, e o tempo médio para desenvolvimento de cada view foi de no máximo dois dias, por apenas um programador. E a maior view implementada possui cerca de 260 linhas de código, muitos dos quais são similares a outros produzidos anteriormente.

Isso evidencia duas coisas: (1) que a proposta é factível, sendo capaz de implementar diversas ferramentas de diversos contextos; (2) a facilidade de implementação, avaliada pela quantidade de código produzido, o tempo gasto para as implementações e o tamanho da equipe de desenvolvimento, apenas uma pessoa.

4.6.2 Estudo de Caso

Um estudo de caso exploratório foi conduzido de forma a determinar como a plataforma lidaria com as demandas por modificação no mundo real. O objetivo desse estudo foi desenvolver hipóteses e proposições pertinentes para um estudo posterior,

de quais tipos de ambientes virtuais são possíveis de serem construídos usando essa plataforma.

Os casos são parte do curso de formação de professores e multiplicadores do Projeto UCA – Um Computador por Aluno (MINISTÉRIO DA EDUCAÇÃO, 2011). São vinte professores e técnicos administrativos do ensino fundamental de dez escolas situadas em diferentes municípios do estado do Amazonas, dois de cada escola, que atuam como multiplicadores. No início do ano de 2011, eles participaram da primeira parte desse curso presencialmente. Agora, eles prosseguem o curso na modalidade a distância. E os tutores desse curso são professores do Instituo de Computação da Universidade Federal do Amazonas.

O cenário do estudo de caso foi um curso com dez multiplicadores (professores responsáveis pelo treinamento de outros professores) no uso de tecnologias da informação e comunicação (TICs) em escolas de ensino fundamental. Esse curso faz parte do Projeto UCA – Um Computador por Aluno (MINISTÉRIO DA EDUCAÇÃO, 2011). Os participantes tinham que obter certo nível de habilidades antes de voltar às suas próprias escolas e replicar as estratégias e materiais lá. Todos os participantes concordaram em serem observados e em usar o protótipo desenvolvido.

Dentre as várias atividades planejadas para aquele curso, um determinado tempo foi alocado para a exploração de arquiteturas pedagógicas, com ambientes virtuais construídos no protótipo. Duas tarefas foram observadas: o uso do Debate de Teses e uma avaliação de objetos de aprendizagem usando fóruns para discussão em grupo e um wiki para documentação.

Os participantes tinham pouco conhecimento no uso de ferramentas da Internet, assim a primeira tarefa teve uma etapa preliminar: depois da discussão de como o Debate de Teses (NEVADO, DALPIAZ e MENEZES, 2009) funciona, um debate estruturado foi realizado com apenas três participantes (os outros ficaram apenas como observadores) e material convencional (papel e caneta). No fim processo, todos

os participantes (três participantes efetivos e sete observadores) disseram que foi muito complicado a aplicação da arquitetura pedagógica sem suporte computacional.

Em seguida foi apresentada a ferramenta de suporte ao Debate de Teses construída usando protótipo e todos os dez participantes começaram a atividade. Os observadores do estudo de caso participaram dando suporte técnico e com questões sobre a arquitetura pedagógica, tomando notas de todos os problemas reportados ou funcionalidades que os participantes gostariam de ter na ferramenta. O mesmo tipo de assistência foi dado na segunda tarefa.

No final das duas atividades, foi possível identificar um total de 31 modificações distintas. Das quais 10 eram simples *bugs* e 11 eram no VCom utilizado para melhor acomodar preferências pessoais. Todas essas 21 modificações foram facilmente realizadas (cerca de 6 horas de trabalho) e foram verificadas pelos participantes no dia seguinte.

Todas as modificações restantes foram relacionadas com variações da arquitetura pedagógica, especialmente nas regras de interação, que o participantes imaginaram que seria mais adequado aos seus cenários de aplicação (estudantes e professores da sua própria escola). Porém, essas alterações não foram implementadas porque elas necessitaram de mudanças no Controller. Pois o Controller não suporta modificações de suas definições de regras de interação.

Além dessas variações no Debate de Teses, alguns participantes sugeriram outro VCom. Esse VCom seria um editor de texto multi-operado cooperativo e síncrono.

Isso apoia a hipótese de que mesmo que um desenvolvedor de groupware seja capaz de desenvolver uma aplicação "ótima" para um grupo, a aplicação irá eventualmente se tornar inadequada devido a novas situações e problemas que certamente aparecerão (FUKS, RAPOSO, *et al.*, 2007).

4.7 Conclusão do Capítulo

Falta escrever.

5 O Arcabouço MX para desenvolvimento de Ambientes Virtuais

Partindo-se das ideias de Ellis e Wainer (2000), de desejarem que seja possível obter aplicações do tipo Guardiãs de Artefato especificáveis, neste capítulo apresenta-se uma abordagem para especificação (ou definição) de ambientes virtuais de interação humano-computador-humano baseado nas ideias do paradigma MOrFEu.

Algumas vezes a colaboração entre um grupo de pessoas é centrada no acesso e modificação de um conjunto de dados compartilhados, que foi chamado de artefato. O Guardião do Artefato é o conjunto de funcionalidades relacionadas com a manipulação desse artefato. Algumas funcionalidades do guardião são: Controle de acesso ao artefato; Controle de acesso simultâneo ao artefato; Versionamento do artefato; Armazenamento de informações do autor e data de modificação (ELLIS e WAINER, 2000).

O paradigma MOrFEu visualiza os ambientes virtuais exatamente dessa forma: os ambientes virtuais são organizadores das produções de seus usuários, que são armazenadas no documento desse ambiente virtual, o artefato.

Normalmente, as aplicações Guardiães têm embutidas (*embedded*) suas regras de negócio definidas *a priori* pelo desenvolvedor dessa aplicação. Mas o interessante seria se seus próprios usuários pudessem definir ou adaptar as regras de negócio dessa aplicação (ELLIS e WAINER, 2000).

Essa é exatamente a ideia central deste trabalho, fazer aplicações Guardiães de Artefatos especificáveis, partindo da ideia do MOrFEu que visualiza as aplicações Coordenadoras e Comunicadoras também como Guardiães. Ou seja, com um modelo

formal de especificação desse tipo de aplicação é possível torná-los concretos e modificáveis em tempo de execução.

Nas próximas seções, apresenta-se uma abordagem para a especificação (ou definição) de ambientes virtuais de interação humano-computador-humano baseado no MOrFEu, que foi chamada de **MX**, ou **"MOrFEu Extendido"**.

A principal diferença entre o MOrFEu e o MX é que a abordagem MX organiza de melhor forma os conceitos do MOrFEu e elenca os elementos dos ambientes virtuais. De forma que é possível descrever precisamente a estrutura, funcionalidades e navegação de um ambiente virtual. Com a abordagem MX, um ambiente virtual possui três elementos: o Documento, que armazena os dados; o Protocolo de Interação, que define as regras de negócio da aplicação; e a Interface, que define a navegação e interação do usuário com a aplicação.

5.1 Principais características

O MX reúne as características do MOrFEu, porém outras características foram adicionadas e outras receberam outra nomenclatura.

Existe um conjunto de **pessoas** que são usuários de ambientes virtuais. Essas pessoas fazem produções que são chamadas **UPIs – Unidades de Produção Intelectual** – assim como no MOrFEu. As UPIs são pequenas unidades que armazenam dados criados pelas pessoas, podem ser de diversos tipos: texto, HTML, código-fonte, imagem, som, etc. As pessoas que são produtoras dessas UPIs são chamadas **autores** das UPIs. Toda UPI possui pelo menos um autor, e podem existir independentemente dos ambientes virtuais onde elas são utilizadas.

Os ambientes virtuais no MX também são os **Veículos de Comunicação (VComs)**. Através dos VComs são feitas as interações de seus usuários.

Um VCom tem três componentes: **Documento, Protocolo de Interação** e **Interface**. Assim como de Ellis e Wainer (1994) também visualizam três

componentes de groupware, respectivamente: modelo ontológico, modelo de coordenação e modelo de interface do usuário. Dessa forma, o Documento armazena os objetos (UPIs) e possui operações sobre esses objetos que estão disponíveis para seus usuários. O Protocolo de Interação é uma descrição dos aspectos dinâmicos do sistema: o controle, o fluxo de dados e as regras de negócio. A Interface descreve a apresentação gráfica do sistema para o usuário, e entre os usuários.

O documento guarda as produções dos usuários. Cada produção dos usuários foi chamada de Unidade de Produção Intelectual (UPI). Uma postagem num fórum, uma mensagem num chat, uma página Wiki, são todas UPIs.

As UPIs podem existir mesmo que não sejam usadas no VCom. Quando é feito o uso de uma UPI em um VCom, diz-se que a UPI foi publicada nele, e essa publicação é registrada no Documento.

A seguir, são apresentadas as **definições formais** para os elementos gerais do MX.

Definição 1. Existe um conjunto P ("Pessoas") de usuários do ambiente.

Definição 2. Um ambiente virtual é chamado de veículo de interação (VCom), de forma que um VCom E ("*environment*") é

$$E = (D, I, W)$$

onde D é um documento, I é um protocolo de interação e T é a interface.

Definição 3. Uma unidade de produção intelectual (UPI) é uma unidade de dados criada por algum usuário, que deve ser de algum tipo definido, como por exemplo: número, texto, hipertexto, imagem, som, arquivo digital, etc., ou NULO.

Definição 4. Existe um conjunto U de todas as UPIs de todos os usuários.

Definição 5. Existe conjunto R ("Redator") de autoria, que é uma relação entre usuários e UPIs que diz quais usuários são autores de quais UPIs, de forma que existe pelo menos um autor para cada UPI. Assim,

$$R = \{(u, p) | \forall u \in U \wedge \exists p \in P\}$$

5.2 Documento

O Documento armazena os UPIs que são publicadas no VCom. As UPIs publicadas relacionam-se entre si através de relações. Elas também podem ter relações com outras pessoas e grupos além de seus autores. Essas relações organizam as UPIs publicadas no VCom. O Documento provê também operações para o acesso a essas UPIs, com ações do tipo CRUD (*Create, Read, Update and Delete*).

É importante ressaltar que o Documento apenas armazena as UPIs publicadas no VCom, e suas relações. A forma como devem ser publicadas e quais as relações entre as UPIs publicadas são especificadas no Protocolo de Interação.

Publicar é o ato de socializar uma UPI em um ambiente virtual (VCom). Publicar uma UPI tem o intuito de promover uma ação de interação com os outros participantes do ambiente. Além de UPIs, as relações entre UPIs e entre UPIs e pessoas/grupos também são publicadas. Ou seja, publicar é incluir UPIs e relações no Documento de VCom. Essa publicação é sempre feita por um usuário, chamado de **publicador**.

Quando uma UPI é publicada, ela recebe um rótulo para ganhar um sentido de por que ela está sendo publicada nesse VCom. Dessa forma, uma mesma UPI pode ser publicada com dois rótulos diferentes, denotando duas semânticas diferentes para essas publicações num ambiente virtual.

As **Relações** são ligações entre UPIs e entre UPIs e pessoas/grupos. Uma relação entre duas UPIs denota que elas têm alguma ligação no contexto do ambiente virtual. É possível que essa relação seja rotulada para dar uma melhor semântica para a ligação entre essas duas UPIs. O mesmo acontece com as relações entre uma UPI e uma pessoa, ou com as relações entre uma UPI e um grupo. Por exemplo, um comentário em um blog deve está associado com a postagem referente àquele comentário. Essas relações podem ser representadas através de um grafo, onde os vértices são UPIs e os arcos são as relações entre essas UPIs.

É possível, inclusive, que existam duas relações distintas entre duas UPIs, com rótulos diferentes, expressando semânticas diferentes para essas relações.

Uma publicação ainda pode está associado com outro usuário (ou grupo), além do publicador. Por exemplo, uma foto no Facebook pode conter marcações de pessoas, que associam essa foto a usuários desse ambiente.

Para se manipular com o Documento são necessárias **operações**. Essas operações podem ser de leitura (consulta aos objetos do Documento) ou de escrita (incluir, remover e alterar os objetos do Documento). Essas operações também são chamadas de operações CRUD – *Create, Read, Update and Delete*.

As **operações de consulta** são relativas à aquisição de informações do Documento, tais como: quais são todas as UPIs publicadas; dada uma UPI, quais UPIs essa UPI se relaciona; quais as relações que possuem um dado rótulo; quais as UPIs relacionadas com um dado usuário; etc.

Uma **Pesquisa** é consulta mais complexa. É uma função composta por uma ou mais operações de consulta que retorna um resultado. As pesquisas em um Documento são especificáveis.

As **operações de alteração** são aquelas que fazem mudanças no Documento. Alguns exemplos dessas mudanças são: publicação de uma UPI; relacionar uma UPI com outra UPI; relacionar uma UPI com uma pessoa; retirar uma relação; modificar o rótulo de uma relação; etc.

Em um Documento, há também o conceito de **Atividade**, que é uma sequência de operações de alteração realizadas sobre um documento. Essas atividades podem ser dinamicamente especificadas. Elas podem ser modificadas, novas atividades podem ser incluídas e removidas. Isso obviamente pode mudar a estrutura do geral do documento, mas não modifica as UPIs já publicadas.

O documento de um VCom possui operações que permitem consultar e alterar os dados contidos nele. A seguir são listadas quais são essas operações quais seus

parâmetros. No caso de operações de consulta, elas são funções, por isso retornam alguma coisa.

A seguir, são apresentadas as **definições formais** para os elementos de um Documento do MX.

Definição 6. Um **Documento** de VCom é formado por um conjunto de UPIs publicadas; um conjunto de relações entre essas UPIs, e um conjunto de relações entre as UPIs publicadas e usuários, um conjunto de relações entre UPIs publicadas e usuários, e um conjunto de UPIs publicadas e grupos. Tal que o documento pode ser representado como $D = (V, R, A, B, C)$ em que:

- V é um conjunto de UPIs publicadas, os vértices do grafo, onde $V \subseteq U \times R \times P$, tal que:
  - U é o conjunto de UPIs
  - P é o conjunto de usuários
- R é um conjunto de rótulos, que são strings e indicam qual a natureza da publicação da UPI ou da relação entre as UPIs;
- $A \subseteq V \times V \times R \times P$ é um conjunto de relações entre UPIs, com o devido autor que publicou essa relação, ou seja, são as arestas de um grafo, rotuladas com um rótulo e usuário publicador, sendo que as UPIs são os vértices;
- $B \subseteq V \times R \times P$ é um conjunto de relações entre UPIs publicadas e usuários, denotando que outros usuários, além do autor, podem ter relação com essa UPI; e
- $C \subseteq V \times R \times G$ é um conjunto de relações entre UPIs publicadas e grupos, denotando que grupos podem ter relação com essa UPI, onde o conjunto de grupos G faz parte do protocolo de interação I.

5.3 Protocolo

O **Protocolo de Interação** (ou apenas Protocolo para simplificar) define como o veículo de interação pode ser utilizado. Ou seja, trata principalmente das permissões de operações serem realizadas. Para isso, o Protocolo precisa lidar com

papéis, atores e grupos de usuários (relações entre usuários); e os estados da aplicação (*workflow*).

O protocolo de interação de um VCom é um conjunto de regras que rege a forma com que os usuários podem interagir com o VCom. Consequentemente, define como os usuários podem interagir entre si.

Um protocolo de interação de um VCom é formado por cinco elementos: um conjunto de papéis e seus atores; um conjunto de grupos e seus participantes; um conjunto de estados; um conjunto de estados atuais; e um conjunto de permissões.

No Protocolo, são definidos os **papéis** que os usuários podem assumir durante o processo de interação. São definidos também os **grupos** de usuários e quem pode participar desses grupos. Os usuários que interpretam um papel são chamados de **atores** daquele papel. E os participantes de um grupo são chamados **membros** do grupo.

Ambas as características dizem respeito a relações entre usuários do veículo de interação. Em essência, um papel é um grupo de usuários, ambos são conjuntos de usuários. No entanto, um papel cumpre uma funcionalidade específica, quando se está definindo as permissões das operações.

No Protocolo, também deve estar definidas questões do fluxo de trabalho (*workflow*). Aqui devem ser definidos os possíveis **estados** que a aplicação pode assumir, assim como os **estados iniciais**. O Protocolo também controla quais são os **estados atuais** na execução do VCom.

Para cada estado, deve ser definido um conjunto de **permissões**. Uma permissão define "**quem pode fazer o que, quando e como**":

- **Quem** é um ator de um papel;
- **O que** é uma operação sobre o Documento, ou sobre o próprio Protocolo;
- **Quando** é um estado;
- **Como** são as condições que a operação pode ser executada.

Dessa forma, uma **atividade** (sequência de operações de escrita no Documento) só pode ser realizada se todas as operações dessa atividade são permitidas de serem executadas. O mesmo acontece para as **pesquisas** (função com várias operações de consulta ao Documento).

A seguir, são apresentadas as **definições formais** para os elementos do Protocolo no MX.

Definição 7. Um protocolo de interação I é formado por **seis elemento**s, de forma que $I = (R, G, S, S', L, SL)$, onde:

- R ("*roles*") é um conjunto de papéis de usuários, de forma que $r \in R = (Rotulo, Q)$, onde $Rotulo$ é uma string e $Q \subseteq P$;
- G ("grupo") é um conjunto de grupos de usuários, de forma que $g \in G = (Rotulo, Q)$, onde $Rotulo$ é uma string e $Q \subseteq P$;
- S ("*state*") é um conjunto de estados;
- S' é um conjunto de estados atualmente ativos, de forma que $S' \subseteq S$;
- L ("licença") é um conjunto de permissões;
- SL é um conjunto de relações entre estados e permissões, denotando quais permissões são aplicáveis em cada estado, de forma que $SL \subseteq S \times L$.

Definição 8. Uma **permissão** $l \in L$ é formada por quatro elementos, de forma que $l = (w_o, w_a, w_e, H)$, onde:

- w_o é um papel, de forma que $w_o \in R$;
- w_a é uma operação, sobre o Documento, ou sobre o Protocolo ou sobre a Interface;
- w_e é um estado, de forma que $w_e \in S$;
- H é uma função de $condicao(u, o)$, que mapeia para o conjunto $\{Verdadeiro, Falso\}$, onde $u \in U$ é o usuário requisitando a permissão e o é a operação requisitado.

5.4 Interface

A Interface descreve a apresentação gráfica do sistema com o usuário, e entre os usuários. Tratando-se de aplicações web, está se falando de páginas web e a navegação entre elas, com links e passagem de parâmetros nos links.

A Interface tem basicamente duas funções: **exibir conteúdo presente no Documento** e **disparar a execução de atividades**.

Existe uma **página inicial** do veículo, que a primeira página a ser exibida quando o VCom é acessado. A partir dela é possível navegar para as outras páginas.

No entanto, essa página inicial pode assumir diversas formas, dependendo do estado do VCom e do usuário que a está acessando. Isso pode acontecer também para as outras páginas. Assim, um gerador de conteúdo diferente pode ser definido para cada par $\langle$estado; papel do usuário visualizador$\rangle$.

Esse gerador de conteúdo é chamado de **template**. Um template realiza pesquisas no Documento e exibe conteúdo em forma de HTML.

Além de exibir o conteúdo de um template, uma página também pode disparar a execução de uma atividade. Dessa forma, a uma página pode estar associada uma atividade diferente para cada par de estado e papel.

Nisso, estão definidas dois tipos de páginas: as **páginas de consulta** e as **páginas de atividades**.

Como qualquer página web, essas páginas também podem receber **parâmetros**, que servem de entrada para o conteúdo que elas exibem ou para a atividade que deve ser executada. Esses parâmetros são então repassados para os templates.

Além das páginas padrão do VCom, cada usuário pode criar suas próprias páginas para acessar o Documento da maneira que desejar. É óbvio que esse acesso ao Documento é regido pelas regras definidas no Protocolo.

Dessa forma, a Interface de um VCom possui cinco componentes: páginas de conteúdo; páginas de execução de atividades; templates; e relações entre os estados, papéis, páginas e templates.

Definição 9. A **interface** W de um VCom é formado por cinco componentes, de forma que $W = (P_c, P_a, T, Y)$, onde:

- A página $p_c \in P_c = (Rotulo)$ possui um rótulo do nome da página, e o template que gera seu conteúdo é definido por Y;
- A página $p_a \in P_a = (Rotulo, Atividade)$ possui um rótulo do nome da página e a atividade que ele executa;
- T é um conjunto de templates de forma que seus elementos são funções que fazem consultas no Documento e geram strings no formato HTML;
- $Y \subseteq E \times R \times P_c \times T$ é um conjunto que define "qual o conteúdo que sera exibido para um usuário, dependendo do estado do VCom e do papel que ele interpreta".

5.5 Agentes

Além do funcionamento do VCom, o MX prevê um relacionamento com agentes auxiliares. É possível identificar inicialmente quatro tipos de agentes: agentes gatilhos (ou reativos), agentes de serviços, agentes de percepção e agentes inteligentes. Esses agentes interpretam papéis de usuários e, em geral, são limitados pelos mesmos mecanismos que um usuário comum, ou seja, estão limitados às permissões de Pesquisas e Atividades.

Assim, os sensores de um agente são as Pesquisas e seus atuadores são Atividades. Além dos sensores de Pesquisas, a hora atual pode ser um sensor importante, já que um agente gatilho pode ser programado para mudar de estado em determinado tempo. Outro tipo de atuador são *popups* em páginas, um agente pode gerar uma página para interagir via interface com o usuário. Para isso, esse tipo de agente precisa de um sensor de quem é o usuário que está visualizando a página atual.

