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

# Um Modelo Conceitual para Ambientes Virtuais Flexíveis

Leonardo Nascimento dos Santos

Manaus/AM
2014


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

Leonardo Nascimento dos Santos

# Um Modelo Conceitual para Ambientes Virtuais Flexíveis

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

Orientador:
Prof. Dr. Alberto N. de Castro Júnior

Co-orientador:
Prof. Dr. Crediné Silva de Menezes
(Universidade Federal do Rio Grande do Sul)

Manaus/AM
2014

# Agradecimentos

Quero agradecer primeiramente a Deus, pela imensa bondade e misericórdia para comigo.

Quero agradecer aos meus pais, seu Lúcio e dona Suely, que em tudo me incentivaram e me proporcionaram: um ambiente de amor e incentivo, de modo que eu pudesse sempre continuar os meus estudos. Quero agradecer também aos meus irmãos, que a seu modo e com a sua convivência pude ser incentivado a continuar estudando.

Quero agradecer à minha amada e adorável esposa Adrienne. Ela que tem estado ao meu lado antes mesmo de eu iniciar essa jornada que foi o curso de Doutorado. Com ela compartilhei minhas aflições, minhas dúvidas, meus sucessos e meus fracassos, que por vezes me levaram a pensar em desistir. Mas como principal motivadora, ela me colocava de pé, me animava e restaurava a minha Fé. Te amo!

E como não deixar de falar da nossa filha, que nasceu e cresceu nessa confusão desse curso de doutorado. Que por vezes não me viu sair ou chegar em casa, pois ainda/já estava dormindo. Loren, a ti agradeço por me ensinar preciosas lições de como o Pai estava cuidando de nós.

Quero agradecer especialmente ao meu orientador, que me fez começar e terminar esse curso de doutorado. A seu modo, me fez pensar, me incentivar e crescer como pessoa e como pesquisador. Por vezes, passamos meses sem reuniões, mas eu sabia que ele estava me ensinando ainda assim. Agradeço pelas conversas de hipóteses imaginárias que eu tanto apreciava, tanto dele quanto do meu co-orientador, professor Crediné Menezes. Acredito que, mesmo não tendo feito a melhor pesquisa que eu poderia ter feito, eu não faria curso melhor de doutorado, aprendendo a pensar e a criticar, graças ao meu orientador Alberto Castro.

Agradeço também aos amigos da Neemu pelo incentivo e cuidado por mim, mesmo quando não pude me aplicar da melhor forma.

*O temor do SENHOR é o princípio do saber, mas os loucos desprezam a sabedoria e o ensino.*

– Provérbios 1:7

*O que foi é o que há de ser; e o que se fez, isso se tornará a fazer; nada há, pois, novo debaixo do sol. Há alguma coisa de que se possa dizer: Vê, isto é novo? Não! Já foi nos séculos que foram antes de nós.*

– Eclesiastes 1:9-10

*Pois que aproveitará o homem se ganhar o mundo inteiro e perder a sua alma? Ou que dará o homem em troca da sua alma?*

– Mateus 16:26

# Resumo

Nos últimos dez anos, a Web tem se transformado numa grande plataforma de interação entre seus usuários. Esses usuários têm demandado cada vez mais flexibilidade das ferramentas que eles utilizam, tal que elas permitam que a interação aconteça de acordo com suas necessidades. Essa lacuna é ainda mais acentuada em ambientes virtuais de suporte a atividades de domínios complexos, como o da Educação. Esses ambientes devem ser condicionados pela natureza evolutiva das pessoas e devem refletir a evolução de suas necessidades. Assim, o objetivo deste trabalho é conceber um modelo conceitual para ambientes virtuais web de interação humano-computador-humano, de forma que os ambientes construídos a partir desse modelo sejam flexíveis em sua criação e modificáveis em tempo de execução. Na busca por tratar esse problema, tem-se investigado uma proposta inovadora para concepção de ambientes virtuais com ênfase na cooperação chamada MOrFEu. Neste trabalho, apresenta-se uma extensão desse modelo conceitual com a capacidade de descrever detalhadamente cada um dos componentes de um ambiente virtual. Como prova de conceito, foram modelados diversos ambientes virtuais característicos e, por suas características peculiares, alguns deles foram escolhidos para serem analisados através da implementação de protótipos. Os ambientes selecionados foram avaliados com respeito à sua flexibilidade, inclusive mostrando a possibilidade de alterações em tempo de execução através de alterações em suas descrições, evidenciando assim que o modelo proposto contribui para que a construção e adaptação de ambientes virtuais, que atendam às demandas de interação de seus usuários, sejam ampliadas de forma significativa.

**Palavras-chave**: Ambientes Virtuais Flexíveis; Groupware; Interação Humano-Computador-Humano.

# Abstract

Over the past decade, the Web has become a major platform for interaction among its users. These users have demanded more flexibility from their the tools, in a way that these tools could support interaction according to needs. This gap is even deeper in virtual environments supporting activities in complex domains like Education. These software tools should be driven by the evolving nature of people and should reflect evolution on their needs. Thus, the main objective of this work is to design a conceptual model for web-based human-to-computer-to-human-interactive virtual environments, so that these environments are flexible for creation and runtime modifiable. In search for dealing with this problem, we have investigated an innovative proposal for devising of virtual environments, called MOrFEu. In this work, we present an extension of this conceptual model able to describe in detail each component of a virtual environment. As proof of concept, we have modeled typical virtual environments and, because of their peculiar characteristics, some of them were chosen to become prototype implementations. The selected environments were evaluated with respect to their flexibility, including their ability for runtime modification through direct changes in their descriptions, giving evidence that the proposed model contributes to development and adaptation of virtual environments, which meet the interaction demands of their users, were significantly increased.

**Keywords**: Flexible Virtual Environments; Groupware; Human-to-Computer-to-Human Interaction.

# Lista de Figuras

# Lista de Tabelas

# Sumário

# 1 Introdução

O desenvolvimento das ferramentas de software não tem conseguido acompanhar as múltiplas demandas por interação e cooperação através da Web, ressaltando a necessidade por ferramentas flexíveis, que permitam que a interação aconteça de acordo com as necessidades de seus usuários, ou como colocado por Paredes (2007): "as ferramentas web de suporte à interação entre as pessoas devem ser condicionadas pela natureza evolutiva das pessoas e devem refletir a evolução de suas necessidades, adaptando seus requisitos de usabilidade e sociabilidade".

A Web também tem se transformado de uma "web de leitura" para uma "web de leitura e escrita" (ULLRICH, *et al.*, 2008), ou seja, tem ido ao encontro 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.

Mas Fuks *et al.* (2007) lembra 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. Este problema é mais agudo quando se trata de aplicações para domínios complexos como a Educação, que demanda ferramentas mais flexíveis, que possam adaptar-se a um processo de descobertas e invenções em busca de

soluções para questões significativas do ponto de vista dos atores envolvidos. Esse caráter "artesanal" implica que as atividades serão diferenciadas para cada grupo de usuários, demandando facilidades para a modificação e adaptação das ferramentas disponíveis, tendo em vista que cada grupo tem sua própria maneira de realizar qualquer atividade.

Na tentativa de solucionar o problema da falta de flexibilidade nos ambientes virtuais web de interação humano-computador-humano, desenvolvedores têm apresentado propostas relacionadas, dentre as quais destacam-se aquelas que são baseados em módulos (DOUGIAMAS e TAYLOR, 2003), em componentes (FARIAS, PIRES e SINDEREN, 2000) (ROTH e UNGER, 2000) (WON, STIEMERLING e WULF, 2005) (GEROSA, *et al.*, 2006), em linhas de produto de software (GADELHA, *et al.*, 2009), em *web-services* e agentes (PESSOA, NETTO e MENEZES, 2002) (MEDEIROS, 2005).