Os **agentes gatilhos** (*triggers*) são agentes reativos. Os gatilhos são atividades especiais que são realizadas automaticamente se alguma condição é satisfeita. Por exemplo, uma **troca de estado** é um tipo de gatilho, no qual espera alguma condição ser atendida e promove uma troca de estado para outro, ou seja, executa uma atividade (uma sequência de operações): remove o estado da lista de estados atuais e coloca outro estado nessa lista.

Os **agentes de serviço** também são agentes reativos, mas eles possuem um sensor que aceita requisições, que são atendidas pelo agente, que provê uma resposta para tal requisição. Um exemplo desse tipo de agente são **agentes de busca**: por exemplo, eles podem indexar o conteúdo das UPIs publicadas utilizando Pesquisas e respondem a buscas por palavras-chave, respondendo com uma lista de UPIs como resposta. As requisições a esses agentes podem ser feitos no código dos templates.

Os **agentes de percepção** dão suporte aos usuários a perceberem o que os outros participantes do ambiente virtual estão fazendo, permitindo que ele contextualize suas atividades dentro do grupo (SPÓSITO, CASTRO e CASTRO, 2008). Esse tipo de agente pode ter uma ação especial relacionada com um template, eles podem gerar *pop-ups* para apresentar as informações relevantes para o usuário, de acordo com o contexto.

Os **agentes inteligentes** realizam processos e tomadas de decisão mais sofisticadas, tomando como base os dados do contexto do usuário, pesquisas no Documento e no histórico das ações e decisões tomadas, eles podem tomar decisões e atuar de forma mais incisiva. Podem utilizar todos os sensores disponíveis, assim como podem usar todos os atuadores disponíveis. Um exemplo comum desse tipo de agente são os **tutores inteligentes**.

Essa divisão de classes de agente apresentada não é restritiva a somente essas classes. Trata-se apenas de uma organização de possíveis tipos de agentes que atuam em um ambiente virtual.

5.6 Exemplos

FALTA FAZER.

5.7 Comparação do MX com o MOrFEu

Na Tabela 3, é apresentado um resumo das características que compõe o arcabouço MX. Adicionalmente, é apresentado uma comparação com o MOrFEu, mostrando as características que foram estendidas do MOrFEu. A característica "1.2. Título da UPI" é a única característica que foi abandonada pelo MX, no entanto o título de uma UPI pode ser modelado como outra UPI relacionada.

Tabela 3. Resumo das características do MX em comparação com o MOrFEu.

Características	Morfeu	MX
1. UPIs	Sim	Sim
1.1. Tipo de Conteúdo:		
1.1.2. Texto	Sim	Sim
1.1.3. HTML	Sim	Sim
1.1.4. Código-Fonte	Sim	Sim
1.1.5. NULO	Sim	Sim
1.1.6. Outro	Sim	Sim
1.2. Título da UPI	Sim	Não
1.3. Usuários (Logados)	Sim	Sim
1.4. Grupos	Sim	Sim
1.5. Papéis	Sim	Sim
1.6. Conjunto de UPIs de todos os usuários	Sim	Sim
1.7. Autor de UPIs	Sim	Sim
2. Documento:		
2.1. UPIs Publicadas	Sim	Sim
2.2. Usuário Publicador	Sim	Sim
2.3. Rótulo de Publicação	Sim	Sim
2.4. Relações Entre UPIs		
2.4.1. Relações 1-N	Sim	Sim
2.4.2. Relações N-N	Não	Sim
2.4.3. Rótulos de Relações UPI-UPI	Não	Sim
2.5. Relações entre UPIs publicadas e Usuários/Grupos	Não	Sim
2.5.1. Rótulo de Relações UPI-Usuário	Não	Sim
2.6. Definição de Operações sobre o Documento	Não	Sim
2.6.1. Operações de Consulta	Sim	Sim

2.6.2. Operações de Escrita	Sim	Sim
2.6.3. Pesquisas	Sim	Sim
2.6.4. Atividades	Sim	Sim
3. Protocolo	Não	Sim
3.1. Estados	Não	Sim
3.1.1. Estados Iniciais	Não	Sim
3.1.2. Estados Atuais	Não	Sim
3.2. Especificação de Permissões	Não	Sim
3.2.1. Quem	Sim	Sim
3.2.2. O que	Sim	Sim
3.2.3. Quando	Não	Sim
3.2.4. Como	Não	Sim
3.3. Operações sobre o protocolo	Não	Sim
3.3.1. Editar Estados	Não	Sim
3.3.1.1. Criar/Remover Estados	Não	Sim
3.3.1.2. Mudar Estado Atual	Não	Sim
3.3.2. Editar Permissões	Não	Sim
4. Interface		
4.1. Página	Não	Sim
4.1.2. Página de Consulta	Não	Sim
4.1.2. Página de Atividade	Sim	Sim
4.1.3. Comportamento condicional das páginas	Não	Sim
4.2. Template	Sim	Sim
4.2.1. Parâmetros	Sim	Sim
4.3. Operações sobre a Interface	Não	Sim
4.3.1. Editar o comportamento das páginas	Não	Sim
4.3.2. Editar/criar/remover templates	Sim	Sim
5. Agentes	Não	Sim
5.1. Tipos	Não	Sim
5.1.1. Agentes Gatilho	Não	Sim
5.1.2. Agentes de Serviço	Não	Sim
5.1.3. Agentes de Percepção	Não	Sim
5.1.4. Agentes Inteligentes	Não	Sim
5.2. Sensores	Não	Sim
5.2.1. Operações de consultas	Não	Sim
5.2.2. Tempo	Não	Sim
5.3. Atuadores	Não	Sim
5.3.1. Operações de escrita e atividades	Não	Sim
5.3.2. Ações na Interface	Não	Sim

Quantitativamente, das 64 características listadas, 34 delas foram extensões do MX.

Qualitativamente, o MX é capaz de modelar ambientes virtuais melhor que o MOrFEu. Alguns ambientes virtuais só eram possíveis de se modelar (e implementar) no MOrFEu utilizando-se alguns artifícios, e outros não eram possíveis de nenhuma forma (SANTOS, CASTRO e MENEZES, 2010) (SANTOS, CASTRO e MENEZES, 2012).

Na Tabela 4, são apresentados os ambientes virtuais que estão servindo de objeto de estudo desta tese. Eles foram escolhidos por suas características representarem bem o universo dos ambientes virtuais. Ou porque tratam-se de ambientes virtuais que são ferramentas de atividades colaborativas dinâmicas, que mudam para cada instância de aplicação, como as Arquiteturas Pedagógicas. Além disso, foram escolhidos dois jogos on-line que podem ser considerados ambientes virtuais, um jogo de dominó on-line e um ambiente para apostas em jogos da copa do mundo de futebol. Maiores detalhes sobre esses ambientes e como eles são modelados com o MX podem ser encontrados no Apêndice A.

Tabela 4. Modelagem de Ambientes Virtuais com o MOrFEu e o MX.

Ambientes Virtuais	MOrFEu	MX
Blog	Sim	Sim
Fórum	Sim	Sim
Glossário	Sim	Sim
Mural	Sim	Sim
CARTOLA	Sim	Sim
Enquete	Sim, com artifícios	Sim
	Precisa de estados	
Debate de Teses	Sim, com artifícios	Sim
	Precisa de estados	
	Precisa de relação UPI-Usuário	
	Precisa de definições flexíveis de permissões e funcionalidades	
Júri Simulado	Sim, com artifícios	Sim
	Precisa de estados	
Bolão Copa do Mundo	Sim, com artifícios	Sim
	Precisa de estados	
	Precisa de comportamento condicional das páginas	
Controvérsia Acadêmica	Sim, com artifícios	Sim
	Precisa de estados	

	Precisa de comportamento condicional das páginas	
Wiki	Sim, incompleto	Sim
	Falta funcionalidades exercidas pelos agentes	
Chat	Sim, incompleto	Sim
	Falta funcionalidades exercidas pelos agentes	
Projeto de Aprendizagem	Sim, incompleto	Sim
	Precisa de definições flexíveis de permissões e funcionalidades	
Curso AVA	Sim, incompleto	Sim
	Precisa de definições flexíveis de permissões e funcionalidades	
Facebook	Não	Sim
	Precisa de relação UPI-Usuário	
	Precisa de definições flexíveis de permissões e funcionalidades	
Dominó	Não	Sim
	Precisa de estados	
	Precisa de protocolo sofisticado	

Analisando a Tabela 4, as características do MX que se sobressaem em relação ao MOrFEu são:

- Estados;
- Precisa de comportamento condicional das páginas;
- Definições flexíveis de permissões e funcionalidades;
- Funcionalidades exercidas pelos agentes.

As funcionalidades de estados do MX eram possíveis de ser implementadas no MOrFEu através de alguns artifícios, implementando UPIs que registravam os estados. Ou seja, no MX essa funcionalidade foi apenas explicitada e formalizada.

Os dois ambientes virtuais que não foi possível modelar com o MOrFEu são o Facebook e o Jogo de Dominó. As funcionalidades impeditivas para modelagem dos dois foram respectivamente: relação UPI-Usuário e Protocolo Sofisticado. As relações UPI-Usuário são essenciais no Facebook para implementar as funcionalidades de "Marcar amigo" e "Mensagem privada". De todos os ambientes analisados, o Jogo de Dominó foi o único a necessitar de um protocolo mais sofisticado, com permissões complexas em nível de operações do Documento. Para os outros ambientes, as

permissões no nível de páginas e templates (comportamento condicional de páginas) foram suficientes. No Capítulo 7, é apresentado um protótipo que explora as características específicas do Protocolo do Jogo de Dominó.

Além disso, uma característica importante do MX é a possibilidade de se definir as permissões de forma declarativa e que essas permissões são flexíveis. Isso possibilita a edição e criação de funcionalidade do ambiente. Esse é um aspecto muito importante quando de trata de aplicações que estão em evolução constante, como o Facebook, e para aplicações que precisam ser personalizadas em cada instâncias, como os ambientes de suporte a Arquiteturas Pedagógicas.

Observa-se, ainda, que os agentes desempenham um papel importante na modelagem de ambientes virtuais. Funcionalidades periféricas, como busca e funcionalidades de percepção, podem ser entendidos como atuações de agentes, que auxiliam no andamento das atividades realizadas pelos usuários do ambiente.

5.8 Conclusão do Capítulo

FALTA FAZER.

Falar das principais extensões do MX e de como o MX tem maior poder de representação que o MOrFEu.

Falar do framework do próximo capítulo, que faz parte do arcabouço MX. Além de modelar, ele também prevê como implementa esses ambientes.

6 Um framework de desenvolvimento de ambientes virtuais

Neste capítulo apresentaremos um framework de desenvolvimento de ambientes virtuais construídos com os conceitos do MX. Com ele é possível implementar todos os conceitos do MX, tais como: UPI, Publicação, Documento, Operações, Atividades, Consultas, Protocolo de Interação, Permissões, Papéis, Interface, Templates, Páginas, etc.

Foi utilizado o framework Yii (Yii, 2011) para agilizar o desenvolvimento do framework do MX. O Yii possui arquitetura Model-Viw-Controller (MVC), e os componentes do MX foram acomodados nessa arquitetura.

As UPIs são armazenadas em um banco de dados relacional. Diferente do protótipo do MOrFEu, o framework MX não necessita de um banco de dados NoSQL, pois o MX possui componentes simples que se encaixam perfeitamente. No protótipo do MOrFEu isso era necessário pois as relações UPI-UPI não haviam sido claramente definidas, necessariamente elas eram hierárquicas e representadas em um XML. No MX, as relações UPI-UPI são bem definidas e são independentes entre si. Assim, as relações UPI-UPI e UPI-Usuário foram também armazenadas no banco de dados relacional.

O Documento é assim um Model que abstrai e controla as publicações no VCom, tanto de UPI quanto de relações. Nesse Model do Documento são previamente implementadas as operações básicas de consulta e alteração. As pesquisas e atividades (que são composições de operações) devem ser escritas pelo desenvolvedor do VCom.

No Model do Protocolo, o desenvolvedor deve escrever o estado atual, as mudanças de estado e os papéis. Além disso, o desenvolvedor deve escrever as

permissões em nível de operações, que determinam se um usuário pode ou não executar uma dada operação.

Os Templates foram implementados como Views, trechos de códigos que produzem uma página HTML, usando as Pesquisas.

As páginas foram implementadas no Controller do VCom. Nesse Controller também são definidas as permissões em nível de páginas, ou seja, as visualizações condicionais de páginas. No Yii, um Controller define qual View será exibido para uma determinada página. Assim, o desenvolvedor precisa escrever um código que define qual Template será exibido para o usuário solicitante, levando em consideração seu papel no VCom.

Assim, esse framework possui suas partes fixas: no Documento, as operações de leitura e escrita. E se suas partes que deve ser desenvolvidas pelo criador do VCom:

- No Documento, as pesquisas e as atividades;
- No Protocolo, as permissões, as trocas de estados e os papéis;
- Na Interface, os templates, as páginas, e exibição condicional de páginas.

6.1 Exemplo de Desenvolvimento o framework MX

Nesta seção, será exposto um exemplo de desenvolvimento de um VCom com o framework MX. O VCom escolhido é o ambiente virtual de apoio à arquitetura pedagógica Diários Virtuais de Resolução de Problemas de Matemática. Esse VCom possui um Documento simples, um Protocolo também simples, mas pode se tornar mais complexos de acordo com a necessidade de seus usuários. Assim, depois de apresentada a implementação dessa ferramenta, será demonstrado como é o processo de alteração de funcionalidades desse VCom, e o que o torna flexível nesse aspecto.

6.1.1 Diário Virtual de Resolução de Problemas de Matemática

O Diário Virtual de Resolução de Problemas de Matemática (SERRES e BASSO, 2009) é uma espécie de revisão por pares, onde os participantes devem registrar suas soluções e comentar as soluções dos outros estudantes. É parecido com a arquitetura pedagógica "Construindo Conceituações", mas numa versão mais simples.

Um problema é proposto pelo professor em sala de aula. Os estudantes devem resolvê-lo e descrever cada passo de suas tentativas de resolução numa espécie de diário.

Em um segundo momento, é a etapa de revisões iniciada. Cada estudante deve comentar a solução de pelo menos outros dois estudantes. Serres e Basso comentaram que nessa essa etapa surpreendeu suas expectativas. Eles imaginavam que os estudantes fossem fazer isso somente por obrigação, ou de mau gosto. Mas na verdade, os estudantes tiraram bastante proveito, comentaram bastante mesmo além da exigência de dois comentários e responderam aos seus revisores. Como todos trataram do mesmo problema, essa etapa contribui para um melhor entendimento do problema proposto.

De forma semelhante aos autores da arquitetura "Construindo Conceituações", Serres e Basso utilizaram uma ferramenta de wiki para implementar essa atividade. Cada estudante possuía uma página na qual tinha que escrever seu diário de resolução do problema proposto e também receber os comentários dos revisores. Essa página possuía um modelo padrão que foi copiado para participante. Claramente isso seria uma dificuldade na implementação de outras instâncias dessa atividade e também na modificação da mesma.

A estrutura do Documento desse VCom é bem similar a de um Blog. Cada participante publica um post, que é seu diário de resolução do problema proposto, e recebe comentários de seus revisores. Na Figura 19, está representado o modelo desse

Documento, muito similar ao do Blog, apresentado na Figura 42. Nesse diagrama são utilizadas caixas para representar as UPIs e setas para representar as relações entre as UPIs

Figura 19. Diagrama do modelo de documento de um Diário de Resolução de Problemas.

Algumas atividades desse VCom são:

- Incluir Post: publica uma UPI com o rótulo "Post";
- Edita Post: edita o conteúdo da UPI publicada;
- Incluir Comentário: publica uma UPI com o rótulo "Comentário" e publica a relação desse comentário com um Post.

Algumas pesquisas desse VCom são:

- Pesquisa o diário de um usuário;
- Pesquisa comentários de um Diário.
- Pesquisa todos os diários;

Sobre o Protocolo, esse VCom possui dois estados (Inicial e Revisões), não possui grupos e tem dois papeis: participante e professor. Para cada estado, há um conjunto de permissões e esquemas de navegação definidos.

Estado Inicial. Esta é a fase em que os participantes escrevem (e editam) seus diários individualmente, descrevendo como elaboraram suas resoluções do problema proposto. Eles só podem ter acesso ao diário que eles mesmos estão produzindo, e não podem ver as respostas dos outros participantes. Dessa forma, as permissões nesse estado são:

- Um participante pode consultar a UPI Post que ele publicou;
- O professor pode consultar todo o Documento;
- Um participante pode publicar uma UPI Post, se ele já não tiver publicado nenhuma UPI Post anteriormente;
- Um participante pode alterar o conteúdo de uma UPI Post publicada;
- O professor pode mudar do estado "Inicial" para o estado "Revisões".

Nesse estado, o professor pode acompanhar como andam todos os diários, enquanto cada participante só visualiza sua própria página. Na Figura 20, é apresentado o diagrama de navegação para esse estado inicial: os participantes visualizam apenas o próprio texto de seus diários, enquanto o professor pode visualizar o texto de todos os participantes. É apresentada apenas uma página, a inicial, com templates diferentes para os dois papéis de usuários. As caixas arredondadas são as páginas, e as caixas com canto reto representam os templates. Os bonequinhos representam a quais papéis o template está associado.

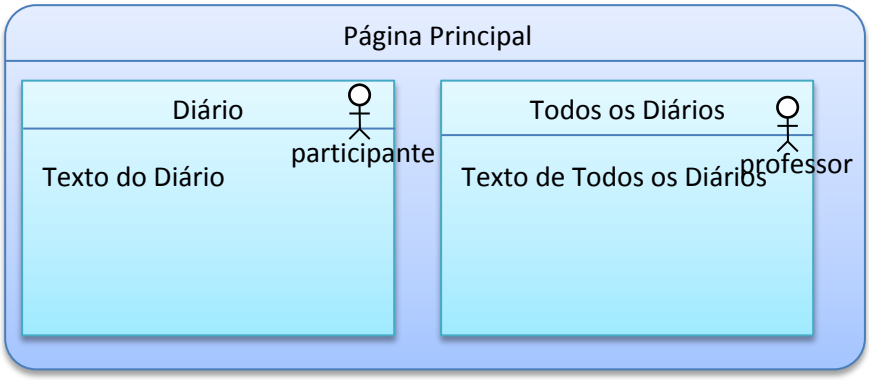


Figura 20. Diagrama de navegação da interface de um diário de resolução de problemas, no estado inicial.

Estado de Revisões. Nesse estado, os participantes podem iniciar as revisões das respostas de seus colegas, acessando uma lista de participantes e examinando o conteúdo do texto apresentado na solução de seus colegas. A partir daí, os estudantes fazem comentários das soluções dos outros, podendo iniciar uma discussão por meio desses comentários. Ao professor, também é possibilitado entrar nessa discussão e fazer comentários a qualquer um dos participantes. Algumas permissões nesse estado são:

- Qualquer um pode consultar todo o Documento;
- Um participante ou professor pode publicar Comentários, e publicar relações desse comentário com um Post;

No estado de revisões, todos visualizam o mesmo conjunto de templates, divididos em duas páginas: uma lista de diários de participantes, que possui links

para uma página ao diário de um participante específico. Na Figura 21, é apresentado o diagrama de navegação para esse estado de revisões.

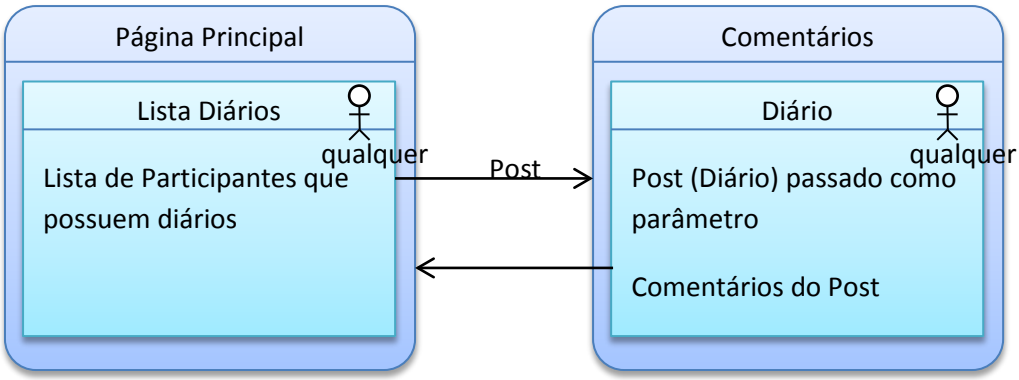


Figura 21. Diagrama de navegação da interface de um diário de resolução de problemas, no estado de revisões.

6.1.2 Implementando o Diário Virtual

Com o esqueleto do framework, o desenvolvedor deve então preenchê-lo com as definições de seu VCom.

Então, o desenvolvedor deve escrever as funções de atividades e pesquisas sobre o Documento. Nas Figura 22 e a Figura 23 é apresentado respectivamente os códigos de uma atividade e de uma pesquisa. Nota-se que elas usam operações e consultas, essas são parte do framework. As outras pesquisas e atividades possuem código semelhante.