No entanto, essas abordagens não são capazes de tratar os tipos de modificações dinâmicas que os usuários estão demandando. Todas essas abordagens tratam o ambiente virtual como **uma aplicação concluída, que não precisará de modificações depois de entregue para o uso**. Consideram que o processo de mudanças nas necessidades dos usuários pode esperar pela reformulação das características da ferramenta por parte dos desenvolvedores. Ou seja, eventualmente a aplicação irá se tornar inadequada, e quando isso acontecer será necessário reformulá-la, recomeçando o projeto. Assim, essas abordagens ainda não são flexíveis o suficiente para lidar com as demandas em domínios notadamente dinâmicos, como é o caso da Educação.

Além das abordagens já citadas, destaca-se um modelo conceitual inovador de ambientes virtuais baseados na Web, o MOrFEu (MENEZES, *et al.*, 2008), no qual seus autores organizam a estrutura e o uso de ambientes virtuais de uma forma diferenciada, centralizado no processo de autoria.

No ponto de vista do MOrFEu, 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.

De modo a aprofundar os elementos originais do MOrFEu, foi feita uma avaliação de suas capacidades, passando por concretizações das ideias em artefatos, implementações de ambientes virtuais que foram construídos segundo sua perspectiva. Através desse estudo foi possível confirmar que as premissas eram de fato válidas e identificar limitações na proposta apresentada por Menezes *et al* (2008). Analisando-se essas caraterísticas e limitações, juntamente com as características dos ambientes virtuais candidatos a serem implementados na arquitetura desenvolvida, propôs-se uma extensão daquele modelo conceitual que foi denominado **MX**, acrônimo para MOrFEu eXtendido, elemento central deste trabalho. Além de cuidar das limitações identificadas, os elementos conceituais foram aprofundados, descrevendo precisamente a ideia e o funcionamento dos artefatos construídos segundo essa perspectiva.

Com o modelo conceitual MX, foi possível especificar precisamente 19 tipos diferentes de ambientes virtuais, desde os mais usuais como fórum, *blog* e *chat*, passando por aqueles que dão suporte a atividades complexas de 'arquiteturas pedagógicas' (CARVALHO, NEVADO e MENEZES, 2005), até sistemas com um intricado conjunto de estados e interações como é o caso de um jogo de Dominó On-line, onde são incorporados agentes de coordenação.

Algumas das ferramentas especificadas, por terem características peculiares, foram implementadas e submetidas a alterações. As modificações na definição desses ambientes alteraram diretamente as funcionalidades dos respectivos protótipos, evidenciando a adequação da proposta aos objetivos definidos.

## 1.1 Objetivo

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

Foram definidos os seguintes objetivos específicos:

- Desenvolver uma plataforma funcional para avaliação dos elementos conceituais propostos pelo MOrFEu
- Caracterizar um conjunto de ambientes virtuais para avaliação de modelo conceitual para desenvolvimento de ambientes virtuais flexíveis na Web

## 1.2 Tese e Hipóteses

A ideia central neste trabalho é de que **os elementos conceituais apresentados pelo MOrFEu possibilitam a concepção e desenvolvimento de ambientes virtuais flexíveis na Web**.

Algumas hipóteses iniciais relacionadas a essa tese são:

H1. Tais elementos são suficientes para especificação de qualquer instância de ambiente virtual utilizado em certo domínio de aplicação.

H2. É possível implementar ferramentas que foram especificadas com os elementos do MOrFEu em ambientes virtuais.

H3. É possível definir esquemas de representação que permitam que modificações na especificação de um ambiente virtual produzam modificações correspondentes na ferramenta implementada. Isso significa que a ferramenta implementada é flexível.

H4. É necessário o uso de banco de dados com esquema flexível para que se obtenha ambientes virtuais que capacidade de alteração em tempo de execução.

## 1.3 Metodologia

A investigação se iniciou com a **revisão bibliográfica** mapeando tópicos pertinentes que conduziram à identificação de trabalhos relacionados onde foi possível constatar que os relatos disponíveis na literatura, tratam o desenvolvimento de ambientes virtuais para apoiar a realização de atividades cooperativas como o desenvolvimento de uma ferramenta de software tradicional, com pouca atenção para as características dos usuários e as diferentes tarefas por eles desenvolvidas. Nessa etapa foi definido um recorte teórico inicial para o projeto.

Buscando uma solução para o problema de flexibilidade em ambientes virtuais, integramos a "força-tarefa" para a proposta do **modelo conceitual inicial** do MOrFEu que apresentou um modo diferente de pensar a concepção e desenvolvimento de ambientes virtuais.

A etapa seguinte envolveu a **avaliação da proposta** do MOrFEu de modo a verificar se ela era adequada para modelar ou explicar as ferramentas atuais de comunicação/interação.

A Hipótese H1 foi verificada através de **prova de conceito**, buscando contraexemplos que refutassem a hipótese. Após a **análise** das principais ferramentas de comunicação/interação conhecidas e ambientes virtuais tradicionais, percebeu-se que os mesmos podem ser descritos segundo os elementos propostos pelo MOrFEu. Também foi concebida uma **representação diagramática** para essas ferramentas e ambientes. Dessa forma foram modeladas várias dessas ferramentas, buscando um contraexemplo, que não foi encontrado.

A Hipótese H2 foi verificada também através de **prova de conceito,** buscando contraexemplos para refutar a hipótese, contraexemplos esses que não foram encontrados. Foram implementadas algumas das ferramentas modeladas e, para agilizar o processo de implementação, foi criado um **meta-ambiente virtual** para o desenvolvimento de ferramentas de software segundo os conceitos do MOrFEu.

Esses ambientes virtuais foram implementados utilizando banco de dados com esquema flexível, como apontado na Hipótese H4.

Houve então um conjunto de ações de **prototipação** que envolveu tanto a reimplementação de ferramentas que já existiam concretamente em alguma forma de produto, quanto a prototipação de várias outras que haviam sido descritas ou especificadas mas para as quais não se tinha nenhuma implementação conhecida.

De modo a verificar a viabilidade da proposta em uma situação real de uso, foi realizado um **estudo de caso** envolvendo o uso de um ambiente virtual implementado para suporte a uma estrutura didática chamada Debate de Teses.

Um resultado importante do estudo de caso foi que muitas das solicitações de modificação puderam ser realizadas em tempo de execução, com pouco tempo de desenvolvimento, o que corroborou a Hipótese H3. Enquanto que a partir das solicitações não atendidas foi possível **identificar limitações,** principalmente no protocolo de interação do modelo conceitual proposto. Buscou-se então **caracterizar situações** similares que ocorreriam com o uso de outras ferramentas de comunicação e interação. Assim, foi possível identificar outras limitações para o modelo conceitual MOrFEu, ao mesmo tempo em que ficavam ainda mais claro vários elementos comuns a todos esses ambientes virtuais.

Assim, foi **refutada a Hipótese H1**, de que os elementos iniciais do MOrFEu eram suficientes para especificação de qualquer instância de ambiente virtual no domínio da Educação. Ela foi anulada através de um contraexemplo apresentado.

Então, foi proposta uma nova hipótese:

H1.1. Os elementos conceituais apresentados inicialmente pelo MOrFEu **acrescidos de alguns elementos** possibilitam a especificação de qualquer instância de ambiente virtual utilizado em certo domínio de aplicação.

Desenvolvemos então uma **extensão do modelo conceitual** proposto pelo MOrFEu, com objetivo de tratar limitações e descrever precisamente um ambiente

virtual que seja flexível em tempo de execução. Essa extensão não distorce os conceitos inicialmente propostos pelo MOrFEu, sendo eles ainda válidos, sendo alguns conceitos acrescidos, tratando de questões que ainda não haviam sido tratadas. Dessa forma, ainda procuramos validar a tese principal.

Procurando validar a Hipótese H1.1, todas as ferramentas anteriormente usadas como objeto de estudo, além de outras inéditas que foram incorporadas ao conjunto, foram **modeladas** segundo o novo modelo, de modo a caracterizar as mais diversas situações e características entre os ambientes virtuais conhecidos.

A especificação desse novo modelo foi auxiliada pela **formalização e teste** de elementos através de um ambiente de programação Prolog[1]. Utilizou-se uma **abordagem em espiral** onde as definições eram propostas; depois, tais definições eram implementadas em protótipos de ambientes virtuais construídos em Prolog; ajustes nas definições e no modelo conceitual eram realizados; então esses ajustes refletiam mais uma vez nas implementações realizadas; na sequência, ambientes mais complexos e que possuíam características de outros pontos do modelo conceitual eram implementados, dando início a novo ciclo do processo. Isso corroborou a Hipótese H2.

**Protótipos Web**, usando PHP e MySQL, também foram implementados, seguindo os mesmo conceitos e ideias sugeridas pelos protótipos em Prolog. Esses protótipos não necessitaram de ajuste no que já havia sido especificado no modelo. Mas ajudaram a construir os **elementos de interface**, que não haviam sido contemplados pelos protótipos em Prolog.

Para construção de tais protótipos, foi desenvolvido um **framework** que abrange a arquitetura e os algoritmos de funcionamento desses ambientes implementados.

[1] Foi utilizado o SWI-Prolog.

Esses protótipos também demonstraram a **factibilidade da proposta** e que os ambientes virtuais se tornavam concretos apenas com a definição declarativa de suas funcionalidades. Corroborando a Hipótese H2.

Foram feitos ainda **testes de alterações** em tempo de execução, através de **provas de conceito**, buscando um contraexemplo para a Hipótese H3.

Esses protótipos Web foram construídos com banco de dados que não possuía um esquema de dados flexível, logo **invalidou a Hipótese H4**. Os protótipos foram construídos usando um banco de dados relacional tradicional, e todos possuíam o mesmo esquema de dados. No entanto, isso não invalidou a Hipótese H3.

Em todas as provas de conceitos realizadas, foram buscados contraexemplos que invalidassem as hipóteses levantadas. No entanto, mesmo com os vários exemplos de ferramentas de comunicação/interação conhecidas não foi possível encontrar contraexemplos que anulassem as hipóteses. Essas ferramentas abrangem uma extensão grande de tipos de ferramentas de comunicação/interação, desde as mais simples e comuns, até àquelas com alto grau de complexidade e controle que dão suporte a atividades pedagógicas complexas.

Dessa forma, podemos afirmar que a Tese proposta é válida. Foram confirmadas as hipóteses H1.1, H2 e H3, que são suficientes para demonstrar a tese de que **os elementos conceituais apresentados pelo MOrFEu possibilitam a concepção e desenvolvimento de ambientes virtuais flexíveis na Web.**

## 1.4 Organização do texto

O texto desta tese é organizado a seguinte maneira: No próximo capítulo é feita uma contextualização do problema. No Capítulo 3 é apresentada a fundamentação teórica, os conceitos onde está ancorada esta tese e as definições dos objetos de estudo aqui considerados. No Capítulo 4 são apresentados trabalhos relacionados que buscam tratar o problema apresentado. No Capítulo 5 apresenta-se o MOrFEu e discute-se a

avaliação de seus elementos definidores, através da modelagem e estudos de caso realizados. No Capítulo 6 apresenta-se o elemento central desta tese – um modelo conceitual para ambientes virtuais flexíveis chamado MOrFEu eXtendido (MX). No Capítulo 7, é apresentada uma arquitetura e algoritmos para o funcionamento de ambientes virtuais segundo o modelo conceitual MX. No Capítulo 8 são discutidos os procedimentos e resultados desta tese, concluindo o trabalho.

# 2 Contextualização

Neste capítulo, apresenta-se uma perspectiva histórica da evolução da demanda por ambientes virtuais flexíveis, presentes desde o desenvolvimento das primeiras tecnologias de hipertexto na Web. Com a intensificação do uso da Web, tais demandas multiplicaram-se e alcançaram proporções bem maiores, culminando no cenário atual.

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. 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, mudando a forma como eles pensam e agem na rede, passando de simples espectadores a 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 computadores de mesa. 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, *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 para esse 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, *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, *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 num contexto dinamicamente configurável, oposto a uma visão organizacional estática e "industrial" (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 histórico no desenvolvimento de ferramentas de software para a Web. E na Seção 2.2, considera-se o domínio da Educação para evidenciar a necessidade por flexibilidade nas aplicações web.

[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".

## 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 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 cujos nós 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 o qual 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

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

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 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

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

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.

Drupal e Joomla, apesar de serem considerados sistemas de gerenciamento de conteúdo, permitem que sejam usadas suas funcionalidades e componentes de maneira similar aos demais da lista acima.

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.

### 2.1.2 Web 2.0

A nova forma como a Web tem sido utilizada na última década, 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, *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 atualizarem um elemento da interface do usuário por vez, ao invés de ter que atualizar a página inteira toda vez que alguma coisa mudar" (TAIVALSAARI, *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 JavaScript 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.

Algumas técnicas têm sido empregadas para entregar essas aplicações web especializadas para grupos de usuários de nicho, tais como *tailoring* e os *mashups*.

Técnicas de *tailoring* têm sido propostas para lidar com esses problemas (AHMADI, *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, *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 *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. Na Figura 1, é apresentada uma pirâmide com os participantes na elaboração colaborativa de ferramentas por usuários-finais.

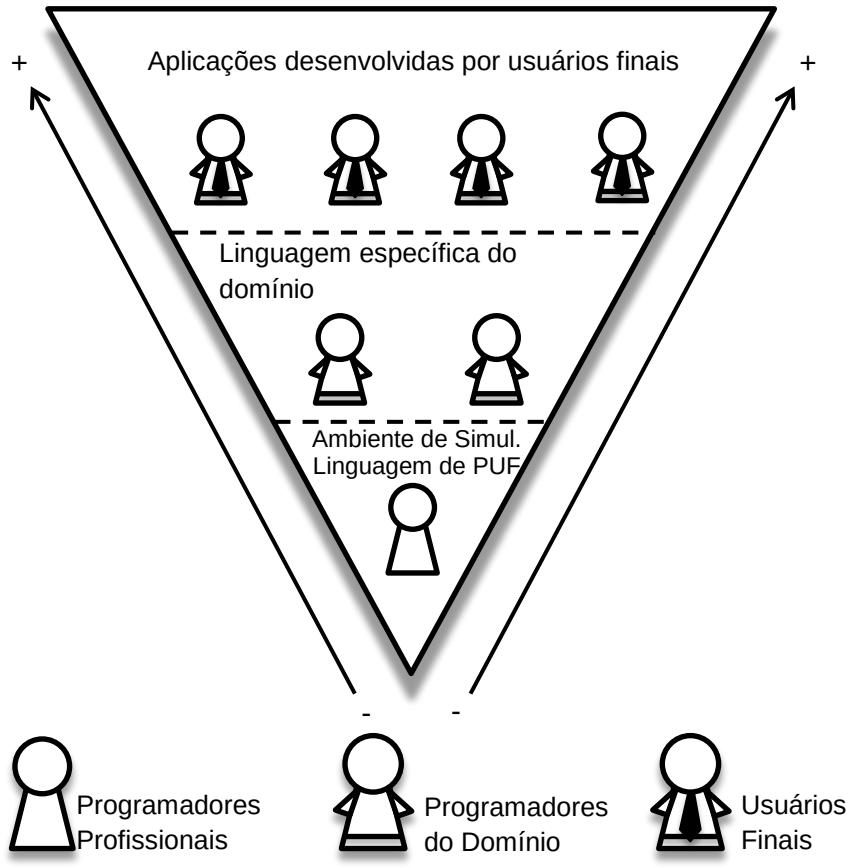


**Figura 1. Infraestrutura para construção de aplicações por usuários-finais. Fonte: Ahmadi *et al.* (2009).**

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 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 é um mediador, um interlocutor, 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, *et al.*, 2008).

Ou seja, por ter um caráter "artesanal", as atividades dessas arquiteturas serão executadas de maneira diferenciada 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 descrições resumidas de algumas arquiteturas pedagógicas. Descrições mais extensas podem ser encontradas no Apêndice A.

### 2.2.1 Projetos de Aprendizagem

Na arquitetura pedagógica Projetos de Aprendizagem (FAGUNDES, *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 (NEVADO, BASSO e BITTENCOURT, 2001). O trabalho de Monteiro *et al.* (2005) também propõe um ambiente virtual para desenvolvimento de projetos de aprendizagem.