O esquema dos dados exibido na Figura 19 não é explicitamente representado. As atividades são as responsáveis por gerar as relações entre as UPIs.

```php
public function atividadePublicaComentario($array){
    //Cria uma UPI, publica essa UPI e depois publica uma relação
    $valor = $array[0];
    $upi_id2 = $array[1];
    if($upi_id = $this->operacaoNovaUpi($valor)){
        if($this->operacaoPublicaUpi($upi_id,'comentario')){
            return $this->operacaoPublicaRelacaoUpiupi($upi_id,$upi_id2,'comentario');
        }else{
            return false;
        }
    }else{
        return false;
    }
}
```

Figura 22. Código da atividade de publicar um comentário em um diário.

```php
public function pesquisaDiario($user_id=null){
    if($user_id==null) $user_id = Yii::app()->user->id;
    $r = $this->consultaUpis_rotulo_publicador('post',$user_id);
    if(count($r)>0){
        return($r[0]);
    }else{
        return(false);
    }

}
```

Figura 23. Código da pesquisa por um diário de um usuário.

Após isso, o desenvolvedor deve escrever a função de retornar o papel de um dado usuário. O código apresentado na Figura 24 é bem simplista: o usuário que possui o *username* igual a "professor" tem o papel de professor, senão possui o usuário possui o papel de "participante". É óbvio que o desenvolvedor pode especificar critérios mais sofisticados aqui, inclusive guardar a tabela de usuários-papéis no banco.

```
public function papel($username){
    switch ($this->estado_atual){
        case "inicial":
            switch ($username){
                case "professor":
                    return "professor";
                default:
                    return "participante";
            }
            break;
    }
}
```

Figura 24. Código da função de papel de um usuário.

A Interface fica divida em duas partes do código, uma parte no Controller do VCom e nas Views que são renderizadas pelo Controller.

Os Actions do Controller no framework Yii fazem a funcionalidade de Páginas. Observa-se na Figura 25 que os templates a serem renderizados dependem do estado atual e do papel do participante. As strings “todosDiarios” e “diario” são Templates (Views), que são chamadas de acordo com o critério definido pelo desenvolvedor. As outras Páginas possuem códigos semelhantes. Na Figura 26, é apresentada uma página de outro tipo, ela executa atividades e depois redireciona para outra página.

```
//os actions são as Páginas do Morfeu
public function actionIndex()
{
    switch ($this->protocolo->estado_atual){
        case "inicial":
            switch ($this->protocolo->papel(Yii::app()->user->name)){
                case "professor":
                    $this->render('todosDiarios');
                    break;
                case "participante":
                    $this->render('diario');
                    break;
                default:
                    print "Operacao nao permitida";
                    return;
            }
            break;

        case "revisoes":
            $this->render('todosDiarios');
            break;
        default:
            print "Operacao nao permitida";
            return;
    }
}
```

Figura 25. Código da Página inicial do VCom.

```
public function actionNovoDiario()
{
    switch ($this->protocolo->estado_atual){
        case "inicial":
            switch ($this->protocolo->papel(Yii::app()->user->name)){
                case "participante":
                    $model=new Upi;

                    if(isset($_POST['Upi']))
                    {
                        $model->attributes=$_POST['Upi'];
                        if($model->validate())
                        {
                            if($this->atividade('PublicaPost',$_POST['Upi']['valor'])){
                                $this->redirect(Yii::app()->user->returnUrl);
                            }
                            return;
                        }
                    }
                    $this->render('/upi/_form',array('model'=>$model));
                    break;
                default:
                    print "Operacao nao permitida";
                    return;
            }
    }
}
```

Figura 26. Código de uma Página de execução de Atividade.

Na Figura 27, é apresentado o código do Template “todosDiarios”. Esse template usa uma pesquisa que retorna todos os diários, e gera uma página HTML que contém um link para a página do diário.

```php
<?php
/* @var $this VintController */
$this->breadcrumbs=array(
    'VInt'=>array('/'),
    'Lista de Diários',
);
$diarios = $this->pesquisa('TodosDiarios');
?>
<ul>
<?php
foreach($diarios as $diario){
    echo "<li>\n";
    echo CHtml::link($diario->user->username,array('diario','user_id'=>$diario->user_id));
    echo "<br/>\n";
    echo $diario->valor."<br/>\n";
    echo "</li>\n";
}
?>
</ul>
```

Figura 27. Código do Template “todosDiarios”.

FALEI DOS CÓDIGOS FIXOS E DOS VARIÁVEIS????

NESSE CAPÍTULO EU QUERO DESCREVER COMO É POSSÍVEL SE DESENVOLVER UM AMBIENTE VIRTUAL SEGUINDO AS IDÉIAS DO MX, USANDO BANCO DE DADOS RELACIONAIS E LINGUAGENS DE PROGRAMAÇÃO WEB COMO O PHP.

QUERO ESCOLHER UM AMBIENTE E EXEMPLIFICAR COMO É POSSÍVEL IMPLEMENTÁ-LO EFICIENTEMENTE USANDO UM MYSQL+PHP.

DEPOIS FAZER UM ESTUDO DE CASO COM USO REAL.

TALVEZ TAMBÉM EXPERIMENTOS COM TESTE DE CARGA, ESTILO OS QUE TEM NA TESE DO PAREDES.

A IDEIA PRINCIPAL DE COMO IMPLEMENTAR ISSO É A SEGUINTE:

- O BANCO VAI TER ALGUMAS POUCAS TABELAS: USUARIO, UPI (SÓ AS PUBLICADAS), RELACAO UPI-UPI, RELACAO UPI-

USUARIO, PAPEL, ATOR, ESTADOS, (VOU DEIXAR GRUPO PARA DEPOIS).

- VOU TER QUE CODIFICAR AS OPERAÇÕES SOBRE O DOCUMENTO (QUE VAI ESTAR ARMAZENADO NO BANCO). ISSO VAI FAZER PARTE DE BIBLIOTECAS PADRAO (FRAMEWORK)
- VOU TER QUE CODIFICAR, PARA A APLICAÇÃO ESPECÍFICA, AS ATIVIDADES E AS PESQUISAS
- VOU TER QUE CODIFICAR TAMBÉM AS PERMISSÕES. SERIAM FUNÇÕES QUE RETORNARIAM TRUE OU FALSE. (SE FOR PERMISSÕES DE ATIVIDADES/PESQUISAS É MAIS FÁCIL)
- TEM QUE PENSAR NUM JEITO DE FAZER O MATCHING DE QUAIS PERMISSÕES (E ETC.) SÃO PARA CADA ESTADO
- AS PÁGINAS NÃO VÃO EXISTIR REALMENTE, SÓ OS TEMPLATES. VÃO SER AQUELAS ASSOCIAÇÕES ENTRE <ESTADO,PAPEL,TEMPLATE>. A PÁGINA DEVE ESCOLHER QUAL TEMPLATE DEVE RODAR
- OS TEMPLATES SÃO FUNÇÕES QUE RETORNAM HTML.
- ESSAS IDEIAS DE IMPLEMENTAR FUNÇÕES QUE FAZEM AS COISAS VÊM DE COMO O OPENKNOWLEDGE IMPLEMENTA AS COISAS.

ALGUMAS CONCLUSÕES (SE TUDO FUNCIONAR COMO ESPERO):

- ISSO É UM FRAMEWORK (NO SENTIDO DO FRAMEWORK Yii) PARA CRIAR APLICAÇÕES WEB USANDO A ABORDAGEM MX;
  - SERAI LEGAL COMPARAR COM ESSES OUTROS FRAMEWROKS ESTILO O Yii.
- AQUELA HISTÓRIA (DO PRIMEIRO PROTÓTIPO) DE QUE É PRECISO TER UM ESQUEMA DE BANCO DE DADOS FLEXÍVEL É FURADA.
- CONSIGO IMPLEMENTAR AMBIENTES VIRTUAIS FLEXÍVEIS (COM EFICÁCIA E EFICIÊNCIA) EM BANCO DE DADOS RELACIONAIS;
  - ESSA IDÉIA SÓ FOI POSSÍVEL DEPOIS QUE IMPLEMENTEI O PROTÓTIPO EM PROLOG

- O ESTUDO DE CASO SERIA VISANDO DEMONSTRAR QUE AS ALTERAÇÕES EM TEMPO DE EXECUÇÃO SÃO POSSÍVEIS.
- NESSE ESQUEMA ATÉ SERIA POSSÍVEL FAZER COM QUE ESSAS FUNÇÕES FOSSEM ESCRITAS PELOS USUÁRIOS, MAS SERIA PRECISO UM CONTROLE MAIOR
    - SEGUINDO A IDEIA DO AMBIENTE DE CRIAR AMBIENTES DOS PRIMEIROS PROTOTIPOS, ONDE O USUÁRIO ESCREVIA CODIGO EM XSLT
    - NÃO VALE A PENA EXPLORAR MUITO, PORQUE ISSO JÁ FOI EXPLORADO. FICA PARA TRABALHOS FUTUROS.
- TERIA QUE COMPARAR ISSO COM O TRABALHO DO ELEU NATALI (EU ACHO) QUE TAMBÉM IMPLEMENTOU UM FRAMEWORK DO MORFEU.

POSSÍVEIS ESCOLHAS DE AMBIENTES PARA SER IMPLEMENTADOS (QUERO IMPLEMENTAR SÓ UM DELES):

- DOMINÓ: UM EXEMPLO COMPLEXO DO PONTO DE VISTA DO PROTOCOLO DE INTERAÇÃO. NÃO QUERO ESCOLHER ELE, PORQUE SERIA ESTRANHO ESTUDAR AS MODIFICAÇÕES EM TEMPO REAL;
- DEBATE DE TESES: ESSE TEM CERTO GRAU DE COMPLEXIDADE NO DOCUMENTO E NO PROTOCOLO DE INTERAÇÃO E DARIA PARA ESTUDAR BEM AS MODIFICAÇÕES EM TEMPO REAL. O POSSÍVEL PROBLEMA É QUE PODE DEMORAR UM POUCO PARA IMPLEMENTAR, MAS ACHO QUE VALE A PENA;
- CONTROVÉRSIA ACADEMICA: DOCUMENTO SIMPLES, PROTOCOLO COM COMPLEXIDADE MÉDIA, E INTERFACES SIMPLES, PORÉM PODENDO EXPLORAR BEM AS POSSIBILIDADES. TERIA QUE PENSAR NUMA FORMA DE ESTUDAR AS ALTERAÇÕES EM TEMPO REAL.
- DIARIO VIRTUAL: DOCUMENTO SIMPLES, PROTOCOLO SIMPLES DEMAIS. MAS AS POSSIBILIDADES DE ALTERAÇÃO EM TEMPO REAL PODEM SER BEM EXPLORADAS, INCLUSIVE DEIXANDO O PROTOCOLO MAIS COMPLEXO.

- SE PENSARMOS NO CENÁRIO DE INTRODUÇÃO À COMPUTAÇÃO QUE O PROFESSOR TEM DISCIPLINA ESSE SEMESTRE, EU ACHO QUE TEMOS UMA BOA ESCOLHA
- AÍ TERÍAMOS QUE MUDAR A APLICAÇÃO ORIGINAL:
  - OS ALUNOS TERÍAM QUE DESCREVER COMO RESOLVERAM O PROBLEMA. ELES RECEBERIAM COMENTÁRIOS DE OUTROS ALUNOS NO DECORRER DO PROCESSO.
  - ALGO COMO A PRIMEIRA ETAPA DO DEBATE DE TESES
  - NESSE CASO, CADA ESTUDANTE (OU GRUPO) PODERIA TER UM PROBLEMA DIFERENTE
  - ELES PODERIAM SER FORÇADOS A FAZER COMENTÁRIOS
  - TERÍAMOS QUE PENSAR MODIFICAÇÕES QUE PODERÍAM SER FEITAS EM TEMPO REAL

7 Protótipo do Jogo de Dominó

Foi feito um protótipo do MX com intuito de avaliar suas características complexas do Protocolo de Interação e também características de Agentes Reativos simples que implementavam gatilhos (*triggers*). Para observar essas características foi implementado um Jogo de Dominó utilizando o Prolog. Nessa implementação, somente o Documento e o Protocolo foram objetos de estudo. Obtiveram-se resultados positivos, com a prova de conceito de características complexas do Protocolo de Interação e sucesso na factibilidade dos Agentes Reativos e os gatilhos.

Num modelo muito parecido com o protótipo apresentado no capítulo anterior, o código possui uma parte fixa que implementa as funcionalidades de acesso ao Documento e as rotinas do Protocolo e em outra parte do código são implementadas as permissões do VCom Jogo de Dominó. Os dados do Documento são inseridos com operações *assert* e retirados com *retract* na base de conhecimento. Os dados de estado atual pertencentes ao Protocolo sofrem o mesmo comportamento.

Na seção a seguir, é apresentado o problema do Jogo de Dominó e como ele foi modelado como um VCom do MX. Depois são apresentados detalhes da implementação em Prolog, finalizando com uma discussão das observações realizadas.

7.1 Jogo de Dominó

Dominó é um jogo que utiliza pequenas pedras (*bones*) divididas ao meio por uma linha, onde cada lado (ponta) possui um número entre zero e seis, totalizando 28 pedras. No estado do Amazonas, no Brasil, ele é um jogo tradicional que é jogado com quatro pessoas, duas duplas, sentadas ao redor de uma mesa. As pedras são encaixadas uma nas outras de maneira que tenham pontas iguais. O objetivo do jogo

é fazer mais pontos que a outra dupla. Os pontos são feitos quando a soma das pontas que estão sem encaixar é múltiplo de cinco.

Há alguns anos havia um site que possuía esse jogo para ser jogado on-line com outras pessoas, o DominóNet[19]. Mas infelizmente ele foi descontinuado no ano de 2004[20] e atualmente a sua página inicial apresenta somente a frase "EM CONSTRUÇÃO". E esse tipo de ambiente virtual que estamos tratando nesta seção, um ambiente virtual para jogar dominó on-line. Um conjunto de regras pode ainda ser acessado através de um serviço do site Archive.org (WAYBACK MACHINE, 2003).

No jogo tradicional, os quatro jogadores sentam-se ao redor de uma mesa, intercalando-se os membros das duplas. No início da rodada, cada jogador recebe aleatoriamente sete pedras. As pedras com as duas pontas iguais são chamadas carroças, e começa o jogo quem possuir a carroça de seis (também chamada de sena), ou seja, a pedra cujas pontas são $\langle 6|6 \rangle$. O jogo segue em sentido horário, sendo o próximo que joga é aquele sentado ao lado esquerdo do jogador que acabou de jogar.

Uma pedra só pode ser jogada se há uma ponta de alguma das pedras que já foram jogadas na mesa sem um encaixe (chamada de ponta de mesa) e que essa ponta sem encaixe possa ser encaixada em uma das pontas da pedra que vai ser jogada.

Na Figura 28, apresenta-se uma foto de uma mesa de dominó, com as pedras encaixadas corretamente. As carroças são encaixadas na perpendicular, e valem o dobro de pontos quando estão em uma ponta de mesa. Na foto, as pontas de mesa são o

[19] http://www.dominonet.com.br
[20] http://web.archive.org/20030603041953/http://www.dominonet.com.br/

6 da pedra $\langle 2|6 \rangle$ e o 5 da pedra $\langle 2|5 \rangle$. Excepcionalmente, a carroça inicial pode receber até quatro encaixes.

Figura 28. Uma mesa de dominó. Fonte: www.photaki.com.

Um passe ocorre quando um jogador não possui nenhuma pedra em sua mão que seja possível jogar na mesa. Nesse caso, a dupla adversária ganha 20 pontos.

Uma rodada acaba quando algum dos jogadores acaba as pedras que estão em sua mão, quando é dito que esse jogador "bateu", ou se nenhum dos quatro jogadores consegue fazer nenhuma jogada, quando é dito que "fechou o jogo". A nova rodada se inicia como jogador que bateu na jogada anterior, e ele deve jogar uma carroça. Se ele não possuir nenhuma carroça para jogar, então ele passa.

Um jogador faz ponto quando joga uma pedra na mesa e soma das pontas da mesa é um múltiplo de 5. Por exemplo, na Figura 28, o jogador não nenhum ponto, pois as pontas são 5 e 6, que somadas são 11, que não é múltiplo de 5.

Ganha o jogo a dupla que no final de uma rodada possuir mais pontos que equipe adversária e tiver atingido o limiar pré-estabelecido. Esse limiar de pontos normalmente é 100 ou 200 pontos.

Num jogo de dominó as pedras são sempre as mesmas, logo o ambiente virtual deve ter essas pedras já pré-publicadas. Além das pedras, as UPIs desse ambiente virtual são os pontos e as jogadas realizadas, formando um log de jogadas. Há também as relações entre as pedras e os jogadores, ou seja, a mão dos jogadores é

representada através dessas relações. Na Figura 29, é apresentada um diagrama do documento desse VCom.

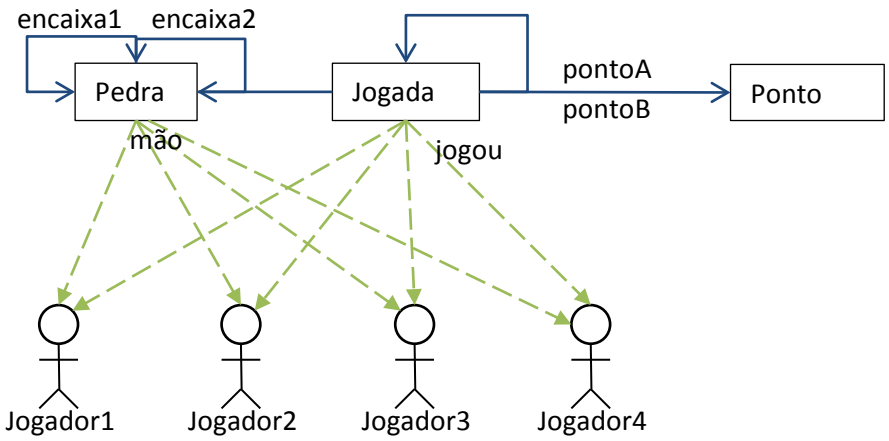


Figura 29. Diagrama do modelo de documento de um jogo de dominó.

A UPI "Jogada" não possui conteúdo, ela é uma UPI do tipo nulo. Ele serve para estruturação de outros elementos do Documento, nesse caso, a jogada promove uma sequência das pedras que foram jogadas e por quem foi jogada, que no caso é o autor da UPI "Jogada".

Esse VCom tem uma particularidade, ele possui UPIs pré-publicadas, as pedras. Elas são sempre as mesmas e não é possível jogar com menos, mais, ou pedras diferentes dessas. Dessa forma, quando um jogo de dominó é instanciado, as UPIs de pedras já estão publicadas no documento.

Algumas atividades desse VCom são:

- Embaralhar as pedras: atribuir aleatoriamente as pedras para os jogadores, publicando uma relação como rótulo "mão" entre as pedras e os jogadores;
- Jogar Pedra: Publica uma relação de encaixe com a pedra da ponta da mesa, publica uma UPI nulo com rótulo "Jogada", publica relação dessa jogada com a jogada anterior, publica uma relação entre esse UPI de jogada com a pedra jogada, processa os pontos e publica os devidos pontos da jogada e um relação da UPI pontos com essa jogada.

O Protocolo de Interação desse VCom é bem complexo, pois não pode dar margem a jogadas erradas, em horas erradas, ou fazendo algo que não é permitido no

jogo. Além disso, precisa controlar as rodadas e a pontuação do jogo, de forma a declarar o vencedor.

Esse jogo possui um sistema de estados bem definidos. Depois de embaralhar as pedras entre os jogadores, o jogador que possui a carroça de sena deve começar o jogo. Após isso, um após o outro, os jogadores vão jogando seguindo sempre essa mesma ordem de que "eles sentaram na mesa". Após cada rodada, é necessário re-embaralhar as pedras e começar uma nova jogada. O Protocolo ainda precisa dar conta de quando acaba o jogo e quem foi o vencedor da partida. Na Figura 30, é apresentado um diagrama de estados ilustrando essas mudanças de estados.

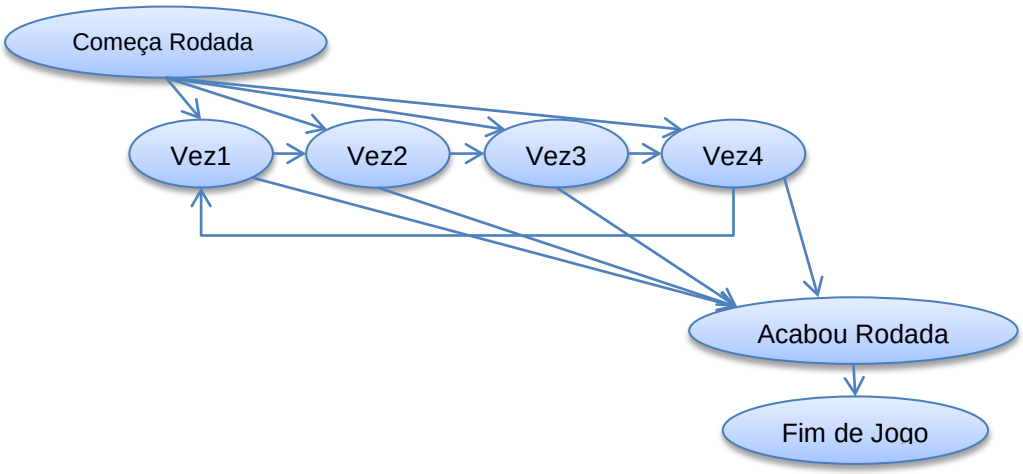


Figura 30. Diagrama de estados de um jogo de dominó.