### 2.2.2 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, apenas idealizada. Em suas especificações propôs que fosse utilizada uma ferramenta wiki para suporte às suas atividades.

### 2.2.3 Debate de Teses

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.

### 2.2.4 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 matemáticos.

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.

### 2.2.5 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 “Debate de Teses” 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.

**Tabela 1. Comparação das dificuldades encontradas nas arquiteturas pedagógicas.**

| | Projeto de Aprendizagem | Júri Simulado | Debate de Teses | 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 |

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.

## 2.3 Discussão

A partir desse levantamento realizado, é possível identificar os requisitos que estão sendo demandados dos ambientes virtuais que dão suporte a atividades complexas como as Arquiteturas Pedagógicas. Observa-se também que esses requisitos fazem intercessão com os requisitos de flexibilidade para ferramentas web que existem hoje em dia, e que os problemas se assemelham. Para isso, foi analisado o contexto da Web 2.0, do desenvolvimento de ambientes virtuais e, por fim, as 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 atribuída aos usuários**.

Esse aumento de produtores de aplicações é desejável, pois com mais produtores, 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, *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 dados de outras aplicações**, assim como os mashups. 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 ou à ferramenta utilizada. 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, *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 capaz de ser analisado pelas máquinas**, de forma que as máquinas possam auxiliar nas atividades rotineiras.

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

O que se observa atualmente é que as atividades são idealizadas de acordo com a disponibilidade das ferramentas para apoiar tais atividades. Ou seja, as ferramentas de apoio são limitantes no processo de desenvolvimento das atividades que se deseja realizar. Assim, acredita-se que as ferramentas devam ser flexíveis o suficientes de forma a não limitar o desenvolvimento da atividade em si, mas sim permitir a criação de atividades que sejam adequadas ao contexto que serão executadas.

## 2.4 Conclusão do Capítulo

Neste capítulo, analisou-se o contexto do problema discutido neste trabalho: a produção de ambientes virtuais flexíveis. Assim, podem-se resumir as necessidades evidenciadas na lista de requisitos 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 entanto, o foco deste trabalho é a busca por uma solução para o problema apresentado no requisito **R2: Os usuários devem ser capazes de fazer alterações nas aplicações conforme sua necessidade e em tempo de execução**.

Esse requisito está fortemente relacionado com o objetivo deste trabalho, de conceber ambientes virtuais flexíveis, modificáveis em tempo de execução.

Os outros requisitos levantados serão endereçados no decorrer do texto, ressaltando sua importância e endereçando trabalhos futuros.

# 3 Fundamentação Teórica

Neste capítulo, são abordados os conceitos que dão sustentação às ideias desta tese. São tratados os conceitos de "Ambiente Virtual", o principal objeto de estudo deste trabalho, e o conceito do "Guardião do Artefato", que é a principal fundamentação das ideias aqui propostas. O Modelo dos Teatros Sociais também serve de embasamento para a proposta deste trabalho, e também é abordado neste capítulo.

Primeiramente, são abordados os conceitos sobre o objeto de estudo deste trabalho: os ambientes virtuais web de interação humano-computador-humano. Começa-se com os conceitos de "virtual", "ambiente virtual", "interação", e "ambiente de interação humano-computador-humano" para situar a problemática deste trabalho. Depois são consideradas algumas tecnologias de suporte a ambientes virtuais de interação humano-computador-humano, visando situar os ambientes virtuais *web* de interação humano-computador-humano. Após isso, são apresentadas algumas ideias de sustentação desta tese como "o guardião do artefato", "agentes de time" e os "teatros sociais", visando embasar o argumento desta tese.

## 3.1 Ambiente Virtual

"Ambiente Virtual" é um termo bastante utilizado na atualidade, principalmente no contexto de educação à distância. São analisados conceitos relacionados com "virtual" e com "interação". Entre eles, define-se o que "ambiente virtual" e o que é "interação humano-computador-humano".

### 3.1.1 Virtual

Para se compreender o objeto de interesse deste trabalho, é preciso primeiramente discorrer sobre o conceito de virtual. Um dos filósofos a abordar o conceito do virtual e sua implicação no atual mundo das tecnologias de informação e comunicação em rede é Lévy.

Lévy (1996, p. 15) discorre acerca do virtual:

> A palavra virtual vem do latim medieval *virtualis*, derivado por sua vez de *virtus*, força, potência. Na filosofia escolástica, é virtual o que existe em potência e não em ato.

Em geral, a palavra virtual é associada com a ausência de realidade. "Acredita-se que uma coisa deva ser ou real ou virtual, que ela não pode, portanto, possuir as duas qualidades ao mesmo tempo. Contudo, a rigor, em filosofia o virtual não se opõe ao real mas sim ao atual" (LÉVY, 1999, p. 47). "Contrariamente ao possível, estático e já constituído, o virtual é como o complexo problemático, o nó de tendências ou de forças que acompanham uma situação, um acontecimento, um objeto ou uma entidade qualquer, e que chama um processo de resolução: a atualização" (LÉVY, 1996, p. 16).

Assim, uma coisa possível quando se torna real já era previsto o que iria acontecer. A realização do possível não é uma criação. O possível se realizará sem que nada mude em sua determinação nem em sua natureza. É um real fantasmático (LÉVY, 1996, p. 16).

Lévy propôs alguns exemplos de entidades virtuais para explicar o conceito.

A árvore está virtualmente presente na semente. A árvore é um complexo problemático pertencente à semente. O problema da semente é fazer brotar uma árvore. A partir das coerções que lhe são próprias, deverá inventá-la, coproduzi-la com as circunstâncias que encontrar. Dessa forma, haverá uma atualização da árvore que estava virtualmente presente na semente. Essa atualização gera outras situações virtuais, entrando assim num ciclo recursivo.

Uma palavra é uma entidade virtual. O vocábulo “árvore” está sempre sendo pronunciado em um local ou outro, em determinado dia numa certa hora. A enunciação deste elemento lexical é chamada de “atualização”. Mas a palavra em si, aquela que é pronunciada ou atualizada em certo lugar, não está em lugar nenhum e não se encontra vinculada a nenhum momento em particular (ainda que ela não tenha existido desde sempre).

O telefone é um sistema de telepresença, uma presença à distância, uma presença virtual. Ele não leva uma imagem ou representação da voz: transporta a própria voz. O telefone separa a voz do corpo tangível e a transmite à distância. Meu corpo tangível está aqui, meu corpo sonoro, desdobrado, está aqui e lá. E o corpo sonoro de meu interlocutor é igualmente afetado pelo mesmo desdobramento. De modo que ambos estamos, respectivamente, aqui e lá, mas com um cruzamento na distribuição dos corpos tangíveis. Ou seja, A conversa através do telefone é virtual, e é atualizada na recepção da voz no fone do telefone, e virtualizada no microfone.

Um hipertexto é uma entidade virtual. Um hipertexto é um conjunto de nós ligados por conexões. Os nós podem ser palavras, páginas, imagens gráficos, ou partes de gráficos, sequências sonoras, documentos complexos que podem eles mesmos ser hipertextos. Os itens de informação não são ligados linearmente, como em uma corda com nós, mas cada um deles, ou a maioria, estende suas conexões em estrela, de modo reticular. Navegar em um hipertexto significa, portanto desenhar um percurso em uma rede que pode ser tão complicada quanto possível. Dessa forma, uma navegação é uma atualização do hipertexto, feita de forma subjetiva pelo leitor.

Simuladores de vôo produzem mundos virtuais. Pois a atualização feita pelo “piloto” é feita de forma subjetiva e particular a cada vez que ele utiliza o simulador. O mundo do simulador existe de forma virtual não se prendendo ao tempo e ao espaço geográfico.

Dessa forma, Lévy atribui três atributos ao processo de virtualização: "desprendimento de um aqui e agora particular, passagem ao público e sobretudo heterogênese ['devir outro']" (LÉVY, 1996, p. 57).

### 3.1.2 Mundos e Ambientes Virtuais

Um mundo virtual é aquele que existe em potencial, não possui tempo nem espaço geográfico definido, de forma que esse mundo passa por um processo de atualização subjetiva da pessoa que está interagindo com esse mundo, passando por um processo de criação quando atualizado.

Lévy (1999) aponta três sentidos de mundos virtuais:

i. No sentido de possibilidade de cálculo virtual: um universo de possíveis calculáveis a partir de um modelo digital e de entradas fornecidas por um usuário. Por exemplo, quando se está visualizando uma foto digitalizada na tela de um computador, o processo que ocorre é o processamento dos dados contidos no arquivo digital que armazena essa imagem de forma que ela se atualiza na tela do computador. Com essa atualização, o observador da imagem tem virtualmente a sensação de estar presente no momento em que foi tirada a foto. E ainda é lhe permitido realizar outras atualizações como aproximar a imagem, aplicar efeitos gráficos, recortar a imagem, etc. Cada uma dessas ações é uma atualização da imagem, contudo a foto continua a existir virtualmente, desprendida de um tempo e de um espaço, mesmo que a foto se refira a um momento específico do tempo.
ii. No sentido do dispositivo informacional: A mensagem é um espaço de interação por proximidade dentro do qual o explorador pode controlar diretamente um representante de si mesmo, um "avatar". Um mundo virtual nesse sentido difere do primeiro ponto por haver uma representação efetiva do explorador/observador. A pessoa passa a se situar dentro de um mundo, através de seu avatar. Possivelmente, mas não necessariamente, há também representações de outros exploradores dentro desse mundo. Exemplos desses mundos virtuais são: mapas, tais

como Google Maps[9], onde a pessoa é representada no mapa através de sua posição atual; os jogos, em que o jogador controla um personagem; e simuladores de voo.

iii. No sentido tecnológico estrito: Ilusão de sensação sensório-motora com um modelo computacional. Trata-se da chamada "realidade virtual", um tipo particular de simulação interativa, na qual o explorador tem a sensação física de estar imerso na situação definida por um banco de dados. Sem, contudo, confundir com a realidade cotidiana. O efeito de imersão sensorial é obtido, em geral, pelo uso de um capacete especial e de datagloves, reproduzindo o mais fielmente possível as sensações sensoriais de um ambiente real, e interagindo com o ambiente de forma mais intuitiva possível.

No primeiro sentido, o explorador do mundo virtual não está inserido nele, é apenas um observador, mas manipula os objetos desse mundo. Esses mundos podem ser hipertextos, hiperdocumentos, simulações e, em geral, todos os objetos lógicos, tais como programas, bancos de dados e seus conteúdos, planilhas, programas para apoio à decisão, programas para pesquisa, etc.

Nos dois últimos sentidos de mundo virtual, se está tratando de ambientes virtuais. A palavra ambiente vem do latim *ambiens*, cercar, rodear. É aquilo que envolve, que está à volta de alguma coisa ou pessoa (PRIBERAM). Em ambos os casos o observador/explorador de certa forma está envolvido por algo, há virtualmente o conceito de lugar, de espaço, onde o explorador está inserido, ou sua representação virtual está presente nesse lugar. Através do conceito de proximidade, ele se desloca por esse ambiente.

Dessa forma, pode-se caracterizar **ambiente virtual** como um mundo virtual no qual o explorador tem a "sensação" de presença, um espaço de interação por proximidade dentro do qual o explorador pode controlar diretamente um representante de si mesmo.

[9] http://maps.google.com/

Neste trabalho se está especialmente interessado nos ambientes virtuais do segundo tipo, aqueles em que o explorador possui uma representação dele próprio dentro desse ambiente, porém não se trata de um ambiente de realidade virtual.

Menezes *et al.* (2008) argumentam que nesse tipo de ambiente:

> as propriedades dos objetos reais e virtuais são diferentes e essa pseudo-analogia [entre os objetos reais e virtuais] pode até mesmo acarretar em dificuldades na diferenciação entre o "comportamento" dos objetos e espaços materiais e virtuais.

Além disso, Lévy (1999, p. 17) conceitua um mundo virtual em particular, o ciberespaço:

> É o novo meio de comunicação que surge da interconexão mundial dos computadores. O termo especifica não apenas a infra-estrutura material da comunicação digital, mas também o universo oceânico de informações que ela abriga, assim como os seres humanos que navegam e alimentam esse universo.

O ciberespaço é de particular importância para o estudo realizado neste trabalho, principalmente pela Web, que compõe parte significativa do ciberespaço.

### 3.1.3 Interação Humano-Computador-Humano

Wagner (1994) define **interação** como: "eventos recíprocos que exigem pelo menos dois objetos e duas ações. Interações ocorrem quando esses objetos e eventos mutuamente influenciam um ao outro". Ressaltando que o termo interação pode ter significados mais específicos dependendo do contexto que está sendo usado, como por exemplo, o termo interação é usado em contextos e campos do saber como a medicina, engenharia, linguísticas e estatísticas (JENSEN, 1998).

No entanto, se está especialmente interessado na interação realizada entre as pessoas, através do uso de computadores conectados em rede, ou através do ciberespaço. Esse tipo de interação pode ser chamada de "Interação Mediada por Computador" (*Computer Mediated Interaction* - CMI) ou "Interação Humano-Computador-Humano" (*Human-to-Computer-to-Human Interaction* – HCHI), tendo os dois termos o mesmo sentido neste contexto.

Paredes (2007) define que interação humano-computador-humano ocorre quando:

> o computador deixa de ser um interveniente direto na interação [como é estudado na Interação Humano-Computador] passando a assumir um papel de simples mediador e/ou intermediário. Nesta mudança de paradigma, o objeto de estudo focaliza-se nos utilizadores, sendo dado particular destaque às questões sociais e à forma como os utilizadores interagem entre si.

Trata-se de uma interação mediada por computador: hardware, software e a interconexão em rede dando suporte a interações humano-humano (JONES e MARSH, 1997).

Thompson (1998, p. 78 *apud* EZEQUIEL, 2009) entende que interações mediadas "implicam o uso de um meio técnico que possibilita a transmissão de informação e conteúdo simbólico para indivíduos situados remotamente no espaço, no tempo, ou em ambos".

A partir de um olhar focado no que se passa entre os interagentes, no relacionamento estabelecido, quer-se propor dois tipos, Primo (2005) propõe dois grandes grupos de interação mediada por computador:

- Interação mútua: os interagentes reúnem-se em torno de contínuas problematizações. As soluções inventadas são apenas momentâneas, podendo participar de futuras problematizações. A própria relação entre os interagentes é um problema que motiva uma constante negociação. Cada ação expressa tem um impacto recursivo sobre a relação e sobre o comportamento dos interagentes. Ou seja, a interação não é mera somatória de ações individuais. Como exemplo pode-se citar um debate na sala em um fórum de um ambiente de educação a distância.
- Interação reativa: Diferentemente das interações mútuas, as reativas precisam estabelecer-se segundo determinam as condições iniciais (relações potenciais de estímulo-resposta impostas por pelo menos um dos envolvidos na interação) – se forem ultrapassadas, o sistema interativo pode ser bruscamente interrompido. Por percorrerem trilhas previsíveis, uma mesma troca reativa pode ser repetida à exaustão (mesmo que os contextos tenham variado). Como exemplo, pode-se citar um jogo simples de perguntas e respostas, onde o jogador chega ao final do jogo se acertas todas as respostas.

Pelo conceito de virtual de Lévy (1996), interação humano-computador-humano ocorre de maneira virtual, sendo atualizada nas telas dos computadores cada vez que um dos interagentes interage com o computador com a finalidade de interagir com outro interagente.