Para cada estado há um conjunto diferente de permissões, de modo que somente o jogador da vez possa jogar. Cada jogador, nesse caso, assume um papel entre os quatro papéis possíveis. O Protocolo então controla as jogadas possíveis.

Mas existem permissões que se aplicam a qualquer estado do VCom, são elas:

- Os jogadores podem consultar as todas as UPIs Pedra;
- Os jogadores podem consultar suas mãos, ou seja, consultar as relações UPI-Usuário entre UPIs Pedra e ele mesmo;
- Os jogadores podem consultar todas as jogadas, as relações entre as jogadas, e as relações entre jogadas e pedras;
- Os jogadores podem consultar os Pontos e as relações entre pontos e jogadas;

- Qualquer pessoa (inclusive os espectadores) podem consultas a mesa de jogo, ou seja, as pedras e as relações de encaixa1 e encaixa2 entre as pedras.

No primeiro estado, **começa rodada**, algum jogador precisa embaralhar as pedras. Há um trigger que dispara a atividade de embaralhar as pedras. Essa atividade associa aleatoriamente uma pedra com um jogador, através de uma relação "mão", de forma que cada jogador fique com sete pedras na mão.

Feito isso, há outro gatilho que é ativado mediante uma condição: a de que todos os jogadores tenham sete pedras na mão. Esse gatilho muda o estado para a vez do jogador que possui a carroça de sena. Na verdade, trata-se de quatro gatilhos, um gatilho para cada jogador, verificando se todos possuem sete pedras e se ele possui a carroça de sena. Quando esse gatilho é ativado ele muda o estado da vez do jogador.

Os estados de **vez do jogador**, são muito parecidos entre si. Possuem permissões que permitem que um jogador, o jogador da vez, jogue uma pedra na mesa, e acionam os gatilhos de passe e pontos. As permissões desse tipo de estados são as seguintes:

- O jogador pode despublicar a relação da Pedra com a mão e relacionar essa Pedra com a Jogada, se essa pedra for a carroça de sena;
- O jogador pode encaixar uma pedra na outra (publicar uma relação de encaixa1 ou encaixa2 entre duas Pedras, publicar uma jogada e associar essa pedra com a jogada), se a ponta que não possui encaixe da pedra que está na mesa for igual a uma das pontas da pedra a ser jogada;
- Há uma exceção para a regra anterior para a carroça inicial, que pode ter até quatro encaixes (dois encaixes no "bucho"), enquanto as outras pedras só podem ter dois encaixes, uma para cada ponta;
- O jogador também pode jogador outra carroça (sem ser a carroça de sena), se ele tiver batido na jogada anterior;

Há ainda gatilhos que publicam os pontos. Eles verificam se a soma das pontas sem encaixe das pedras que estão na mesa é múltiplo de cinco. Se for, publica uma

UPI Ponto (A ou B, respectivamente para cada dupla) com a quantidade de pontos marcada naquele jogada, e a associa com a última jogada.

Se a última jogada for do jogador da vez, então é acionado um gatilho de mudança de estado, passando para o estado da vez do próximo jogador.

Um terceiro gatilho desse estado é em relação a passes. Se o jogador não possui nenhuma pedra que ele possa jogar na mesa, então é publicada uma jogada sem associação com nenhuma pedra, e é atribuído vinte pontos para a dupla adversária associada com essa jogada. Esse trigger também faz com que o estado seja mudado para a vez do próximo jogador. Esse gatilho ainda precisa verificar se a última jogada foi também de passe, pois isso é considerado "passe nas costas" e não devem ser contabilizados pontos para ninguém.

Outro gatilho verifica se o jogador da jogada anterior não possui nenhuma pedra na mão, sinalizando que a rodada acabou. Esse gatilho, então, muda o estado para o estado de acabou rodada. Se não há nenhuma pedra possível de ser jogada, ou seja, o jogo está fechado, então esse gatilho também deve ser acionado.

No estado de **acabou rodada**, é verificado se o jogo acabou ou se é preciso mais uma rodada.

O jogo acaba quando uma das duplas atinge a uma determinada soma de pontos (geralmente 100 ou 200 pontos) ao final de uma rodada. Se isso acontece deve ser acionado um gatilho que muda o estado para o estado de **fim de jogo**. Ganha a dupla que tiver a maior soma dos pontos.

Se isso não ocorrer e o jogo não foi fechado, é acionado um gatilho que re-embaralha as pedras e as distribui entre os jogadores. Esse gatilho também é responsável por publicar uma jogada especial marcando que um dos jogadores bateu, e por mudar o estado para a vez de um dos jogadores, para aquele que bateu a rodada anterior.

Se não foi fim de jogo e o jogo foi fechado, outro gatilho é acionado. Ele deve re-embaralhar e distribuir as pedras e fazer com se troque de estado para o jogador que possuir a carroça de sena.

A Interface desse VCom é relativamente simples, é a mesma para qualquer dos estados. Para os jogadores é executado o template que exibe a mesa de jogo, sua mão e as jogadas anteriores. Para os espectadores, é executado o template que exibe a mesa de jogo e as jogadas anteriores. Isso foi representado na Figura 31. Nesse protótipo, no entanto, as funcionalidades de Interface não foram implementadas.

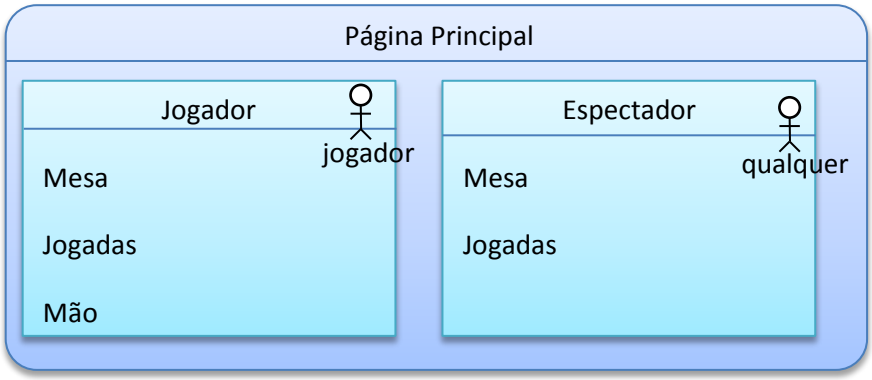


Figura 31. Diagrama de navegação da interface de um jogo de dominó.

7.2 Implementação do Protótipo em Prolog

Esse protótipo foi implementado utilizando-se três bases de conhecimentos diferentes:

- Base de Conhecimento do Documento (KB-Doc): uma base de conhecimento que armazena dados sobre o Documento do VCom;
- Base de Conhecimento do VCom (KB-VCom): uma base de conhecimento sobre VComs em geral;
- Base de Conhecimento do Jogo de Dominó (KB-Dominoes): possui o conhecimento específico sobre o VCom Jogo de Dominó, ou seja, que possui o conhecimento sobre as permissões do Protocolo.

A base de conhecimento contendo os dados do Documento é dinâmica, seus conceitos são inseridos e retirados com *assert* e *retract*, e seu conhecimento trata de dados sobre as UPIs publicadas e suas relações.

A base de conhecimento sobre VComs é estática, não necessitando de alterações, podendo ser reutilizada para a construção de qualquer VCom. Ou seja, nesse arcabouço para construção de VComs, ela é um ponto fixo.

Na terceira base de conhecimento, são expressos o conhecimento sobre o VCom Jogo de Dominó (ou o VCom que se esteja implementando), nele são expressas as permissões, as atividades e as pesquisas do Protocolo.

O VCom, portanto, necessita das três bases de conhecimento para funcionar.

7.2.1 KB-Doc: Base de Conhecimento do Documento

Os dados (fatos) do Documento são inseridos com operações *assert* e retirados com *retract* na base de conhecimento que armazena os dados de UPIs e Publicações. As regras que inserem esses fatos na base de conhecimento sobre do Documento estão presentes na base de conhecimento de VComs.

Na versão atual desse protótipo, essa base de conhecimento armazena três tipos de fatos:

- `publishedUPI/1`: upi publicada, onde o parâmetros é uma UPI publicada, no formato de uma tupla (Conteúdo, Rótulo, Publicador).
- `relationUPI/4`: relações entre duas UPIs, onde os parâmetros são duas UPIs publicadas, o rótulo da relação e o publicador dessa relação.
- `relationUPIUser/4`: relações entre UPIs e usuários, onde os parâmetros são uma UPI publicada, um usuário, o rótulo da relação e o publicador dessa relação.

Há ainda outros dados que são armazenados no Documento que são previsto pelo arcabouço MX, no entanto, esses outros tipos de relações não foram necessárias para o funcionamento desse protótipo.

Especificamente para o Jogo de Dominó, é necessária uma publicação prévia de UPIs, antes mesmo de iniciar sua utilização, são as pedras do jogo. São as 28 pedras do jogo de dominó, que são publicadas com o rótulo "pedra", porém não possuem usuário publicador. Esse é um exemplo de uma dessas pedras: `publishedUPI(((3, 5), pedra, ''))`.

Dessa forma, um VCom possui Documento inicial, que é replicado para cada nova instância desse VCom.

Além, dos dados do Documento, essa base de conhecimento guarda também fatos dinâmicos de responsabilidade do Protocolo, como o estado atual e algumas variáveis que guardam valores globais.

7.2.2 KB-VCom: Base de Conhecimento de VCom

Nessa base de conhecimento são implementadas as operações de alteração e de consulta ao Documento.

Um exemplo de operação de alteração do Documento é a operação de publicar uma UPI. Na Figura 32, é apresentado o código responsável por essa funcionalidade. Para se publicar uma UPI é preciso perguntar se é permitido ao usuário publicá-la no estado atual (`canPublishUPI`) e se já não existe uma UPI publicada igual. A regra de permissão de publicar UPI `canPublishUPI` deve ser obrigatoriamente implementada na KD-Dominoes. Depois dessas condições serem satisfeitas, um novo conhecimento é incluído na base de conhecimento (`assert`) e salvo no arquivo da KB-Doc, para assegurar a persistência desses dados.

```
publishUPI(UPI):-
        currentState(S),
        canPublishUPI(S,UPI),
        \+publishedUPI(UPI),
        assert(publishedUPI(UPI)),!,
        saveFile.
```

Figura 32. Código em Prolog da operação de publicar uma UPI.

Além, da operação de publicar uma UPI, há também as operações de publicar uma relação entre duas UPIs e entre uma UPI e um usuário. Ambas possuem funcionamento muito similar à operação de publicar uma UPI.

A KB-VCom também disponibiliza operações de consulta ao Documento. Na Figura 33, é apresentado o código da regra `queryUPI` que retorna em seu segundo parâmetro uma lista com todas as UPIs publicadas por um usuário. Porém, há uma restrição especial, somente as UPIs que são permitidas de ser feita operações de consulta pelo usuário que está solicitando essa consulta são retornadas na resposta. Essa verificação é feita pela regra `canQueryUPI`, que deve ser obrigatoriamente implementada na KB-Dominoes.

```
queryUPI(User,R):-
        currentState(S),
        all(
                X,
                (
                        publishedUPI(X),
                        X=(_Content,_Label,_Publisher),
                        canQueryUPI(S,User,X)
                ),
                R
           ).
```

Figura 33. Código em Prolog para a operação de consulta de UPIs publicadas por um usuário.

Todas as consultas e alterações no Documento são feitas através das operações implementadas nessa base de conhecimento.

7.2.3 KB-Dominoes: Base de Conhecimento do Jogo de Dominó

Na base de conhecimento específica para um VCom, o desenvolvedor deve implementar: as permissões de publicações e operações de alteração e consulta ao Documento; as Pesquisas e as Atividades; e os gatilhos (*triggers*) de troca de estado.

Ou seja, programar um VCom é basicamente programar as condições das permissões, as pesquisas e as atividades e as trocas de estado. Além disso, ainda teria que programar as páginas e templates, que não foram alvos de estudo nesse protótipo, onde buscou-se analisar propriedades complexas do Protocolo.

Diferente do protótipo do capítulo anterior, que implementava permissões apenas no nível de páginas e atividades, nesse protótipo em Prolog buscou-se a exploração das características de permissões complexas no nível de operações de escrita e consultas no Documento.

Além desse aspecto, buscou-se também a exploração de gatilhos (*triggers*), agentes reativos simples. Foram implementados dois tipos de gatilhos para o Dominó, os gatilhos de troca de estados e os gatilhos de contagem de pontos.

7.2.3.1 Papéis, Atores e Estados

É preciso definir quais usuários (atores) interpretam quais papéis. E qual o estado inicial da aplicação. Ambas as informações são usadas para definir as permissões, que são condicionadas pelo papel do usuário que tenta realizar uma operação e pelo estado atual da aplicação.

Os fatos sobre os papéis são representados apenas com uma lista de papéis, como apresentado na Figura 34.

```
role(jogador).
role(jogador1).
role(jogador2).
role(jogador3).
role(jogador4).
```

Figura 34. Código em Prolog para a representação de papéis.

A associação entre usuários e os papéis produz a relação de atores, que devem ser listados da mesma forma. Na Figura 35, é apresentado o código que descreve que o papel dos quatro usuários é "`jogador`" e que cada um dos jogadores possui um papel de jogador específico: "`jogador1`", "`jogador2`", "`jogador3`", "`jogador4`". Além dessa maneira apresentada, ainda é possível haver atribuições de papéis condicionais, como

por exemplo, um usuário possui um papel somente em um dado estado, ou é possível descrever que se um usuário interpreta algum papel, ele automaticamente interpreta algum outro.

```
actor(jogador1,user1).
actor(jogador2,user2).
actor(jogador3,user3).
actor(jogador4,user4).

actor(jogador,user1).
actor(jogador,user2).
actor(jogador,user3).
actor(jogador,user4).
```

Figura 35. Código em Prolog para a definição de atores.

Além dos papéis e atores é preciso definir qual o estado inicial. Para isso, faz-se uso do fato `initialState`. Os outros possíveis estados do VCom devem ser definidos de acordo com a troca de estados. Nesse método é possível ter um ou mais estados iniciais e um ou mais estados atuais.

7.2.3.2 Permissões

O comportamento do VCom é dado principalmente por suas permissões. As permissões são invocadas no momento de execução das operações de alteração e de consulta ao Documento.

Na Figura 32 e na Figura 33, podem-se observar duas verificações de permissões: `canPublishUPI` e `canQueryUPI`. As duas operações somente são realizadas se essas permissões forem verdadeiras na base de conhecimento, no caso a KB-Dominoes.

Na Figura 36, é apresentado um código que define uma permissão de consulta de UPIs com rótulo "`pedra`", em qualquer momento (estado) do jogo, quando o usuário que solicitou uma consulta de UPIs interpretar o papel "`jogador`". Essa regra tem três parâmetros: um estado, um usuário, e uma UPI. De acordo com as condições descritas essa permissão é aprovada ou não.

```
%os jogadores podem consultar todas as pedras do jogo
canQueryUPI(_S,User,(_Content,pedra,_Publisher)):-
        actor(jogador,User).
```

Figura 36. Código Prolog que define que todos os usuários podem a qualquer momento consultar todas as 28 pedras do jogo.

Na Figura 37, é apresentado um código de permissão para jogar uma pedra na mesa. Trata-se de uma permissão para relacionar duas UPIs. A regra `canRelateUPIs` possui cinco parâmetros: um estado, a UPI da pedra a ser jogada, a UPI da pedra da ponta da mesa, um rótulo da relação (qual dos dois lados da pedra será encaixado) e o publicador. Essa permissão trata se a pedra a ser jogada possui uma das pontas igual a umas das pontas da pedra de ponta da mesa, incluindo se a ponta da mesa for à carroça inicial.

```
%pode jogar pedra
canRelateUPIs(s1, ((PontaA1,PontaA2), pedra, _Pub1), ( (PontaB1,PontaB2), pedra, _Pub2), Label, User):-
        actor(jogador1,User),
        %encontra qual foi a ponta encaixada anteriormente ela pedra que será referenciada
        (
                (
                        queryUPIRelatedBy_Label(( (PontaB1,PontaB2), pedra, _Pub2),encaixa1,User,R),
                        R \= [],
                        PontaB = PontaB2
                );
                (
                        queryUPIRelatedBy_Label(( (PontaB1,PontaB2), pedra, _Pub2),encaixa2,User,R),
                        R \= [],
                        PontaB = PontaB1
                );
                %encaixar na carroça inicial
                (
                        queryUPIRelatedBy(( (PontaB1,PontaB2), pedra, _Pub2),User,R),
                        R = [],
                        %a carroça inicial não pode ter mais de quatro encaixes
                        all(
                                ((P1,P2), pedra,_PubP),
                                (
                                        %upis que relacionam e são pedra
                                        queryUPIRelate( ((PontaB1,PontaB2), pedra, _Pub2), User, R2),
                                        member( ((P1,P2), pedra,_PubP), R2)
                                ),
                                R3
                        ),
                        length(R3,Tam),
                        Tam=<4,
                        PontaB = PontaB1
                )
        ),
        %encaixa ponta com ponta
        (
                (
                        Label = encaixa1,
                        PontaA1 = PontaB
                );
                (
                        Label = encaixa2,
                        PontaA2 = PontaB
                )
        ).
```

Figura 37. Código Prolog de permissão de jogar uma pedra na mesa.

Cada estado possui um conjunto dessas permissões, que muitas vezes somente são diferentes em qual o papel permitido a ser realizada a operação. Ou seja, em cada estado, somente pode jogar o jogador que está na vez. Dessa forma, o VCom toma forma, descrevendo suas permissões em cada estado.

7.2.3.3 Pesquisas e Atividades

Os usuários finais somente poderão interagir com o VCom através das Pesquisas e Atividades. Mais precisamente com Páginas e Templates que usam as Pesquisas para exibir resultados e Páginas de Atividades que modificam o Documento.

Na Figura 38, é apresentado um código que uma Pesquisa utilizada pelo Agente Gatilho que conta a quantidade de pontos na mesa. Seus parâmetros são o usuário solicitante e a resposta da consulta. Essa consulta verifica dentre todas as pedras jogadas na mesa aquelas que possuem apenas uma relação, ou seja, que ainda estejam com uma ponta livre.

```
%consulta quais são as pedras das pontas pra poder contar quantos pontos tem na mesa
queryPedrasPonta(User,PedrasPonta):-
        queryUPI_Label(User,jogada,Jogadas),
        all(Pedra,(member(Jogada,Jogadas),queryUPIRelatedBy(Jogada,User,[Pedra|_])),PedrasJogadas),
        all(P,(member(P,PedrasJogadas), queryUPIRelate(P,User,[_P2])  ),PedrasPonta).
```

Figura 38. Código Prolog de uma Pesquisa.

Uma Atividade é uma sequência de operações de alteração do Documento. Na Figura 39, é apresentado da atividade `jogaPedra`, que tem três parâmetros: o usuário solicitante da atividade, a pedra (UPI) a ser jogada e a ponta da mesa (UPI) que será jogada essa pedra. Essa atividade realiza uma série de operações: publica a relação de `encaixa` entre as duas UPIs; publica a "`jogada`" e associa com a pedra jogada; associa a "`jogada`" com a "`jogada`" anterior, para manter uma lista das jogadas realizadas; e finalmente tira a pedra jogada da mão do usuário, desfazendo a relação desse usuário com essa pedra.

```
%jogarPedra: se pode jogar, joga uma pedra numa ponta (UPI)
jogaPedra(User,Pedra,PontaMesa):-
        %publica o encaixa
        (
                relateUPIs(Pedra, PontaMesa, encaixa1, User);
                relateUPIs(Pedra, PontaMesa, encaixa2, User)
        ),!,
        %publica jogada
        variableValue(numero_jogada,NJ),
        publishUPI((NJ, jogada, User)),
        %associa a jogada com a pedra jogada
        relateUPIs((NJ, jogada, User), Pedra, '', User),
        %associa jogada com a jogada anterior
        NJa is NJ-1,
        queryUPI_Label(User, jogada, R),
        member((NJa,jogada,PubJogadaAnterior),R),
        relateUPIs((NJ, jogada, User), (NJa,jogada,PubJogadaAnterior), '', User),
        %muda o numero da jogada
        NJ2 is NJ+1,
        attribVariableValue(numero_jogada,NJ2),
        %tira pedra da mão
        unrelateUPIUser(Pedra,User,mao,_PubMao).
```

Figura 39. Código Prolog de uma Atividade.

7.2.3.4 Gatilhos

O VCom Jogo de Dominó possui dois tipos de gatilhos: os gatilhos para mudar de estado automaticamente e os gatilhos para a contagem automática de pontos.