### 3.1.4 Ambiente Virtual de Interação Humano-Computador-Humano

Assim, depois de ter os conceitos de "ambiente virtual" e de "interação humano-computador-humano" bem delineados, é possível expor o conceito de "ambiente virtual de interação humano-computador-humano": trata-se de um mundo virtual no qual o explorador desse mundo tem a "sensação" de presença, que possui um espaço de interação por proximidade dentro do qual ele pode controlar diretamente um representante de si mesmo. Ainda nesse espaço, o explorador pode se encontrar com outras pessoas e se engajar em uma interação com tais exploradores. Dessa forma, o principal objetivo desse ambiente é promover e dar sustentação a essas interações entre as pessoas. Tratam-se de interações virtuais que ocorrem através desses ambientes, estando assim dissociadas de um agora e um aqui específico, ocorrendo muitas vezes de forma assíncrona. Essas interações virtuais são atualizadas nas telas dos computadores dos exploradores, que apresentam uma representação desse mundo virtual, inclusive situando os outros exploradores. O explorador então realiza uma ação de interação, que por sua vez é virtualizada para dentro do ambiente, promovendo assim um ciclo recursivo de virtualização/atualização.

Para simplificar, no restante trabalho, se fará menção ao termo mais curto "ambiente virtual", quando se estiver tratando de ambiente virtual de interação humano-computador-humano.

## 3.2 Tecnologias de Ambiente Virtuais

Nesta seção, apresentam-se algumas tecnologias que estão sendo utilizadas para dar suporte à criação e navegação de ambientes virtuais. Em destaque estão os groupware e os ambientes virtuais de aprendizagem. O objetivo é delinear o principal objeto de estudo deste trabalho: os ambientes virtuais web de interação humano-computador-humano.

Para tal são expostos os conceitos de groupware, ambientes virtuais de aprendizagem, relacionando-os com o contexto da Web, de *e-learning* e de sistemas de interação humano-computador-humano. Delineando assim, precisamente a área de interesse deste trabalho.

### 3.2.1 Groupware

Os groupware foram primeiro definidos como um processo intencional de grupos de pessoas mais o software para dar suporte a ele (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. Ellis e Wainer (2000) ainda definem groupware como sistemas baseados em tecnologias de computação e comunicação que ajudam grupos de participantes, e ajudam a dar suporte a um ambiente compartilhado.

Além disso, recentemente, surgiu o conceito de "software social": um software que suporta a interação em grupo. No entanto, Allen (2004) argumenta que essa definição é muito similar à definição de Johson-Lenz: um processo intencional de grupos de pessoas mais o software para dar suporte a ele. Dessa forma considera-se neste trabalho que o termo groupware e o termo software social são referentes à mesma coisa.

Allen (2004) ainda critica que o termo groupware tem sido muito usado em associação com "trabalho", pois foi muito popularizado pelas empresas como software que dá suporte ao trabalho em conjunto.

Dessa forma, o conceito adotado neste trabalho é de que **groupware é um processo intencional de grupos de pessoas mais o software para dar suporte a ele** (JOHNSON-LENZ e JOHNSON-LENZ, 1981 *apud* PENICHET, MARIN, *et al.*, 2007).

Várias ferramentas são consideradas groupware. Podem-se citar: fax, e-mail, sistema de telefone por IP, chat, gerenciador de documentos, fórum, mecanismos de aprovação, calendários de grupo, planejamento compartilhados, workflow, gerenciamento de eventos, agenda, quadro branco compartilhado, sistemas de notificação, etc. (PENICHET, *et al.*, 2007). No entanto, neste trabalho se está interessado naqueles groupware que possuem um ambiente (virtual) compartilhado e na Web. Esses groupware são ambientes virtuais que possuem interface na Web. Seus espaços e objetos são abstrações, e muitas vezes fazem analogias a espaços e objetos do mundo real. No entanto, são diferentes dos ambientes de realidade virtual. Assim, essa pseudo-analogia pode até mesmo acarretar em dificuldades na diferenciação entre o "comportamento" dos objetos e espaços materiais e virtuais (MENEZES, *et al.*, 2008).

### 3.2.2 Ambiente Virtual de Aprendizagem

Ambiente Virtual de Aprendizagem (AVA) é um groupware com objetivo de promover o ensino e a aprendizagem. As atividades realizadas em AVAs necessitam de interação entre as pessoas e um espaço compartilhado.

É um tipo de sistema de *e-Learning* (aprendizagem eletrônica), que é a aprendizagem promovida através de tecnologias digitais e computadores.

Watson e Watson (2007) classificam os AVAs como *Learning Management System* (LMS), *Course Management System* (CMS[10]) ou *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.

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).

LCMS é um sistema usado para criar, armazenar, montar e entregar conteúdo personalizado de *e-Learning* na forma de objetos de aprendizagem (OAKES, 2002 *apud* WATSON e WATSON, 2007).

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[11]. No entanto, o ambiente virtual de maior sucesso atualmente no Brasil e também no resto no mundo é o Moodle[12] (DOUGIAMAS e TAYLOR, 2003) (MAGALHÃES, *et al.*, 2010). 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

[10] CMS também pode se referir a *Content Management System*, que são sistemas diferentes dos *Course Management Systems*, tratam-se de sistemas que auxiliam na criação de websites de conteúdo.
[11] http://amadeus.cin.ufpe.br/
[12] http://moodle.org/

disponíveis no mercado está no uso de diferentes tecnologias que podem facilitar seu uso, customização e atualização.

Superficialmente, eles são apenas um conjunto ferramentas de comunicação/interação 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.

Grande parte desses AVAs são desenvolvidos com tecnologias Web.

Como foi visto no Capítulo 2, os usuários desses ambientes virtuais trazem uma demanda forte por flexibilidade nesses ambientes, tendo em vista o domínio de aplicação, o da educação. Dessa forma, se está particularmente interessado na de classe de ambientes virtuais dos AVAs.

### 3.2.3 Ambiente Virtual Web de Interação Humano-Computador-Humano

O interesse deste trabalho é em ambientes virtuais *web* de interação humano-computador-humano. Ou seja, sistemas com um ambiente compartilhado com o objetivo de promover a interação entre as pessoas através da Web. Seus principais representantes são groupware e AVAs baseados na Web, mas não se restringindo a esses. Para simplificar, neste trabalho se fará menção ao termo mais curto "ambiente virtual", quando se estiver tratando de ambiente virtual de interação humano-computador-humano.

Observa-se que ferramentas de comunicação e interação também estão incluídas nessa descrição. Ferramentas como Fórum, Blog, Chat, Wiki, Enquete, quando

baseados na Web, são tidos como ambientes virtuais de interação humano-computador-humano.

Na Figura 2, apresenta-se uma representação em diagrama de Venn dos conceitos abordados nesta seção. A área hachurada representa os ambientes virtuais de interação humano-computador-humano. Observa-se assim a forte relação desses ambientes com os groupware e AVAs baseados na Web. No entanto, existem outros sistemas de interação humano-computador-humano que não são groupware nem AVAs, como por exemplo, os blogs e chats.