O gatilho de troca de estados é um gatilho especial que verifica condições para troca de estados ao final de cada consulta às bases de conhecimento. De forma parecida com uma permissão, foi implementado um axioma que verifica se deve ser feita uma troca de estado. Na Figura 40, é apresentado um código com o axioma `changeState`, que possui dois parâmetros: o estado atual e o estado para o qual deve ser mudado o estado. Se a condição for aceita, o sistema troca de estado: retira o estado atual da lista de estados atuais (`retract`) e coloca o próximo estado nessa mesma lista (`assert`). Essa regra verifica se a última jogada foi realizada pelo usuário da vez, no caso um ator do papel "`jogador1`". O sistema então verifica a cada consulta à base de conhecimento se é possível realizar alguma troca de estado e o faz se for possível.

```
%jogou normal -> passa pro próximo jogador
changeState(s1,s2):-
        actor(jogador1,User),
        %existe uma UPI jogada que não é relacionada por ninguém (a último jogada)
        queryUPI_LabelPublisher(User,jogada,User,Jogadas),
        all(
                J,
                (
                        member(J,Jogadas),
                        queryUPIRelate(J,User,R1),
                        R1=[]
                ),
                R
        ),
        R=[UltimaJogada|_T],
        %se a ultima jogada foi do User
        UltimaJogada = (_Content, jogada, User).
```

Figura 40. Código Prolog para um gatilho de troca de estado.

Há a possibilidade de implementar outros tipos de gatilhos. Na Figura 41, é apresentado um código para um gatilho de contagem de pontos. Esses gatilhos são executados antes das trocas de estados. E diferente das trocas de estados que são implementados apenas como permissões de troca de estados, com os gatilhos é preciso implementar também a execução das operações desejadas. No gatilho de pontos da Figura 41, é verificado se a última jogada foi do usuário da vez (de acordo com o estado atual) e se há pontos múltiplos de 5 na mesa. Atendendo às condições, são executadas a publicação de UPI de pontos, relacionando-a com a última jogada.

```
%Trigger dos pontos
trigger(s1):-
        actor(jogador1,User),
        %existe uma UPI jogada que não é relacionada por ninguém (a último jogada)
        queryUPI_LabelPublisher(User,jogada,User,Jogadas),
        all(
                J,
                (
                        member(J,Jogadas),
                        queryUPIRelate(J,User,R1),
                        R1=[]
                ),
                R
        ),
        R=[UltimaJogada|_T],
        %se a ultima jogada foi do User
        UltimaJogada = (_Content, jogada, User),
        %tem pontos na mesa?
        queryPontosMesa(User,Pontos),
        X is Pontos mod 5,
        X=0,
        %publica os pontos
        publishUPI((Pontos,pontos,User)),
        %relaciona a jogada com os pontos
        relateUPIs(UltimaJogada, (Pontos,pontos,User), pontosA,User).
```

Figura 41. Código Prolog para um gatilho de contagem de pontos.

7.3 O protótipo funcionando

7.4 Conclusão do Capítulo

DISCUSSÃO: COMO MODELAR UMA SITUAÇÃO DINÂMICA QUE TEM QUE TER CONTROLE EXTERNO ALÉM DOS AUTORES.

O que é possível fazer com o protótipo e se ele roda de verdade.

O que ele não tem em relação à proposta do MX.

Fazer link com o capítulo do protótipo web.

8 Análise de Escopo

Neste capítulo, é apresentada uma análise sobre o escopo da proposta apresentada. Será discutido o escopo do MX comparativamente ao Morfeu original, o escopo da proposta conceitual comparando-a com as implementações apresentada, e uma análise sobre a escala de aplicação desses conceitos. Assim, neste capítulo será discutido o que é possível ser modelado e desenvolvido (ou não) com o MX.

8.1 Comparativo de escopo entre MX e MOrFEu

O principal motivo da extensão do MOrFEu com o MX é a falta de adequabilidade do MOrFEu em modelar e implementar alguns tipos de ambientes virtuais comuns e também outros mais complexos. Assim, o MX propôs uma abstração em nível mais geral que o MOrFEu para poder conseguir resolver tal problema. Essa generalização dos elementos de modelagem de ambientes virtuais tornou o MX capaz de modelar outros tipos de ambientes virtuais que não eram considerados pelo escopo do MOrFEu original.

FALAR DAS DEFICIÊNCIAS DO MORFEU. COMO ELE IMPLEMENTAVA COM GAMBIARRAS AS COISAS: ESTADOS; RELAÇÃO UPI-USUÁRIO; COMPORTAMENTO CONDICIONAL DAS PÁGINAS; DEFINIÇÕES FLEXÍVEIS DE PERMISSÕES E FUNCIONALIDADES.

COMPARAR A MODELAGEM DO MX COM O DO MORFEU:

- DOCUMENTO: HIERÁRQUICO VS RELAÇÃO UPI-UPI + RELAÇÃO UPI-USUÁRIO.
- PAPÉIS: ERAM RESTRITOS: AUTORES E LEITORES.
- INTERFACE: FUNCIONAMENTO CONDICIONAL DAS PÁGINAS
- ESTADOS: NÃO EXISTIAM.

- PERMISSÕES: NÃO EXISTIAM.

FALAR QUE O ESCOPO DO MORFEU ERAM DE EDUCAÇÃO E AMBIENTES COLABORATIVOS.

FALAR DOS AMBIENTES QUE O MX CONSEGUE MODELAR AMBIENTES VIRTUAIS QUE NÃO SÃO NEM COMUNS NEM EDUCACIONAIS: BOLÃO DA COPA DO MUNDO; DOMINÓ; E FACEBOOK. RELACIONAR ESSES AMBIENTES COM AMBIENTES EDUCACIONAIS, REAIS OU SUPOSIÇÕES.

FALAR DE COMO O DOMINÓ FOI UMA EXPERIÊNCIA PARA DETALHAR O PROTOCOLO DE INTERAÇÃO. E COMO ESSE TIPO DE AMBIENTE É PARECIDO COM OUTROS AMBIENTES VIRTUAIS, SEJAM EDUCACIONAIS OU NÃO.

FALAR DO BOLÃO DA COPA QUE NÃO É BEM UM AMBIENTE VIRTUAL MAS QUE TAMBÉM TEM UMA SÉRIE DE REGRINHAS.

QUAL A CONCLUSÃO? QUE O MX É CAPAZ DE MODELAR QUE TIPOS DE SISTEMAS?

8.2 Análise do escopo das implementações com o MX

FAZER A ANÁLISE DE COMO AS IMPLEMENTAÇÕES INFLUENCIARAM NA CONCEPÇÃO DO MX (PRINCIPALMENTE O DOMINÓ) E DE COMO AS IMPLEMENTAÇÕES REFLETEM OS CONCEITOS DO MX.

CONTAR A HISTÓRIA DO DESENVOLVIMENTO DO DOMINÓ E DE COMO A CONCEPÇÃO CONCEITUAL DO MX FOI FEITA EM NÍVEL CONCEITUAL E NA IMPLEMENTAÇÃO AO MESMO TEMPO. ANALISANDO O QUE ERA PRECISO PARA FAZER O DOMINÓ E TENTANDO GENERALIZAR O MÁXIMO POSSÍVEL.

CONTAR A HISTÓRIA DO PROTÓTIPO WEB QUE FOI FORTEMENTE BASEADO NA IMPLEMENTAÇÃO DO PROTÓTIPO EM PROLOG E NO PRÓPRIO YII. MOSTRANDO QUE OS CONCEITOS DO MX SÃO IMPLEMENTADOS NO PROLOG DE UMA FORMA "AUTOMÁTICA" (TEM OUTRA PALAVRA PRA ISSO) E QUE A IMPLEMENTAÇÃO WEB SEGUE A MESMA IDÉIA.

MOSTRAR COMO SERIA A IMPLEMENTAÇÃO DOS FEATURES QUE FALTAM PARA A WEB: PERMISSÕES EM NÍVEL DE AÇÕES BÁSICAS SOBRE O DOCUMENTO.

ASSIM, DESEJA-SE DEMONSTRAR QUE AS IMPLEMENTAÇÕES SÃO POSSÍVEIS DE SE TORNAR FACTÍVEIS (COM AS TÉCNICAS APRESENTADAS) UMA VEZ ESPECIFICADAS CORRETAMENTE COM O MX.

8.3 Eficiência das implementações

PESSOALMENTE, EU PREFIRO NÃO ENTRAR NESSE MÉRITO.

É POSSÍVEL FAZER UMA ANÁLISE DE TEMPO DE EXECUÇÃO DAS CONSULTAS, COMO ANÁLISE DO ALGORITMO E TESTES DE CARGA.

8.4 Agentes

AINDA TEM O PROBLEMA DOS AGENTES, QUE NÃO FOI MUITO BEM TRABALHADO EM NÍVEL CONCEITUAL E EM NÍVEL IMPLEMENTACIONAL FORAM FEITOS ALGUNS TESTES COM O DOMINÓ.

SE OS AGENTES SE COMPORTAREM COMO AGENTES HUMANOS, ELES PODEM ACESSAR PÁGINAS E FAZER PESQUISAS E ATIVIDADES COMO OS HUMANOS.

MAS O MAIS INTERESSANTE SERIA UMA API PARA ACESSO DE AGENTES. E TAMBÉM PARA QUE OS TEMPLATES TAMBÉM ACESSEM DADOS SERVIDOS POR AGENTES.

DISCUTIR TAMBÉM A ATUAÇÃO DE AGENTES DE PERCEPÇÃO, QUE PODEM AGIR COM POPUPS.

8.5 Fora do escopo do MX

QUAIS AS CARACTERÍSTICAS DOS SISTEMAS QUE O MX NÃO SE PROPÕEM A MODELAR?

8.6 Conclusão do Capítulo

O MX É ADEQUADO PARA MODELAGEM DE DIVERSOS AMBIENTES VIRTUAIS.

A MODELAGEM DE UM SISTEMA PRESUPÕE QUE É POSSÍVEL IMPLEMENTÁ-LO.

O QUE ELE NÃO CONSEGUE IMPLEMENTAR.

9 Conclusão

FALTA ESCREVER AQUI.

10 Referências

AHMADI, N. et al. Towards the web of applications: incorporating end user programming into the web 2.0 communities. **Proceedings of the 2nd international workshop on Social software engineering and applications (SoSEA '09)**, Amsterdam, The Netherlands, 2009. 9-14.

ALVES, E. et al. Escrever e reescrever pela WEB: práticas de escrita utilizando o objeto de aprendizagem CARTOLA. **Anais do Simpósio Brasileiro de Informática na Educação (SBIE)**, 2007.

ANDERSON, C. **The Long Tail:** How endless choice is creating unlimited demand. London, UK: Random House Business Books, 2006. ISBN 1-4013-0237-8.

ANDERSON, P. What is Web 2.0? Ideas, technologies and implications for education. **JISC Technology and Standards Watch**, Bristol, 2007.

BEDER, D. M. et al. A Case Study of the Development of e-Learning Systems Following a Component-based Layered Architecture. **Seventh IEEE International Conference on Advanced Learning Technologies (ICALT 2007)**, Niigata, 2007. 21 - 25.

BERNERS-LEE, T. **Weaving the Web:** the original design of the World Wide Web by its inventor. 1st. ed. New York: HarperCollins, 2000. ISBN 0-06-251587-X.

BERNERS-LEE, T.; CAILLIAU, R. WorldWideWeb: Proposal for a HyperText Project, 1990. Disponivel em: <http://www.w3.org/Proposal.html>. Acesso em: 8 Março 2011.

BERNERS-LEE, T.; CONNOLLY, D. Hypertext Markup Language (HTML): A Representation of Textual Information and MetaInformation for Retrieval and

Interchange, 1993. Disponivel em: <http://www.w3.org/MarkUp/draft-ietf-iiir-html-01.txt>. Acesso em: 8 Março 2011.

BERNERS-LEE, T.; FIELDING, R.; FRYSTYK, H. Hypertext Transfer Protocol -- HTTP/1.0, 1996. Disponivel em: <http://www.w3.org/Protocols/rfc1945/rfc1945>. Acesso em: 8 Março 2011.

BERNERS-LEE, T.; FIELDING, R.; MASINTER, L. Uniform Resource Identifier (URI): Generic Syntax, 2005. Disponivel em: <http://labs.apache.org/webarch/uri/rfc/rfc3986.html>. Acesso em: 8 Março 2011.

BERNERS-LEE, T.; MASINTER, L.; MCCAHILL, M. Uniform Resource Locators (URL), 1994. Disponivel em: <http://www.w3.org/Addressing/rfc1738.txt>. Acesso em: 8 Março 2011.

BURBECK, S. Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC), 1987. Disponivel em: <http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html>. Acesso em: 18 Julho 2011.

BUSCHMANN, F. et al. **Pattern-oriented Software Architecture:** A System of Patterns. Chichester, England: John Wiley & Sons, 1996. ISBN 0471958697.

CARVALHO, M. J. S.; NEVADO, R. A. D.; MENEZES, C. S. D. Arquiteturas Pedagógicas para Educação a Distância: Concepções e Suporte Telemático. **Anais do XVI Anais do Simpósio Brasileiro de Informática na Educação (SBIE 2005)**, UFJF, 2005.

CASTRO, T. et al. Fleshing Out Clues on Group Programming Learning. **ICEIS 2009 - Proceedings of the 11th International Conference on Enterprise Information Systems**, Milan, Italy, May 2009. 68-73.

CHIEU, V. M. COFALE: An Authoring System for Supporting Cognitive Flexibility. **Proceedings of the Sixth International Conference on Advanced Learning Technologies (ICALT'06)**, 2006.

CHRISTENSEN, E. et al. Web Services Description Language (WSDL) 1.1, 2001. Disponivel em: <http://www.w3.org/TR/wsdl>. Acesso em: 9 Março 2011.

CLUBB, O. L. Human-to-Computer-to-Human Interactions (HCHI) of the communications revolution. **Interactions**, 14, n. 2, 2007. 35-39.

DICT.ORG. Morpheus. **The DICT Development Group**. Disponivel em: <http://www.dict.org/>. Acesso em: 27 Julho 2011.

DOUGIAMAS, M.; TAYLOR, P. C. Moodle: Using learning communities to create an open source course management system. **Proceedings of world conference on educational multimedia, hypermedia and telecommunications**, 2003.

ELLIS, C.; GIBBS, S. J.; REIN, G. Groupware: some issues and experiences. **Magazine Communications of the ACM**, 34, January 1991. 39-58.

ELLIS, C.; WAINER, J. A conceptual model of groupware. **Proceedings of the 1994 ACM conference on Computer supported cooperative work**, 1994. 79-88.

ELLIS, C.; WAINER, J. Groupware and Computer Supported Cooperative Work. In: WEISS, G. **Multiagent systems:** a modern approach to distributed artificial intelligence. [S.l.]: MIT Press, 2000. Cap. 10, p. 425.

FAGUNDES, L. C. et al. Projetos de Aprendizagem-Uma Experiência Mediada por Ambientes Telemáticos. **Anais do XXV Congresso da Sociedade Brasileira de Computação (CSBC), XI Workshop de Informática na Escola (WIE)**, São Leopoldo/RS, 2005. 2771-2779.

FUKS, H. Aprendizagem e Trabalho Cooperativo no Ambiente AulaNet. **Revista Brasileira de Informática na Educação (RBIE)**, n. 6, 2000. 53-73.

FUKS, H. et al. The 3C Collaboration Model. In: KOCK, N. **Encyclopedia of E-Collaboration**. Hershey, New York: Information Science Reference, 2007. p. 637-644. ISBN 978-1-59904-000-4.

GADELHA, B. et al. An Approach for Developing Groupware Product Lines (GPL) based on the 3C. **CRIWG 2009, LNCS 5784**, Portugal, 2009. 328-343.

GEROSA, M. A. et al. Development of Groupware Based on the 3C Collaboration Model and Component Technology. **CRIWG 2006, LNCS 4154**, 2006. 302 – 309.

GETTYS, J. Hypertext Transport Protocol HTTP/1.1, 1996. Disponivel em: <http://www.w3.org/Talks/9608HTTP/>. Acesso em: 8 Março 2011.

GUTWIN, C.; GREENBERG, S. The mechanics of collaboration: Developing low cost usability evaluation methods for shared workspaces. **Proeedings. IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2000.(WET ICE 2000)**, 2000. 98-103.

ITMAZI, J. **Sistema Flexible de Gestión Del eLearning para Soportar El Aprendizaje en las Universidades Tradicionales y Abiertas**. Universidad de Granada. [S.l.]. 2005.

JACKSON, M. A. **Principles of program design**. London: Academic Press, 1975. ISBN 0123790506.

KEPSER, S. A Simple Proof for the Turing-Completeness of XSLT and XQuery. **Proceedings of Extreme Markup Languages**, Montréal, Québec, 2004. Disponível em http://conferences.idealliance.org/extreme/html/2004/Kepser01/EML2004Kepser01.html.

KESTEREN, A. V.; JACKSON, D. The XML HTTP Request Object. **The World Wide Web Consortium (W3C)**, 3 August 2010. Disponivel em: <http://www.w3.org/TR/XMLHttpRequest/>. Acesso em: 8 Março 2011.

LIE, H. W.; BOS, B. Cascading Style Sheets, level 1, 2008. Disponivel em: <http://www.w3.org/TR/2008/REC-CSS1-20080411/>. Acesso em: 8 Março 2011.

LIMA, P. S. R. **Um Ambiente Colaborativo de Aprendizagem Interdisciplinar Apoiado por Interfaces Adaptativas.** [S.l.]: Tese de Doutorado em Engenharia Elétrica, 2006.

MACLEAN, A. et al. User-tailorable systems: pressing the issues with buttons. **Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '90)**, Seattle, Washington, United States, 1990. 175-182.

MAGALHÃES, E. et al. Impacto da usabilidade na educação a distância: um estudo de caso no Moodle IFAM. **Proceedings of the IX Symposium on Human Factors in Computing Systems**, 2010. 231-236.

MEDEIROS, V. C. L. **Um Ambiente de Autoria para Estações de Aprendizagem**. Universidade Federal do Amazonas. [S.l.]. 2005.

MENDONÇA, A. P. et al. Um Ambiente Telemático para. **Anais do Simpósio Brasileiro de Informática na Educação (SBIE)**, 2003.

MENEZES, C. S. et al. MOrFEU: Multi-Organizador Flexível de Espaços VirtUais para Apoiar a Inovação Pedagógica em EAD. **Anais do XIX Simpósio Brasileiro de Informática na Educação (SBIE 2008)**, Fortaleza - CE, 2008. 451-460.

MENEZES, C. S.; CURY, D.; CASTRO, A. N. An Architecture of an Environment for Cooperative Learning (AmCorA). **Proceedings of ICECE 2000 - International Conference on Engineering and Computer Education**, São Paulo, 2000.

MINISTÉRIO DA EDUCAÇÃO. e-Proinfo: Ambiente Colaborativo de Aprendizagem., 2000. Disponivel em: <http://www.eproinfo.mec.gov.br/>.

MINISTÉRIO DA EDUCAÇÃO. **UCA | Um Computador por Aluno**, 2011. Disponivel em: <http://www.uca.gov.br/>. Acesso em: 12 Julho 2011.

MONGODB. **MongoDB**, 2011. Disponivel em: <http://www.mongodb.org/>. Acesso em: 25 Agosto 2011.

MONTEIRO, V. et al. Ferramenta de Autoria e Interação para apoio ao desenvolvimento de Projetos de Aprendizagem. **Renote Revista Novas Tecnologias na Educação**, 3, n. 2, 2005.

MOODLE. **Moodle.org**, 2011. Disponivel em: <http://moodle.org/>. Acesso em: 25 Agosto 2011.

MOODLE.ORG. Moodle Statistics. **Moodle.org**, 2011. Disponivel em: <http://moodle.org/stats/>. Acesso em: 2 Março 2011.

MYERS, B. A.; KO, A. J.; BURNETT, M. M. Invited research overview: end-user programming. **CHI '06 extended abstracts on Human factors in computing systems**, 2006. 75-80.

NEVADO, R. A. D.; DALPIAZ, M. M.; MENEZES, C. S. D. Arquitetura Pedagógica para Construção Colaborativa de Conceituações. **Workshop Sobre Educação na Escola (WIE)**, Bento Gonçalves - RS, 2009.

NEVADO, R. A.; CARVALHO, M. J. S.; MENEZES, C. S. (Eds.). **Aprendizagem em rede na educação a distância:** estudos e recursos para formação de professores. Porto Alegre: Ricardo Lenz, 2007. ISBN 978-85-87787-42-2.

O'REILLY, T. What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. **Communications & Strategies**, 2007.

PAREDES, H. **Uma Arquitectura de Software Dinâmica para a Criação de Ambientes de Interacção Social Regulada na Web**. Universidade do Minho. Braga, Portugal. 2007.

PAREDES, H.; MARTINS, M. F. Social interaction regulation in virtual web environments using the Social. **Journal of Network and Computer Applications**, Janeiro 2010.

PENICHET, V. M. R. et al. A Classification Method for CSCW Systems. **Journal Electronic Notes in Theoretical Computer Science (ENTCS)**, 168, February 2007. 237-247.

PESSOA, J. M.; NETTO, H. V.; MENEZES, C. S. D. FAmCorA: um framework para a construção de ambientes cooperativos inteligentes de apoio a aprendizagem na Internet baseado em web services e agentes. **Anais do XIII Simpósio Brasileiro de Informática na Educação (SBIE 2002)**, UNISINOS, 2002. 94-104.

PREECE, J. **Online communities:** designing usability and supporting sociability. [S.l.]: John Wiley & Sons, Inc, 2000. ISBN 0471805998.

RAGGETT, D. Client-side Scripting and HTML, 1997. Disponivel em: <http://www.w3.org/TR/WD-script-970314>. Acesso em: 8 Março 2011.