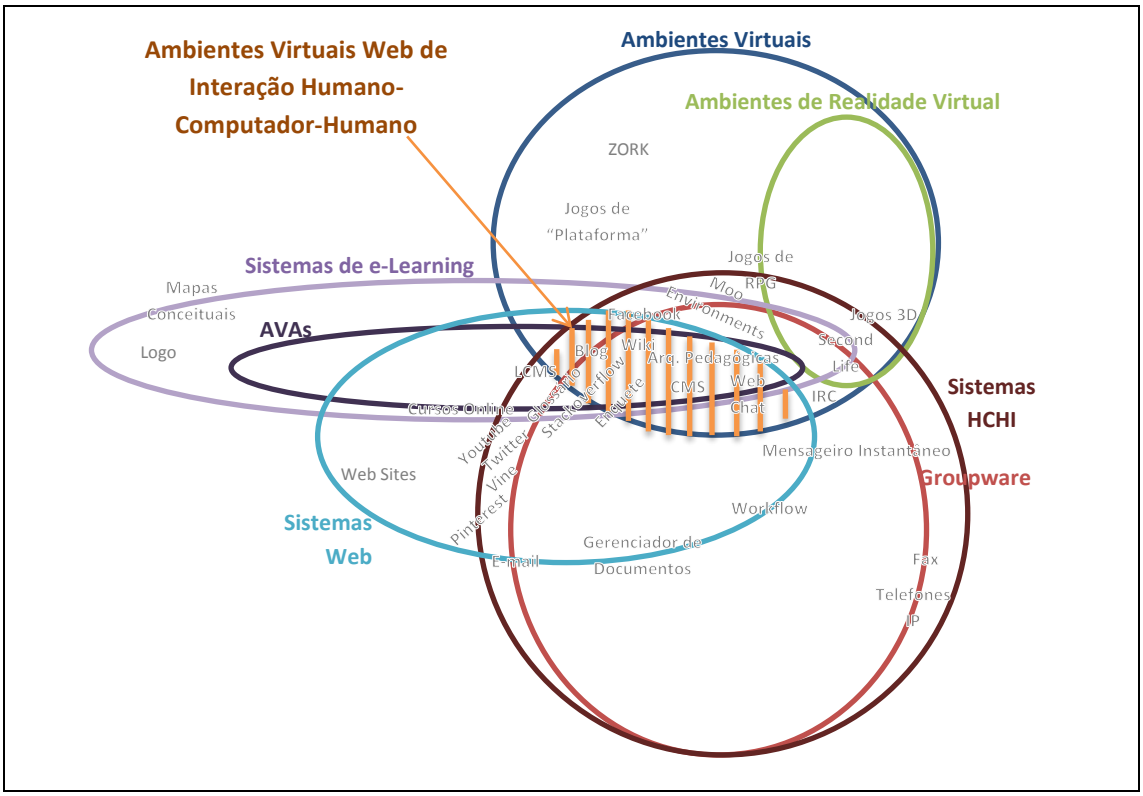


**Figura 2. Diagrama de Venn representando os conceitos relacionados com os Ambientes Virtuais de Interação Humano-Computador-Humano.**

Na Figura 2, também são situadas algumas ferramentas que são caracterizadas com os conceitos discutidos. Alguns deles não são de interesse direto deste trabalho, mas são listados para situar as definições:

- Zork: é um jogo de aventura, perigo e astúcia com interface e comandos baseados em texto, lançado em 1980[13]. É um exemplo de ambiente virtual que não produz uma realidade virtual, pois apresenta apenas

[13] http://pot.home.xs4all.nl/infocom/zork1.html

descrições textuais do ambiente. Os comandos que o usuário pode realizar são textuais, do tipo "go left" (ir à esquerda), "open door" (abrir a porta), "look around" (olhar em volta), e as respostas do sistema são descrições textuais do ambiente em volta, de sua localização, e dos efeitos das ações realizadas.

- Jogos de Plataforma: onde o jogador controla um personagem dentro jogo, saltando entre "plataformas". Por exemplo, o clássico jogo Mario Bros., dos anos 80, onde o jogador controla um encanador italino chamado Mario que, junto com seu irmão Luigi, corre por entre plataformas e tubos, quebrando tijolos, derrotando seus inimigos, que são tartarugas, cogumelos e balas de canhão, com o objetivo de salvar a princesa Peach.
- MOO Environments: são ambientes virtuais baseados em textos para uso por várias pessoas, possuindo "salas" e interação através de texto (DILLENBOURG, MENDELSOHN e JERMANN, 1999).
- Jogos de Realidade Virtual: vários são os jogos que possuem interface de realidade virtual, com gráficos em três dimensões, simulando a realidade e a física do mundo real.
- Jogos de RPG: são jogos onde o jogador controla (e interpreta) um personagem (ou um conjunto de personagens) e o encaminha por entre o enredo de uma história, juntando itens de valor e armas, e geralmente possuindo um sistema de comércio interno. A interface de tais jogos podem ser simples personagens representados em duas dimensões, como o jogo The Legend of Zelda, até com gráficos em três dimensões e jogabilidade avançada como os jogos da série Grand Theft Auto (que por muitos não é classificado como RPG, mas que se encaixa na definição).
- Second Life: é um ambiente virtual de três dimensões que simula a realidade, onde o usuário possui um avatar que o representa naquele mundo[14]. Vários trabalhos o citam como ferramenta para aprendizagem cooperativa.
- IRC: é um protocolo de comunicação de chat (síncrono) por texto muito popular nos anos 90.
- Mapas Conceituais: são diagramas que relacionam palavras/conceitos através de linhas e palavras de ligação, como verbos, formando frases. CmapTools é um software destinado a se desenvolver mapas

[14] http://secondlife.com/

conceituais[15], mas também podem ser confeccionados utilizando-se apenas lápis e papel.

- Logo: é uma linguagem de programação de gráficos que possui uma tartaruga como seu cursor de desenho. Muitos trabalhos foram publicados utilizando-se dessa linguagem como forma de ensino de computação e programação.
- Web Sites Tradicionais: basicamente páginas com informações gerais sobre algo ou empresa.
- Telefones IP: o uso da rede de internet para comunicação por voz entre as pessoas tem sido muito usado para comunicação entre duas ou mais pessoas, tanto num contexto interno de uma empresa quanto no contexto mundial.
- Fax: o envio de faxes pelo computador pode viabilizar a comunicação e interação entre as pessoas ou um grupo de pessoas.
- E-mail: sistema de comunicação assíncrona baseado em servidores de mensagens e caixas de mensagem pessoais. Usado para troca de mensagens entre duas ou mais pessoas. Existem vários clientes de e-mail, como o Outlook da Microsoft[16], mas hoje os clientes web são muito populares, como o Gmail do Google[17].
- Gerenciadores de Documentos Compartilhados: Dropbox[18], GoogleDrive[19] e SkyDrive[20] são os representantes mais famosos atualmente nessa categoria de ferramentas. Trata-se de gerenciadores de arquivos com replicação em servidores em nuvem, e compartilhamento entre usuários. Provêm também interface web para que seus arquivos sejam acessados.
- Workflow: são ferramentas que organizam as tarefas e entregas de uma equipe de trabalho. É um exemplo de ferramenta desse tipo o Microsoft Project[21].
- Mensageiro Instantâneo: permitem a comunicação síncrona entre as pessoas que utilizam um mesmo tipo de mensageiro. São exemplo de

[15] http://cmap.ihmc.us/
[16] http://office.microsoft.com/pt-br/outlook/
[17] http://mail.google.com/
[18] http://www.dropbox.com/
[19] http://drive.google.com/
[20] http://windows.microsoft.com/pt-br/skydrive/
[21] http://www.microsoft.com/project/

ferramentas desse tipo o Skype[22], o Microsoft Messenger[23], o GTalk e Hangout do Google[24].

- Youtube: um grande repositório de vídeos que podem ser assistidos online[25].
- Twitter: uma rede social onde qualquer um expressa suas ideias com textos pequenos e "seguem" as publicações de outros usuários[26].
- Vine: uma rede social muito parecida com o Twitter, porém em vez de textos, são publicados vídeos de no máximo dez segundos[27].
- Pinterest: uma rede social onde os usuários colecionam coisas e links que acham interessantes e "seguem" as coleções de outros usuários[28].
- Cursos Online: são cursos onde os estudantes têm acesso a matérias online, e alguns deles permitem interação com o tutor/professor. Um exemplo de site que contém esse tipo de curso é o Coursera[29].
- LCMS: um sistema para criar, armazenar, montar e entregar conteúdo de e-Learning na forma de objetos de aprendizagem (ver Seção 3.2.2).

Dessa forma, é possível caracterizar alguns representantes dos Ambientes Virtuais de Interação Humano-Computador-Humano: Ambientes de Apoio às Arquiteturas Pedagógicas, CMS's, Blogs, Fórums, Wikis, Glossários Colaborativos, Web Chats, o Facebook[30] e o Stackoverflow[31].

No restante deste trabalho, os ambientes virtuais de interação humano-computador-humano serão tratados apenas como "**Ambientes Virtuais"**, para facilitar e diminuir a menção que é feita a esse tipo de ambiente.