RAGGETT, D.; HORS, A. L.; JACOBS, I. HTML 4.01 Specification, 1999. Disponivel em: <http://www.w3.org/TR/html4/>. Acesso em: 8 Março 2011.

RAMA, J.; BISHOP, J. A survey and comparison of CSCW groupware applications. **Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries (SAICSIT '06)**, 2006. 198-205.

RAMAN, T. V. Toward 2W, beyond web 2.0. **Communications of ACM**, February 2009. 52-59.

REAL, L. M. C.; MENEZES, C. S. Júri simulado: possibilidade de construção de conhecimento a partir de interações em grupo. In: NEVADO, R. A.; CARVALHO, M. J. S.; MENEZES, C. S. **Aprendizagem em rede na educação a distância**. Porto Alegre: Ricardo Lenz, 2007.

ROBERTSON, D. et al. **Open Knowledge: Semantic webs through peer-to-peer interaction**. University of Trento. [S.l.]. 2006.

ROBINSON, D.; COAR, K. The Common Gateway Interface (CGI) Version 1.1, 2004. Disponivel em: <http://tools.ietf.org/html/rfc3875>. Acesso em: 8 Março 2011.

ROCHA, H. V. TelEduc: Software livre para educação a distância. In: SILVA, M. **Educação On-line.** 1a Edição. ed. São Paulo: Edições Loyola, 2003. p. 377-396.

SANTOS, L. N.; CASTRO, A. N.; MENEZES, C. S. MOrFEu: Criando Ambientes Virtuais Flexíveis na Web para Mediar a Colaboração. **Anais do Congreso Iberoamericano de Informática Educativa (IE2010)**, Santiago, Chile, 2010. 114-121.

SANTOS, L. N.; CASTRO, A.; MENEZES, C. MOrFEu: Criando Ambientes Virtuais Flexíveis na Web para Mediar a Colaboração. **Anais do Congreso Iberoamericano de Informática Educativa (IE2010)**, Santiago, Chile, 2010. 114-121.

SANTOS, L. N.; CASTRO, A.; MENEZES, C. Flexible Virtual Environments for Teaching and Learning. **42nd ASEE/IEEE Frontiers in Education Conference Proceedings (FIE)**, Seattle, WA, USA, October 2012. 1388-1393.

SCHMIDT, K.; BANNON, L. Taking CSCW Seriously: Supporting Articulation Work. **Computer Supported Cooperative Work (CSCW): An International Journal**, 1, 1992. 7-40.

SERRES, F. F.; BASSO, M. V. D. A. Diários virtuais – Uma ferramenta de comunicação social para a autoria e aprendizagem de Matemática. **Anais do XX Simpósio Brasileiro de Informática na Educação (SBIE 2009)**, 2009.

SPÓSITO, M. A. F.; CASTRO, T. H. C. D.; CASTRO, A. N. D. Estação de Percepção: Uma Abordagem para o Monitoramento em Ambientes Virtuais de Aprendizagem. **Anais do XIX Simpósio Brasileiro de Informática na Educação (SBIE 2008)**. 288-298.

TAIVALSAARI, A. et al. Web browser as an application platform: the lively Kernel experience. **SMLI TR-2008-175**, Mountain View, CA, USA, 2008.

ULLRICH, C. et al. Why web 2.0 is good for learning and for research: principles and prototypes. **Proceeding of the 17th international conference on World Wide Web (WWW '08)**, Beijing, China, 2008. 705-714.

VASSILEVA, J. et al. Lessons from Deploying I-Help. **Proocedings Workshop Multi-Agent Architectures for Distributed Learning Environments, International Conference on AI and Education**, 2001. 3-11.

WATSON, W. R.; WATSON, S. L. An Argument for Clarity: What are Learning Management Systems, What are They Not, and What Should They Become? **Journal TechTrends**, 51, 2007. 28-34.

WAYBACK MACHINE. DominoNet Regras. **Internet Archive**, 2003. Disponivel em: <http://web.archive.org/web/20030618054744/http://www.dominonet.com.br/regras.asp>. Acesso em: 6 Dezembro 2012.

WIKIMEDIA FOUNDATION. **Wikipedia**, 2011. Disponivel em: <http://www.wikipedia.org/>. Acesso em: 3 Março 2011.

WON, M.; STIEMERLING, O.; WULF, V. Component-Based Approaches to Tailorable Systems. **End User Development**, 9, 2006. 115-141.

YII. **Yii Framework**, 2011. Disponivel em: <http://www.yiiframework.com/about/>. Acesso em: 16 Agosto 2011.

11 Apêndice A: Modelagem de ferramentas com o MX

Neste apêndice são listadas as modelagens de algumas ferramentas de comunicação/interação e ambientes virtuais, seguindo o arcabouço MX.

O propósito desses exemplos é mostrar na prática como esses conceitos são aplicados e tentar abranger os mais diversos ambientes possíveis.

11.1 Blog

Um blog é uma aplicação comum na Internet de hoje. Trata-se de um log (registro de fatos) virtual. Mais do que isso, um blog tem sido usado para expressar ideias pessoais, opiniões, séries de humor, notícias de algum segmento, promoções, revisões de filmes ou produtos, perguntas e respostas, entre outras tantos modos de usar um blog nos dias de hoje.

Basicamente, um blog é um sequencia de textos com tamanho médio escritos pelos autores do blog, organizados do mais recente para o mais antigo. Cada um desses textos, chamados de posts, pode receber comentários de seus leitores. A maneira mais comum de comentários em blogs é uma lista de comentários organizados pela data de submissão do comentário, os mais antigos primeiro, depois os mais recentes no final.

Em blogs grandes, como muitos posts, é necessária uma funcionalidade a mais, chamada de tagueamento, a inclusão de tags (etiqueta). Um post pode receber várias tags, que são pedaços pequenos de texto, em geral de uma a três palavras, que definem qual o assunto daquele post. Por exemplo, em um blog sobre futebol, o autor

do post pode estar falando sobre um jogo entre a seleção brasileira e a seleção italiana, assim algumas possíveis tags seriam: “Brasil”, “Itália”. Essas tags funcionam como organizadores de posts semelhantes e também auxiliam na busca por posts naquele blog.

Assim, têm-se algumas UPIs identificadas em um VCom Blog. As relações entre essas UPIs são: comentário são referentes a posts; posts possuem tags. Assim, pode-se modelar o documento desse ambiente virtual. Para uma melhor representação, apresenta-se, na Figura 42, um modelo de um VCom Blog utilizando-se caixas para representar as UPIs e setas para representar as relações entre as UPIs.

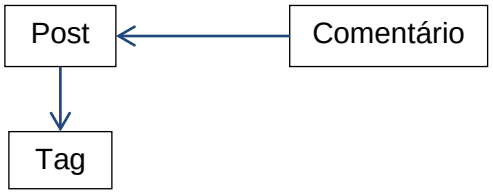


Figura 42. Diagrama do modelo de documento de um blog.

Algumas atividades desse VCom são:

- Incluir Post: publica uma UPI com o rótulo “Post”;
- Atribuir Tag: publica uma UPI como rótulo “Tag” e publica a relação entre um Post e essa Tag;
- Incluir Comentário: publica uma UPI com o rótulo “Comentário” e publica a relação desse Comentário com um Post.

A modelagem da Figura 42 não impõe nenhuma restrição explicitamente, ela serve de guia para a especificação desse VCom. As atividades são a forma como o Documento é alterado, ou seja, somente haverá alterações desse tipo. Porém, as atividades não dizem quem pode ou não pode fazer essas operações, isso fica a cargo do Protocolo de Interação.

Sobre o Protocolo, esse VCom possui apenas um estado, não possui grupos e tem dois papéis: escritor e leitor. Segue uma lista das permissões para esse VCom:

- O dono do VCom pode designar atores do papel escritor;
- Os escritores podem designar outros escritores;

- Qualquer pessoa pode ser leitor;
- Leitores podem consultar o Documento inteiro;
- Escritores podem publicar Posts;
- Escritores podem publicar Tags;
- Escritores podem publicar relações entre Posts e Tags;
- Leitores podem publicar Comentários;
- Leitores podem publicar relações entre Comentários e Posts.

Com relação à Interface, esse VCom possui três páginas: a página inicial, a página de comentários de um post e a página de filtragem de posts por tags. Essas páginas só executam os templates “Posts”, “Comentários” e “Posts de uma Tag”, respectivamente, para qualquer papel.

Na Figura 43, é apresentado um diagrama de navegação que representa essas associações entre páginas e templates, para o estado único do VCom. As caixas arredondadas são as páginas, e as caixas com canto reto representam os templates. Os bonequinhos representam a quais papéis o template está associado.

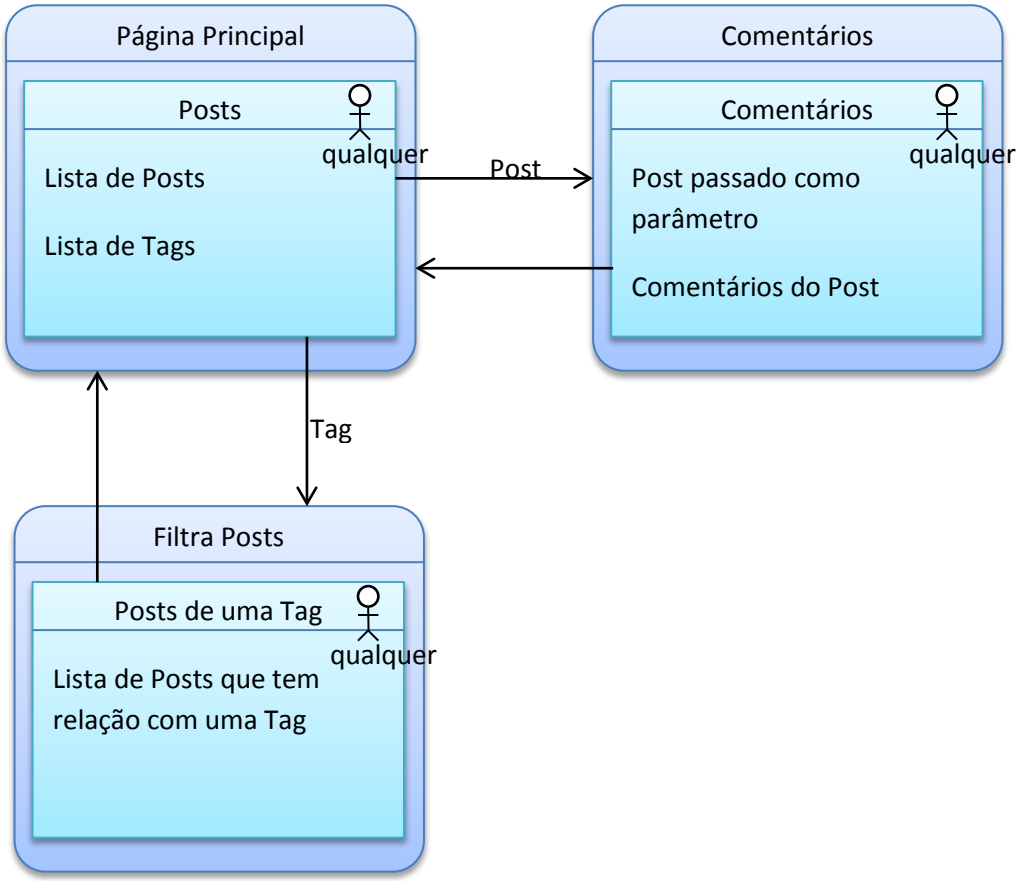


Figura 43. Diagrama de navegação da interface de um blog.

11.2 Fórum

Um fórum é uma ferramenta de discussão de um certo tema. Entre os diversos tipos de fórum existentes, estão aqueles mais simples que apenas apresentam uma mensagem após a outra, organizadas por data de publicação, como os comentários de um post em um fórum. De certa forma, as mensagens de e-mail agregadas do Gmail[21] têm essa mesma organização. Porém, nesse exemplo, serão tratados alguns fóruns mais interessantes, aqueles organizados hierarquicamente.

Os fóruns organizados hierarquicamente são aqueles em que um post de resposta é atribuído explicitamente a outro post, formando assim uma árvore de respostas. Esses fóruns têm o intuito de estabelecer um debate mais organizado.

[21] http://mail.google.com

As UPIs desse VCom são posts, pedaços de texto de tamanho pequeno a médio. Um post pode estar iniciando um novo ramo na discussão (não é feito em resposta a outro post), ou ser feito em reposta a outro post já publicado no VCom. Na Figura 44, está o modelo do documento desse tipo de VCom.

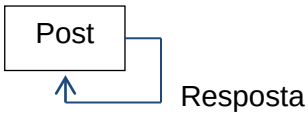


Figura 44. Diagrama do modelo de documento de um fórum.

Algumas atividades desse VCom são:

- Incluir Post-Raiz: publica uma UPI com o rótulo "Post";
- Incluir Post-Resposta: publica uma UPI como o rótulo "Post" e publica a relação de resposta entre esse post e outro post.

Sobre o Protocolo, esse VCom possui apenas um estado, não possui grupos e tem apenas um papel, participante. Segue uma lista das permissões para esse VCom:

- O dono do VCom pode designar atores do papel participante;
- Os participantes podem consultar o Documento inteiro;
- Participantes podem publicar Posts;
- Participantes podem publicar relações de Resposta entre Posts;

Com relação à Interface, esse VCom possui apenas uma página, a página principal, que executa, para qualquer papel, o único template, chamado de "Posts".

Na Figura 45, é apresentado um diagrama de navegação que representa essas associações entre páginas e templates, para o estado único do VCom. As caixas arredondadas são as páginas, e as caixas com canto representam os templates. Os bonequinhos representam a quais papéis o template está associado.

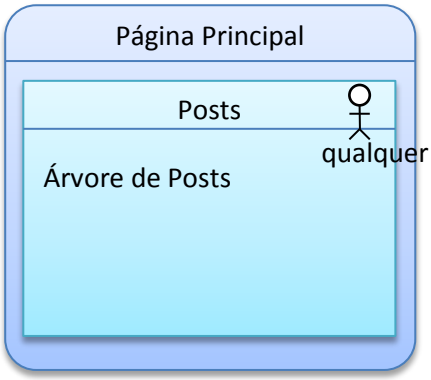


Figura 45. Diagrama de navegação da interface de um fórum.

11.3 Enquete

Uma enquete é uma forma de saber a opinião das pessoas sobre determinado assunto. Por exemplo, faz-se uma pergunta de caráter pessoa para saber a opinião dos usuários da enquete. As possíveis respostas são fixas, de forma que os usuários têm que escolher sua resposta apenas entre as possíveis respostas apresentadas.

Dessa forma, as UPIs desse Documento são: a Questão, que se estar perguntando; e a Opções de resposta. Além disso, há também uma relação entre as UPIs de opção de resposta e as pessoas que respondem à enquete. Na Figura 46, é apresentado um diagrama que representa o modelo do Documento de uma enquete, com as relações entre UPIs com uma seta cheia e azul-escuro e as relações entre o usuário que responde à enquete com a opção de resposta com uma seta pontilhada e verde.

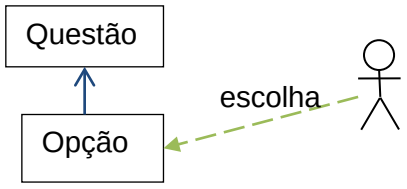


Figura 46. Diagrama do modelo de documento de uma enquete.

Algumas atividades desse VCom são:

- Incluir Questão: publica uma UPI com o rótulo “Questão”;

- Incluir Opção: publica uma UPI com o rótulo “Opção” e publica uma relação entre essa opção e a questão;
- Incluir Escolha: publica uma relação com o rótulo “escolha” entre o usuário que fez a escolha com a opção escolhida.

Nota-se que nesse Documento não há como representar restrições do tipo: “deve haver somente uma questão por enquete”; “um participante não pode responder duas vezes à mesma enquete”; “os resultados são exibidos somente no final da enquete”. Essas questões são resolvidas pelo Protocolo de Interação.

Além das permissões, o Protocolo define os estados e as transições de estado do VCom. Uma enquete possui três estados, que podem ser visualizados na Figura 47:

Figura 47. Diagrama de estados de uma enquete.

Para cada estado, há um conjunto de permissões e esquemas de navegação definidos.

Estado Inicial. Nesse momento o dono da enquete cadastra a pergunta e as opções de resposta da enquete. Dessa forma, as permissões nesse estado são:

- O dono pode consultar o Documento todo;
- O dono do VCom pode publicar UPI com rótulo “Questão”, se não há nenhuma UPI desse tipo já publicada;
- O dono pode publicar UPI com o rótulo “Opção”, se houver uma UPI Questão já publicada;
- O dono pode associar uma UPI Opção com a UPI Questão;
- O dono pode mudar de estado para o estado “Responder”, se há uma UPI Questão publicada e há uma ou mais UPIs Opção publicadas.

Há somente uma página nesse estágio, a página inicial, e essa página mostra a questão e as questões que estão sendo cadastradas. Essa associação da página inicial com o template de cadastro de questão pode ser visto na Figura 48.

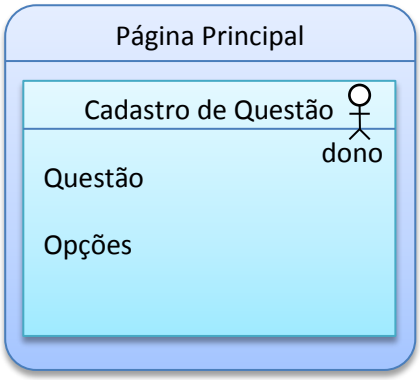


Figura 48. Diagrama de navegação da interface de uma enquete em seu estado inicial.

Estado Responder. Nesse estado, os usuários podem dar suas respostas à pergunta da enquete, mas isso só pode ser feito uma vez por usuário. De modo que depois que o usuário já votou, a ele só exibida a mensagem "Obrigado por votar nesta enquete", e fica esperando o fim da enquete. Enquanto isso, o dono do VCom pode ver o andamento das votações e verificar o resultado parcial. Dessa forma, as permissões nesse estado são:

- O dono pode consultar o Documento todo;
- Qualquer pessoa pode consultar a Questão, as Opções e as relações entre UPIs (mas não as relações entre UPIs e usuários);
- Qualquer pessoa pode publicar uma relação entre ele o uma UPI Opção, se não houver outra relação dessa já publicada, com qualquer outra UPI Opção;
- O dono pode mudar de estado, para o estado "Fechada".

O diagrama de navegação apresentando somente a página inicial com diferente templates condicionados pelo papel do usuário é apresentado na Figura 49.

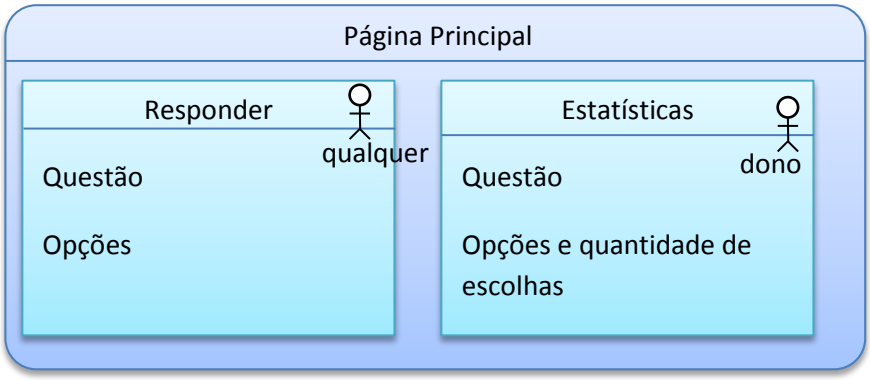


Figura 49. Diagrama de navegação da interface de uma enquete em seu estado de responder.

Estado Fechada. Nesse estado todos podem ver os resultados da enquete. O template de estatísticas é então exibido para todos (conforme Figura 50). A única permissão nesse estado é de que todos podem consultar o Documento inteiro.

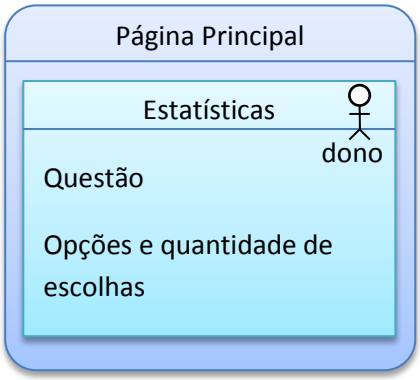


Figura 50. Diagrama de navegação da interface de uma enquete quando ela se encontra fechada.

Nesse último estado ocorre uma situação interessante: é possível obter quantas pessoas escolheram uma dada opção, mas não é possível pesquisar quais foram essas pessoas. Isso se deve ao fato de somente ser possível fazer as pesquisas listadas.

11.4 Wiki

Um wiki é uma ferramenta de autoria de páginas em formato de hipertexto. Nesse ambiente uma página é criada se foi feita uma referência a ela em outra página. Formando assim, uma teia de páginas que são ligadas entre si através de hiperlinks.

Assim, as UPIs do Documento de um wiki são as páginas, que podem possuir mais de um autor. Os hiperlinks entre as páginas são relações entre as UPIs de páginas. Assim, o modelo para esse Documento pode ser representado como na Figura 51 abaixo.

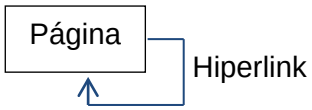


Figura 51. Diagrama de um modelo de documento de um wiki.

Algumas atividades desse VCom são:

- Incluir uma nova Página: publica uma UPI com rótulo "Página", publica uma relação entre essa página e a página que tem um hiperlink para ela, cria relações dos hiperlinks da página criada;
- Editar uma página: edita o conteúdo da UPI, e atualiza (insere e/ou remove) as relações de hiperlinks entre as páginas.

11.5 Controvérsia Acadêmica

Um fórum modificado para apoio à prática pedagógica **Controvérsia Acadêmica** realizada em ambientes virtuais (MENDONÇA, CASTRO, *et al.*, 2003), onde o segundo nível do fórum é composto pelas postagens "Concordo", "Discordo" e "Depende", cada uma referente a uma das três etapas da atividade. A atividade é composta por dois grupos de pessoas, onde na primeira etapa um grupo deve discutir a favor do assunto a ser debatido e na segunda etapa contra, e vice-versa para o outro grupo. Na última etapa, todos devem discutir no tópico "Depende".

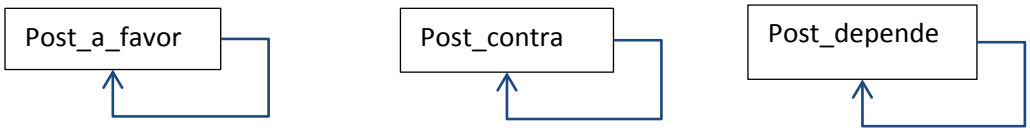


Figura 52. Diagrama do modelo de documento de um fórum da Controvérsia Acadêmica.

Algumas atividades desse VCom são:

- Incluir Post-Raiz a Favor: publica uma UPI com o rótulo "Post_a_favor";
- Incluir Post-Resposta a Favor: publica uma UPI como o rótulo "Post_a_favor" e publica a relação de resposta entre esse post e outro post;
- Incluir Post-Raiz Contra: publica uma UPI com o rótulo "Post_contra";
- Incluir Post-Resposta Contra: publica uma UPI como o rótulo "Post_contra" e publica a relação de resposta entre esse post e outro post;
- Incluir Post-Raiz Depende: publica uma UPI com o rótulo "Post_depende";
- Incluir Post-Resposta Depende: publica uma UPI como o rótulo "Post_depende" e publica a relação de resposta entre esse post e outro post;

As questões relacionadas a quais pessoas podem postar em qual fórum são resolvidas e definidas no protocolo de interação.

11.6 Facebook

O Facebook[22] é uma das redes sociais mais populares no dia de hoje, que funciona como um ambiente virtual para relacionamentos pessoais.

Uma pessoa nessa rede social pode postar textos, fotos e vídeos em seu mural. As pessoas que são amigas dessa pessoa veem essa publicação em sua linha do tempo (*timeline*), que reúne as postagens de vários de seus amigos e de pessoas que ele é seguidor.

A qualquer uma dessas postagens, os usuários leitores podem publicar comentários. Além disso, existe o botão "Curtir", que permite a um usuário sinalizar explicitamente que gostou de alguma postagem ou comentário.

Outro recurso bastante popular é o compartilhamento de postagens. É possível compartilhar as postagens de outros usuários em sua própria linha do tempo, fazendo

[22] http://www.facebook.com

uma descrição da postagem original. Os comentários dessa "nova" postagem ficam no escopo do usuário que compartilhou, e não da postagem original.

É possível fazer menção a um usuário em uma postagem, dessa forma, será acionada uma notificação para aquele usuário, informando que alguém o mencionou em uma postagem ou comentário.

Uma funcionalidade semelhante é o post direcionado, no qual um usuário publica um post explicitando que ele é direcionado a outro usuário, mas ainda assim ele é visível a todos os seus amigos.

Há ainda outro tipo de comunicação direta entre dois usuários, a mensagem privada, que funcionam como pequenos chats entre dois usuários, onde apenas os usuários envolvidos leem essas mensagens.

O Facebook ainda permite que os usuários se organizem em grupos de pessoas, disponibilizando um espaço compartilhado para esse grupo, onde os seus participantes podem publicar postagens nesse grupo.

A Facebook ainda possuem outras funcionalidades que não foram descritas aqui. Na Figura 53, é apresentado um diagrama do modelo de Documento de uma ferramenta como o Facebook, com as funcionalidades descritas acima.

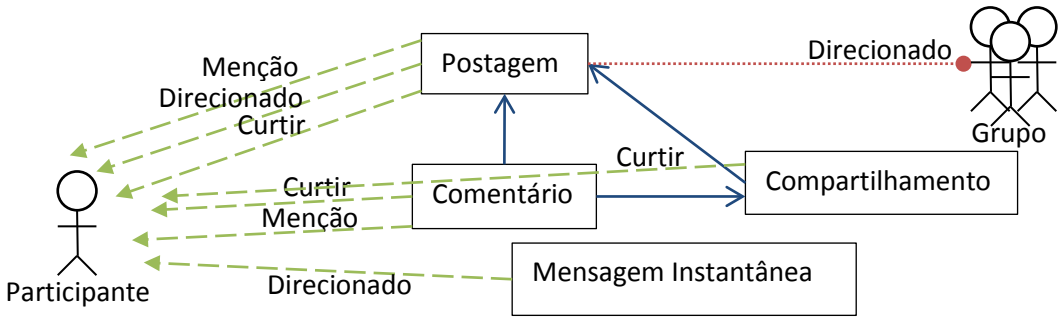


Figura 53. Diagrama do modelo de documento do Facebook simplificado.

Interessante notar que a relação de autoria está implícita nesse modelo. Por exemplo, não é preciso representar que um participante é autor (e publicador) de uma postagem. Toda publicação tem uma relação como usuário que a publicou. As outras relações existentes entre uma UPI e usuários devem ser modeladas. Assim,

para modelar uma mensagem instantânea entre duas pessoas, basta representar que a mensagem foi direcionada a outro usuário, com a origem dessa mensagem no autor da publicação da UPI.

Questões como quem são os amigos de um usuário devem ser tratadas no Protocolo de Interação.

Algumas atividades desse VCom são:

- Nova Postagem: publica uma UPI com o rótulo Postagem e processa as possíveis menções, publicando relações com o rótulo "menção" para os usuários mencionados;
- Nova Postagem Direcionada: publica uma UPI com o rótulo Postagem, publica uma relação entre essa UPI e o usuário mencionado e processa as possíveis menções, publicando relações com o rótulo "menção" para os usuários mencionados;
- Compartilhar uma Postagem: publica uma UPI com o rótulo "Compartilhamento", publica uma relação entre essa UPI e a UPI compartilhada e processa as possíveis menções, publicando relações com o rótulo "menção" para os usuários mencionados;
- Comentar uma Postagem/Compartilhamento: publica uma UPI com o rótulo "Comentário", publica uma relação com a postagem/compartilhamento comentada e processa as possíveis menções, publicando relações com o rótulo "menção" para os usuários mencionados;
- Curtir: publica uma relação com rótulo "Curtir" entre a UPI de Postagem/Comentário/Compartilhamento.
- Mandar Mensagem: publica uma UPI com o rótulo "Mensagem Instantânea" e publica uma relação entre essa UPI e o usuário de destino.

11.7 Jogo de Dominó

Dominó é um jogo que utiliza pequenas pedras (*bones*) divididas ao meio, onde cada lado (ponta) possui um número entre zero e seis, totalizando 28 pedras. No estado do Amazonas, ele é um jogo tradicional que é jogado com quatro pessoas, duas duplas, sentadas ao redor de uma mesa. As pedras são encaixadas uma nas outras de

maneira que tenham pontas iguais. O objetivo do jogo é fazer mais pontos que a outra dupla. Os pontos são feitos quando a soma das pontas que estão sem encaixar é múltiplo de cinco.

Há alguns anos havia um site que possuía esse jogo para ser jogado on-line com outras pessoas, o DominóNet[23]. Mas infelizmente ele foi descontinuado no ano de 2004[24] e atualmente a sua página inicial apresenta somente a frase "EM CONSTRUÇÃO". E esse tipo de ambiente virtual que estamos tratando nessa seção, um ambiente virtual para jogar dominó on-line. Um conjunto de regras pode ainda ser acessado através de um serviço do site Archive.org (WAYBACK MACHINE, 2003).

No jogo tradicional, os jogadores sentam-se ao redor de uma mesa, intercalando-se os membros das duplas. No início da rodada, cada jogador recebe aleatoriamente sete pedras. As pedras com as duas pontas iguais são chamadas carroças, e começa o jogo quem possuir a carroça de seis (também chamada de sena), ou seja, a pedra cujas pontas são (6,6). O jogo segue em sentido horário, sendo o próximo que joga é aquele sentado ao lado esquerdo do jogador que acabou de jogar.

Uma pedra só pode ser jogada se há uma ponta de alguma das pedras que já foram jogadas na mesa sem um encaixe (chamada de ponta de mesa) e que essa ponta sem encaixe possa ser encaixada em uma das pontas da pedra que vai ser jogada.

Na Figura 54, apresenta-se uma foto de uma mesa de dominó, com as pedras encaixadas corretamente. As carroças são encaixadas na transversal, e valem o dobro de pontos quando estão em uma ponta de mesa. Na foto, as pontas de mesa são o 6 da pedra (2,6) e o 5 da pedra (2,5). Excepcionalmente, a carroça inicial pode receber até quatro encaixes.

[23] http://www.dominonet.com.br
[24] http://web.archive.org/20030603041953/http://www.dominonet.com.br/

Figura 54. Uma mesa de dominó.

Um passe ocorre quando um jogador não possui nenhuma pedra em sua mão que seja possível jogar na mesa. Nesse caso, a dupla adversária ganha 20 pontos.

Uma rodada acaba quando algum dos jogadores acaba as pedras que estão em sua mão, quando é dito que esse jogador "bateu", ou se nenhum dos quatro jogadores consegue fazer nenhuma jogada, quando é dito que "fechou o jogo". A nova rodada se inicia como jogador que bateu na jogada anterior, e ele deve jogar uma carroça. Se ele não possuir nenhuma carroça para jogar, então ele passa.

Um jogador faz ponto quando joga uma pedra na mesa e soma das pontas da mesa é um múltiplo de 5. Por exemplo, na Figura 54, o jogador não nenhum ponto, pois as pontas são 5 e 6, que somadas são 11, que não é múltiplo de 5.

Ganha o jogo a dupla que no final de uma rodada possuir mais pontos que equipe adversária e tiver atingido o limiar pré-estabelecido. Esse limiar de pontos normalmente é 100 ou 200 pontos.

Num jogo de dominó as pedras são sempre as mesmas, logo o ambiente virtual deve ter essas pedras já pré-publicadas. Além das pedras, as UPIs desse ambiente virtual são os pontos e as jogadas realizadas, formando um log de jogadas. Há também as relações entre as pedras e os jogadores, ou seja, a mão dos jogadores é representada através dessas relações. Na Figura 55, é apresentada um diagrama do documento desse VCom.

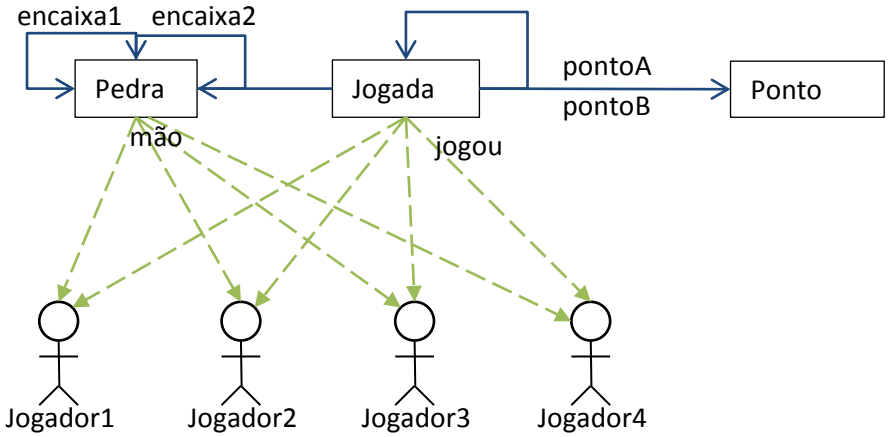


Figura 55. Diagrama do modelo de documento de um jogo de dominó.

A UPI "Jogada" não possui conteúdo, ela é uma UPI do tipo nulo. Ele serve para estruturação de outros elementos do Documento, nesse caso, a jogada promove uma sequência das pedras que foram jogadas e por quem foi jogada, que no caso é o autor da UPI "Jogada".

Esse VCom tem uma particularidade, ele possui UPIs pré-publicadas, as pedras. Elas são sempre as mesmas e não é possível jogar com menos, mais, ou pedras diferentes dessas. Dessa forma, quando um jogo de dominó é instanciado, as UPIs de pedras já estão publicadas no documento.

Algumas atividades desse VCom são:

- Embaralhar as pedras: atribuir aleatoriamente as pedras para os jogadores, publicando uma relação como rótulo "mão" entre as pedras e os jogadores;
- Jogar Pedra: Publica uma relação de encaixe com a pedra da ponta da mesa, publica uma UPI nulo com rótulo "Jogada", publica relação dessa jogada com a jogada anterior, publica uma relação entre esse UPI de jogada com a pedra jogada, processa os pontos e publica os devidos pontos da jogada e um relação da UPI pontos com essa jogada.

O Protocolo de Interação desse VCom é bem complexo, pois não pode dar margem a jogadas erradas, em horas erradas, ou fazendo algo que não é permitido no jogo. Além disso, precisa controlar as rodadas e a pontuação do jogo, de forma a declarar o vencedor.

Esse jogo possui um sistema de estados bem definidos. Depois de embaralhar as pedras entre os jogadores, o jogador que possui a carroça de sena deve começar o jogo. Após isso, um após o outro, os jogadores vão jogando seguindo sempre essa mesma ordem de que "eles sentaram na mesa". Após cada rodada, é necessário re-embaralhar as pedras e começar uma nova jogada. O Protocolo ainda precisa dar conta de quando acaba o jogo e quem foi o vencedor da partida. Na Figura 56, é apresentado um diagrama de estados ilustrando essas mudanças de estados.

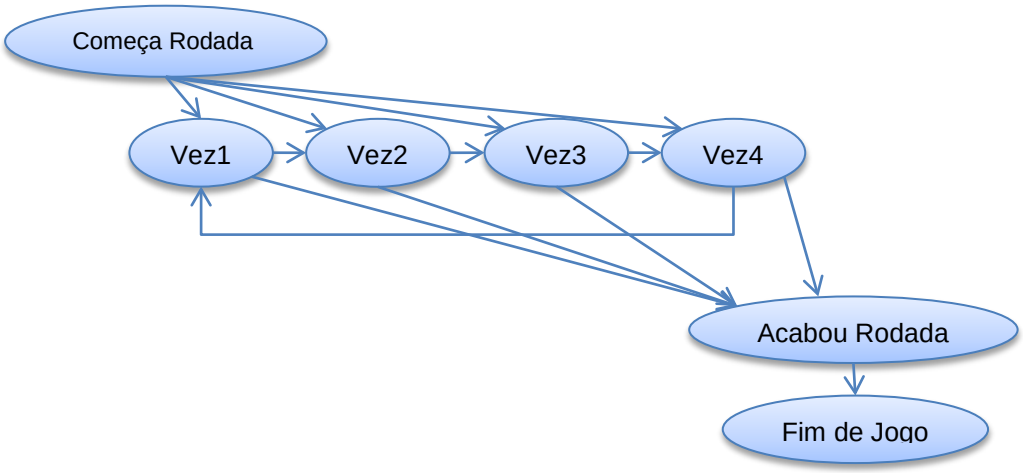


Figura 56. Diagrama de estados de um jogo de dominó.

Para cada estado há um conjunto diferente de permissões, de modo que somente o jogador da vez possa jogar. Cada jogador, nesse caso, assume um papel entre os quatro papéis possíveis. O Protocolo então controla as jogadas possíveis.

Mas existem permissões que se aplicam a qualquer estado do VCom, são elas:

- Os jogadores podem consultar as todas as UPIs Pedra;
- Os jogadores podem consultar suas mãos, ou seja, consultar as relações UPI-Usuário entre UPIs Pedra e ele mesmo;
- Os jogadores podem consultar todas as jogadas, as relações entre as jogadas, e as relações entre jogadas e pedras;
- Os jogadores podem consultar os Pontos e as relações entre pontos e jogadas;
- Qualquer pessoa (inclusive os espectadores) podem consultas a mesa de jogo, ou seja, as pedras e as relações de encaixa1 e encaixa2 entre as pedras.

No primeiro estado, **começa rodada**, algum jogador precisa embaralhar as pedras. Há um trigger que dispara a atividade de embaralhar as pedras. Essa atividade associa aleatoriamente uma pedra com um jogador, através de uma relação "mão", de forma que cada jogador fique com sete pedras na mão.

Feito isso, há outro gatilho que é ativado mediante uma condição: a de que todos os jogadores tenham sete pedras na mão. Esse gatilho muda o estado para a vez do jogador que possui a carroça de sena. Na verdade, trata-se de quatro gatilhos, um gatilho para cada jogador, verificando se todos possuem sete pedras e se ele possui a carroça de sena. Quando esse gatilho é ativado ele muda o estado da vez do jogador.

Os estados de **vez do jogador**, são muito parecidos entre si. Possuem permissões que permitem que um jogador, o jogador da vez, jogue uma pedra na mesa, e acionam os gatilhos de passe e pontos. As permissões desse tipo de estados são as seguintes:

- O jogador pode despublicar a relação da Pedra com a mão e relacionar essa Pedra com a Jogada, se essa pedra for a carroça de sena;
- O jogador pode encaixar uma pedra na outra (publicar uma relação de encaixa1 ou encaixa2 entre duas Pedras, publicar uma jogada e associar essa pedra com a jogada), se a ponta que não possui encaixe da pedra que está na mesa for igual a uma das pontas da pedra a ser jogada;
- Há uma exceção para a regra anterior para a carroça inicial, que pode ter até quatro encaixes (dois encaixes no "bucho"), enquanto as outras pedras só podem ter dois encaixes, uma para cada ponta;
- O jogador também pode jogador outra carroça (sem ser a carroça de sena), se ele tiver batido na jogada anterior;

Há ainda gatilhos que publicam os pontos. Eles verificam se a soma das pontas sem encaixe das pedras que estão na mesa é múltiplo de cinco. Se for, publica uma UPI Ponto (A ou B, respectivamente para cada dupla) com a quantidade de pontos marcada naquele jogada, e a associa com a última jogada.

Se a última jogada for do jogador da vez, então é acionado um gatilho de mudança de estado, passando para o estado da vez do próximo jogador.

Um terceiro gatilho desse estado é em relação a passes. Se o jogador não possui nenhuma pedra que ele possa jogar na mesa, então é publicada uma jogada sem associação com nenhuma pedra, e é atribuído vinte pontos para a dupla adversária associada com essa jogada. Esse trigger também faz com que o estado seja mudado para a vez do próximo jogador. Esse gatilho ainda precisa verificar se a última jogada foi também de passe, pois isso é considerado "passe nas costas" e não devem ser contabilizados pontos para ninguém.

Outro gatilho verifica se o jogador da jogada anterior não possui nenhuma pedra na mão, sinalizando que a rodada acabou. Esse gatilho, então, muda o estado para o estado de acabou rodada. Se não há nenhuma pedra possível de ser jogada, ou seja, o jogo está fechado, então esse gatilho também deve ser acionado.

No estado de **acabou rodada**, é verificado se o jogo acabou ou se é preciso mais uma rodada.

O jogo acaba quando uma das duplas atinge a uma determinada soma de pontos (geralmente 100 ou 200 pontos) ao final de uma rodada. Se isso acontece deve ser acionado um gatilho que muda o estado para o estado de **fim de jogo**. Ganha a dupla que tiver a maior soma dos pontos.

Se isso não ocorrer e o jogo não foi fechado, é acionado um gatilho que re-embaralha as pedras e as distribui entre os jogadores. Esse gatilho também é responsável por publicar uma jogada especial marcando que um dos jogadores bateu, e por mudar o estado para a vez de um dos jogadores, para aquele que bateu a rodada anterior.

Se não foi fim de jogo e o jogo foi fechado, outro gatilho é acionado. Ele deve re-embaralhar e distribuir as pedras e fazer com se troque de estado para o jogador que possuir a carroça de sena.

A Interface desse VCom é relativamente simples, é a mesma para qualquer dos estados. Para os jogadores é executado o template que exibe a mesa de jogo, sua mão

e as jogadas anteriores. Para os espectadores, é executado o template que exibe a mesa de jogo e as jogadas anteriores. Isso foi representado na Figura 57.

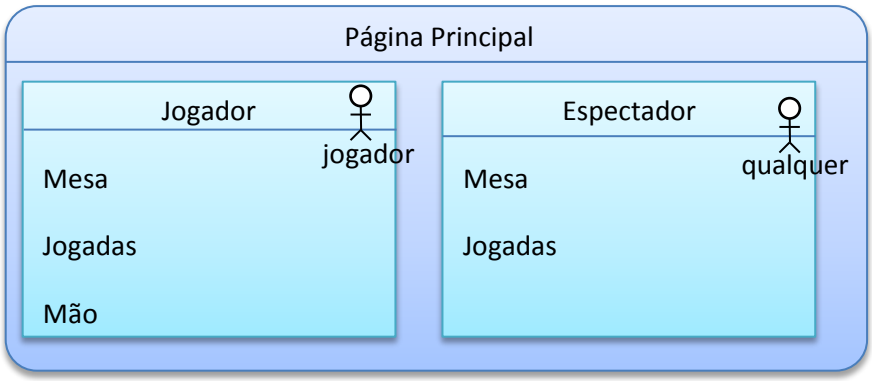


Figura 57. Diagrama de navegação da interface de um jogo de dominó.

11.8 Glossário

Um glossário é um ambiente virtual de construção de significados de palavras e termos, normalmente de um domínio específico ou de um tema que está sendo abordado. Essa construção ocorre de maneira coletiva, onde qualquer participante pode criar um significado para um termo e editar significados de outros termos.

Em sua ideia geral, um glossário se assemelha muito com um wiki, diferenciando pelo fato de não ser necessário existir um hiperlink para que um termo (ou página) exista. Dessa forma, o modelo de documento de um glossário pode ser o mesmo de um wiki, apresentado na Figura 51. No entanto, suas diferenças são aparentes no Protocolo de Interação e em sua Interface.

Algumas atividades desse VCom são:

- Incluir uma nova Página: publica uma UPI com rótulo “Página”, publica uma relação entre essa página e a página que tem um hiperlink para ela, cria relações dos hiperlinks da página criada;
- Editar uma página: edita o conteúdo da UPI, e atualiza (insere e/ou remove) as relações de hiperlinks entre as páginas.

11.9 Chat

Uma sessão de chat é um ambiente virtual para diálogos síncronos. Qualquer participante do chat pode publicar suas mensagens. A ordem dessas mensagens é dada pela data de publicação das mesmas. Cada vez que uma nova mensagem é publicada, os outros participantes imediatamente a recebem.

O modelo de um documento de um chat é bem simples. Admite a publicação de mensagens apenas, como pode ser visto na Figura 58. Diferente das mensagens instantâneas do Facebook, essas mensagens não são direcionadas a um usuário específico, mas a todos os participantes do chat.

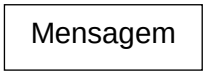


Figura 58. Diagrama do modelo de documento de um chat.

A atividade desse VCom é:

- Enviar Mensagem: publica uma UPI com o rótulo “Mensagem”.

11.10 Mural

Um mural é uma coleção de postagens (UPIs), sem réplica, com prazo de expiração, postados de forma assíncrona. Difere de um chat em duas características: é um veículo de interação assíncrona, e o chat utiliza interação síncrona; as mensagens tem um prazo de expiração.

Ambas as características que diferem um mural de um chat não influenciam seu modelo de Documento. As modificações ficam a cargo da Interface. Assim, o modelo de Documento de um mural é o mesmo de um chat, como apresentado na Figura 58.

A atividade desse VCom é:

- Incluir Postagem: publica uma UPI com o rótulo Mensagem.

11.11 Debate de Teses (Construindo Conceituações)

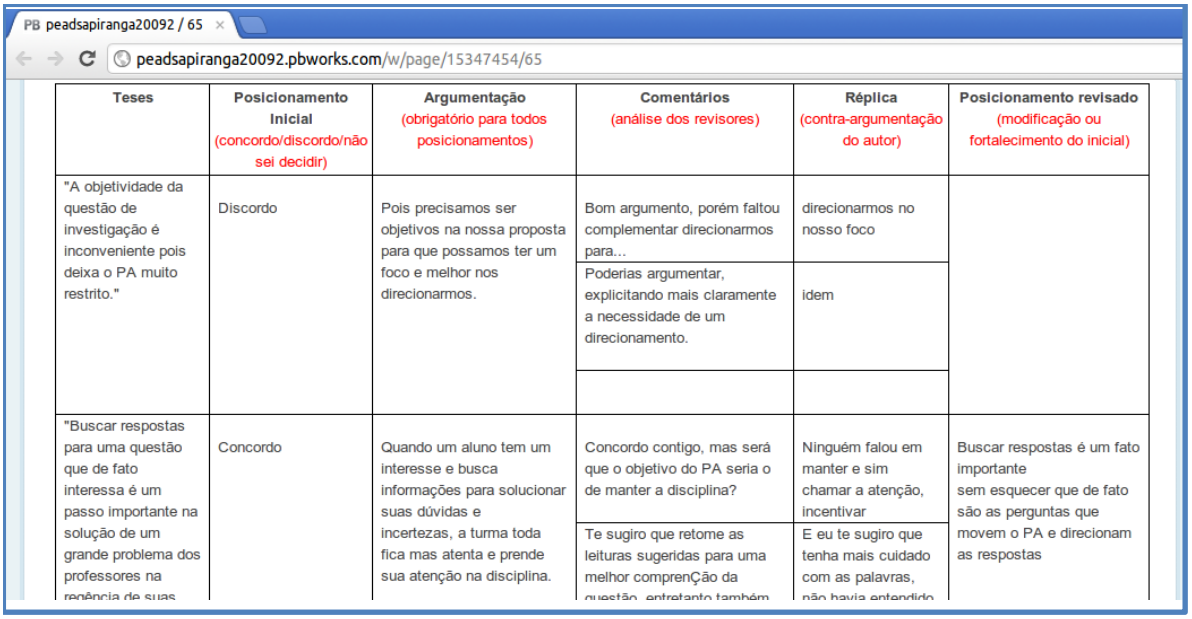
A arquitetura pedagógica "Construindo Conceituações", também chamada de "Debate de Teses", foi desenvolvida por Nevado, Dalpiaz e Menezes (2009) no contexto de um curso de formação continuada de tutores. "Partindo de um arcabouço inicial, as etapas, produtos e interações foram sendo definidas ao final de cada etapa, sendo inclusive necessária a inserção de novas etapas". Como resultado, foi elaborada uma atividade com cinco etapas.

Na primeira etapa o moderador (professor) seleciona as teses a serem debatidas a partir de um levantamento sobre o conhecimento prévio dos estudantes. É feito um quadro, com as teses e cada aluno (etapa 2) deve se manifestar através de um posicionamento de concordância ou discordância, seguido de uma justificativa. A seguir (etapa 3), dois ou mais membros do grupo, são designados para analisar, através de um parecer, cada argumentação. Na etapa 4 os autores da argumentação podem, se desejarem, fazer uma réplica ao comentários dos avaliadores. Na etapa 5 os usuários, podem, apresentar uma nova versão de seus posicionamentos e argumentos. O documento resultante possui uma estrutura e um protocolo de interação complexo que não tem sido contemplado pelas ferramentas usuais. Pode-se ver a estrutura geral deste documento na Figura 59.

Os autores relatam que para experimentação foi adotado uma ferramenta wiki para implementar o ambiente mediador que se desejava. No entanto, as regras de interação e o cronograma para a criação do documento eram disponibilizadas de forma verbal, e esperava-se que cada participante respeitasse e se responsabilizasse pelo cumprimento do estabelecido. Na Figura 60, pode-se ver uma tela de um desses debates de teses funcionando num wiki.

Autor					
Revisores					
Teses	Posicionamento Inicial (concordo / discordo / não sei dizer)	Argumentação (obrigatório para todos os posicionamentos)	Comentários (análise dos revisores)	Réplica (contra-argumentação do autor)	Posicionamento Revisado (modificação ou fortalecimento do inicial)
Etapas: [1]	[2]	[3]	[4]	[5]	[6]
Tese 1					
...					
Tese n					

Figura 59. Exemplo de quadro de discussão da arquitetura pedagógica Construindo Conceituações.



Teses	Posicionamento Inicial (concordo/discordo/não sei decidir)	Argumentação (obrigatório para todos posicionamentos)	Comentários (análise dos revisores)	Réplica (contra-argumentação do autor)	Posicionamento revisado (modificação ou fortalecimento do inicial)
"A objetividade da questão de investigação é inconveniente pois deixa o PA muito restrito."	Discordo	Pois precisamos ser objetivos na nossa proposta para que possamos ter um foco e melhor nos direcionarmos.	Bom argumento, porém faltou complementar direcionarmos para...	direcionarmos no nosso foco	
			Poderias argumentar, explicitando mais claramente a necessidade de um direcionamento.	idem	
"Buscar respostas para uma questão que de fato interessa é um passo importante na solução de um grande problema dos professores na regência de suas	Concordo	Quando um aluno tem um interesse e busca informações para solucionar suas dúvidas e incertezas, a turma toda fica mas atenta e prende sua atenção na disciplina.	Concordo contigo, mas será que o objetivo do PA seria o de manter a disciplina?	Ninguém falou em manter e sim chamar a atenção, incentivar	Buscar respostas é um fato importante sem esquecer que de fato são as perguntas que movem o PA e direcionam as respostas
			Te sugiro que retome as leituras sugeridas para uma melhor comprenÇão da questão, entretanto também	E eu te sugiro que tenha mais cuidado com as palavras, não havia entendido	

Figura 60. Tela de um debate de teses funcionando num wiki.

Dessa forma, podem-se identificar algumas UPIs: as teses; a argumentação; os comentários sobre uma argumentação; as réplicas e o posicionamento revisado. Como pode ser visto na Figura 61, abaixo.

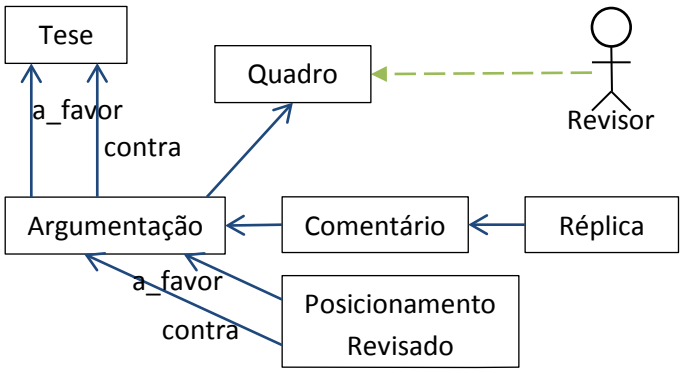


Figura 61. Diagrama do modelo de documento de um quadro de debate de teses.

Algumas atividades desse VCom são:

- Incluir Tese: publica uma UPI Tese;
- Incluir Argumentação marcando um posicionamento: publica uma UPI Argumentação e a relaciona com a Tese com o rótulo da relação sendo seu posicionamento inicial, se o autor ainda não possui um quadro publica uma UPI Quadro do tipo nulo e relaciona a Argumentação com esse Quadro;
- Incluir Comentário de uma Argumentação: publica uma UPI Comentário e relaciona essa UPI com a Argumentação;
- Incluir Réplica de um Comentário: publica uma UPI Réplica e relaciona essa UPI com o Comentário;
- Incluir Posicionamento Revisado: publica uma UPI Posicionamento Revisado e relaciona essa UPI com a Argumentação Inicial com o rótulo do posicionamento.

11.12 CARTOLA

Trata-se de **objeto de aprendizagem** chamado **CARTOLA** (ALVES, NUNES, *et al.*). Nesse jogo, o participante deve escrever um texto a partir de três cartas sorteadas. As cartas podem conter são palavras ou figuras.

No jogo original, não havia a possibilidade de interação entre os jogadores, mas já havia a intenção de seus criadores em colocar esse tipo de funcionalidade. O tipo de interação mais simples é o de possibilitar a leitura e inclusão de comentários nas histórias dos outros participantes.

Assim, é possível identificar algumas UPIs: as cartas, as histórias e os comentários. Conforme isso, foi modelado esse VCom na Figura 62.

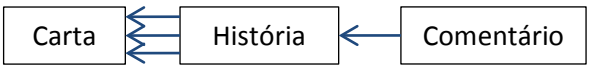


Figura 62. Diagrama do modelo de documento de um jogo CARTOLA.

Uma característica interessante desse VCom é que as UPIs Cartas devem estar previamente publicadas, antes da instanciação desse VCom, para que ele funcione corretamente. Esse conjunto de cartas inicial não é necessariamente fixo, pode ser modificado.

Algumas atividades desse VCom são:

- Inserir Carta: publica uma UPI com rótulo UPI;
- Escrever História: publica uma UPI História e a relaciona com as cartas sorteadas (na Interface);
- Inserir Comentário de História: publica uma UPI Comentário e associa essa UPI com uma História.

11.13 Júri Simulado

Trata-se de um quadro de discussões para a arquitetura pedagógica **Júri Simulado** (REAL e MENEZES, 2007). Nessa arquitetura pedagógica, os participantes são divididos em três grupos, Acusação, Defesa e Júri, mais o Juiz. O caso a ser julgado é um assunto geralmente polêmico, que divide opiniões. O papel do Juiz é apenas intermediar o debate entre Acusação e Defesa, e o do Júri é votar a favor ou contra, quando chegar à etapa determinada.

Na verdade, são dois fóruns, o da Acusação e o da Defesa. O Juiz pode interagir em ambos os fóruns. Ao final de um prazo estipulado, o Júri, que lê ambos os fóruns, deve votar se quem deve ganhar é a Acusação ou a Defesa. Aquele que receber a maioria dos votos ganha. Os dois fóruns e os votos podem ser modelados conforme o diagrama da Figura 63.

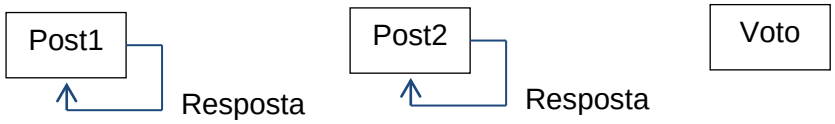


Figura 63. Diagrama do modelo de documento de um quadro de discussão para o júri simulado.

Algumas atividades desse VCom são bem parecidas com as de um Fórum:

- Incluir Post-Raiz da Acusação: publica uma UPI com o rótulo "Post1";
- Incluir Post-Resposta da Acusação: publica uma UPI como o rótulo "Post1" e publica a relação de resposta entre esse post e outro post;
- Incluir Post-Raiz da Defesa: publica uma UPI com o rótulo "Post2";
- Incluir Post-Resposta da Defesa: publica uma UPI como o rótulo "Post2" e publica a relação de resposta entre esse post e outro post;
- Votar: publica uma UPI com o rótulo "Voto" com o conteúdo do voto.

11.14 Diário Virtual de Resolução de Problemas

O Diário Virtual de Resolução de Problemas de Matemática (SERRES e BASSO, 2009) é uma espécie de revisão por pares, onde os participantes devem registrar suas soluções e comentar as soluções dos outros estudantes. É parecido com a arquitetura pedagógica "Construindo Conceituações", mas numa versão mais simples.

Um problema é proposto pelo professor em sala de aula. Os estudantes devem resolvê-lo e descrever cada passo de suas tentativas de resolução numa espécie de diário.

Em um segundo momento, é a etapa de revisões iniciada. Cada estudante deve comentar a solução de pelo menos outros dois estudantes. Serres e Basso comentaram que nessa essa etapa surpreendeu suas expectativas. Eles imaginavam que os estudantes fossem fazer isso somente por obrigação, ou de mau gosto. Mas na verdade, os estudantes tiraram bastante proveito, comentaram bastante mesmo além da exigência de dois comentários e responderam aos seus revisores. Como todos

trataram do mesmo problema, essa etapa contribui para um melhor entendimento do problema proposto.

De forma semelhante aos autores da arquitetura "Construindo Conceituações", Serres e Basso utilizaram uma ferramenta de wiki para implementar essa atividade. Cada estudante possuía uma página na qual tinha que escrever seu diário de resolução do problema proposto e também receber os comentários dos revisores. Essa página possuía um modelo padrão que foi copiado para participante. Claramente isso seria uma dificuldade na implementação de outras instâncias dessa atividade e também na modificação da mesma.

A estrutura do Documento desse VCom é bem similar a de um Blog. Cada participante publica um post, que é seu diário de resolução do problema proposto, e recebe comentários de seus revisores. Na Figura 64, está representado o modelo desse Documento, muito similar ao do Blog, apresentado na Figura 42.

Figura 64. Diagrama do modelo de documento de um Diário de Resolução de Problemas.

Algumas atividades desse VCom são:

- Incluir Post: publica uma UPI com o rótulo "Post";
- Edita Post: edita o conteúdo da UPI publicada;
- Incluir Comentário: publica uma UPI com o rótulo "Comentário" e publica a relação desse comentário com um Post.

Algumas pesquisas desse VCom são:

- Pesquisa diário de um usuário;
- Pesquisa comentários de um Diário.
- Pesquisa todos os diários;

Sobre o Protocolo, esse VCom possui dois estados (Inicial e Revisões), não possui grupos e tem dois papeis: participante e professor. Para cada estado, há um conjunto de permissões e esquemas de navegação definidos.

Estado Inicial. Esta é a fase em que os participantes escrevem (e editam) seus diários individualmente, descrevendo como elaboraram suas resoluções do problema proposto. Eles só podem ter acesso ao diário que eles mesmos estão produzindo, e não podem ver as respostas dos outros participantes. Dessa forma, as permissões nesse estado são:

- Um participante pode consultar a UPI Post que ele publicou;
- O professor pode consultar todo o Documento;
- Um participante pode publicar uma UPI Post, se ele já não tiver publicado nenhuma UPI Post anteriormente;
- Um participante pode alterar o conteúdo de uma UPI Post publicada;
- O professor pode mudar do estado "Inicial" para o estado "Revisões".

Nesse estado, o professor pode acompanhar como andam todos os diários, enquanto cada participante só visualiza sua própria página. Na Figura 65, é apresentado o diagrama de navegação para esse estado inicial: os participantes visualizam apenas o próprio texto de seus diários, enquanto o professor pode visualizar o texto de todos os participantes. É apresentada apenas uma página, a inicial, com templates diferentes para os dois papéis de usuários.

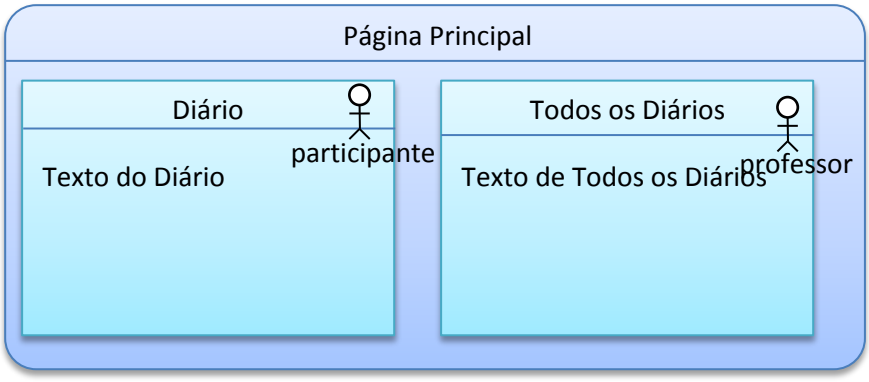


Figura 65. Diagrama de navegação da interface de um diário de resolução de problemas, no estado inicial.

Estado de Revisões. Nesse estado, os participantes podem iniciar as revisões das respostas de seus colegas, acessando uma lista de participantes e examinando o conteúdo do texto apresentado na solução de seus colegas. A partir daí, os estudantes fazem comentários das soluções dos outros, podendo iniciar uma discussão por meio

desses comentários. Ao professor, também é possibilitado entrar nessa discussão e fazer comentários a qualquer um dos participantes. Algumas permissões nesse estado são:

- Qualquer um pode consultar todo o Documento;
- Um participante ou professor pode publicar Comentários, e publicar relações desse comentário com um Post;

No estado de revisões, todos visualizam o mesmo conjunto de templates, divididos em duas páginas: uma lista de diários de participantes, que possui links para uma página ao diário de um participante específico. Na Figura 66, é apresentado o diagrama de navegação para esse estado de revisões.

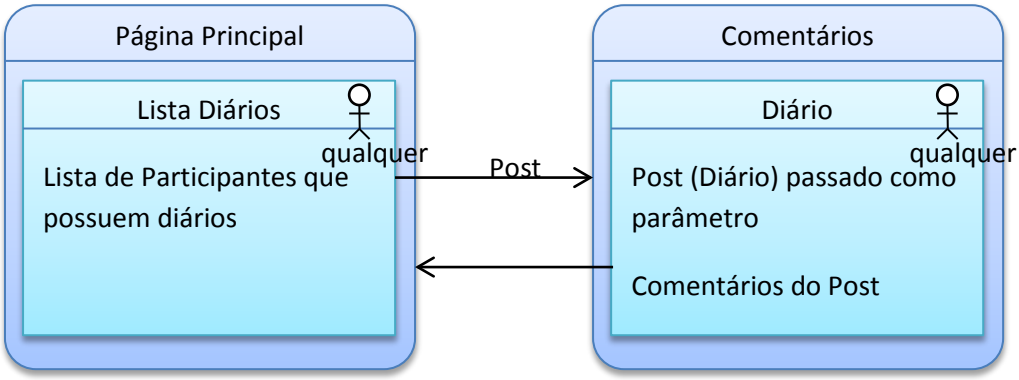


Figura 66. Diagrama de navegação da interface de um diário de resolução de problemas, no estado de revisões.

11.15 Projeto de Aprendizagem

11.16 Curso em Ambiente Virtual de Aprendizagem

11.17 Bolão de Apostas da Copa do Mundo

11.18 Apresentação de Artigos

11.19 Estudo Bíblico

11.20 Interpretador de Código Haskell