[22] http://www.skype.com/
[23] http://messenger.live.com/
[24] http://www.google.com/intl/pt-BR/+/learnmore/hangouts/
[25] http://www.youtube.com/
[26] http://www.twitter.com/
[27] http://vine.co/
[28] http://www.pinterest.com/
[29] http://www.coursera.org/
[30] http://facebook.com/
[31] http://stackoverflow.com/

## 3.3 Categorias de Groupware: Guardião, Coordenador, Comunicador

Existem diversas taxonomias de groupware, mas uma delas serviu de base conceitual para este trabalho. Ellis (1998), e também em conjunto com Wainer (ELLIS e WAINER, 2000), apresentou uma categorização funcional para groupware, dividindo-os em três categorias (não disjuntas): Coordenador, Comunicador e o Guardião do Artefato. O maior interesse deste trabalho é com relação ao conceito do Guardião do Artefato, que é uma dos principais ideais por trás do MOrFEu, com o diferencial de que o MOrFEu é uma espécie de "Guardião das Interações".

Essa taxonomia de Ellis é uma taxonomia funcional, que classifica groupware de acordo com a função genérica que o sistema provê.

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** (*The Keeper of the Artifact*) 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 a colaboração ocorre quando cada participante do grupo executa alguma atividade, em uma ordem previamente definida. **O Coordenador** (The Coodinator) 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 coordenador 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** (The Communicator) 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 deixa 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.

Na seção a seguir é explorado mais a fundo o aspecto do Guardião do Artefato. Neste trabalho, os groupware pertinentes aos aspectos de Coordenador e Comunicador são tratados como Guardiões, por isso a maior ênfase.

## 3.4 O Guardião de Artefato

Existem groupware que agem basicamente como repositórios e gerentes de dados compartilhados entre seus usuários. Ou seja, gerenciam os objetos que são manipulados pelo grupo e têm controle sobre as operações e atuando sobre esses objetos compartilhados. A semântica e a maneira como esses objetos devem ser

usados também é dada pelo groupware. Esse tipo de groupware foi chamado de Guardião do Artefato, por Ellis (1998).

Suponha a escrita de um documento em conjunto, por exemplo com o Google Docs[32]: um artigo sendo escrito por estudantes de doutorado e seus orientadores para ser publicado em alguma conferência; ou um contrato que está sendo redigido por executivos e advogados de uma empresa. O principal meio de interação desses participantes é o próprio documento que está sendo editado, eles aditam, marcam e grifam, colocam notas em alguns pontos do texto, etc. Ou seja, a interação proporcionada pelo groupware é centrada no acesso ao documento, ao artefato compartilhado, um espaço de dados.

Associado a um guardião há um modelo de objeto (artefato), implícito ou explícito. Esse modelo é a descrição dos repositórios, os itens de informação (ou tipos) e as operações sobre esses itens que o sistema fornece. É possível enxergar um artefato como um banco de dados de objetos e o guardião como um gerente desses objetos.

Guardiões normalmente estão preocupados com o acesso e o compartilhamento de dados no **espaço de dados**. Cada participante numa interação em um groupware pode ter múltiplas **esferas** de acessos de dados dentro do espaço de dados. Uma esfera engloba os pontos de dados que o participante pode ler; outras esferas dever englobar os pontos de dados que o participante pode escrever. A sobreposição de esferas de diferentes participantes indica possibilidades de compartilhamento.

Voltando ao exemplo do editor de documento compartilhado. Editores diferentes têm diferentes classes de **objetos** (por exemplo, parágrafos de texto, diagramas, páginas, linhas ou palavras), e diferentes **operações** (por exemplo, ler, anexar, inserir, substituir, copiar, colar ou deletar).

[32] http://docs.google.com

O Guardião ainda é responsável pela gerência de acesso simultâneo ao artefato. Há groupware que permitem acesso simultâneo de mais de um usuário ao mesmo tempo para alterar o artefato e outros que não permitem esse acesso.

Um exemplo de um Guardião fora do contexto eletrônico é um quadro branco. Os participantes de uma sessão de interação que possui um quadro branco podem utilizar esse quadro de maneira simultânea, ou seja, mais de um participante pode usar o quadro ao mesmo tempo.

Outro exemplo fora do contexto eletrônico é a redação de um contrato. Um executivo redige uma versão inicial do contrato, que é passado para outros executivos e advogados, que fazem marcações e anotações nas margens das páginas do contrato, um de cada vez. Voltando para o redator inicial que é responsável pelas alterações sugeridas.

O MOrFEu é um modelo de desenvolvimento de ambientes virtuais que se baseia na categoria do **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.

Isso vai de acordo com necessidades levantadas no Capítulo 2, apresentados nos requisitos R1 (Os usuários devem ser capazes de construir suas próprias ferramentas) e R2 (Os usuários devem ser capazes de fazer alterações nas aplicações conforme sua necessidade e em tempo de execução).

Essa é exatamente a ideia central deste trabalho**, desenvolver aplicações Guardiães de Artefatos especificáveis, partindo da ideia do MOrFEu, visualizando as aplicações Coordenadoras e Comunicadoras também como Guardiães**. Ou seja, com um modelo 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). É possível existir uma evolução dessas ferramentas, impulsionada por essas comparações e facilidades no seu desenvolvimento (requisitos R1 e R3).

## 3.5 O Agente de Time

Ellis (1998) ainda fala de uma quarta categoria de groupware, o Agente de Time (Equipe). No entanto, pela sua descrição, o Agente de Time é mais um serviço para groupware que um groupware por si só. Dessa forma, é apresentada nesta seção a descrição separada dessa "categoria" de groupware.

"Agentes [de time] executam funções especializadas dentro de um cenário de grupo" (ELLIS, 1998). Ou seja, os agentes realizam tarefas dentro do contexto de um groupware que não são primariamente relacionadas com a interação do grupo. São "módulos" de um groupware construídos para executar sub-tarefas específicas, e não globais. Eles frequentemente envolvem conhecimento de domínio especializado.

Os agentes de time contribuem de forma útil para um grupo, tal como se fizessem efetivamente parte do grupo, auxiliando com sua função especializada e específica. Ou seja, são membros artificiais do grupo.

Ellis e Wainer (2000) introduzem quatro sub-categorias dos Agentes de Time: agentes autônomos, agente de usuário único e agentes de grupo. Agentes autônomos trabalham sozinhos em alguma sub-tarefa. Agentes de usuários únicos (por exemplo, agentes de interface de usuário) interagem com um único usuário, trabalhando para ele, dentro de um grupo. Agentes de grupo interagem e colaboram com vários membros do grupo como um verdadeiro colega, ou agem como um dos participantes do grupo.

É importante ressaltar que esses agentes não necessariamente são agentes "inteligentes", como foi bem colocado por Ellis e Wainer (2000): "Agente é um processo automático; ele não precisa ser 'inteligente' ou 'autônomo'". Russel e Norvig (2004, p. 33) corroboram esse pensamento, eles defendem que um agente é programa de computador que interage em certo ambiente (não necessariamente virtual ou na web). Um agente é capaz de perceber seu ambiente por meio de sensores e de agir sobre esse ambiente por intermédio de atuadores.

Algumas tarefas (não muito "inteligentes") que podem ser executadas por tais agentes são: contar votos de decisões em uma reunião; compilar software em uma ferramenta de workflow de desenvolvimento de software; imprimir uma carta de aprovação baseado em dados presentes num banco de dados; etc.

Alguns exemplos de tarefas mais "inteligentes" que esses agentes executam são: escolher certa metodologia e ferramenta para uma reunião, baseado num problema; planejar uma sequência de atividades para serem executadas, baseado nas metas que devem ser alcançadas; negociar marcação de compromissos em sua agenda e nas agendas de seus colegas; etc.

Assim, os agentes de time são especialmente importante na redução cognitiva dos participantes do grupo.

Além disso, esses agentes têm sido especialmente importantes no desenvolvimento de soluções que atendam ao requisito R5 (As máquinas devem ser capazes de analisar os dados contidos nessas aplicações). De forma que os agentes que

auxiliam grupos têm sido utilizados no auxílio ao entendimento do conteúdo produzido pelos utilizadores do grupo.

## 3.6 O Modelo dos Teatros Sociais

O Modelo dos Teatros Sociais por proposto por Paredes (2007) (PAREDES e MARTINS, 2010) e apresenta uma concepção flexível para regulação de atividades realizadas em ambientes virtuais, primariamente atividades do cotidiano "portadas" para ambientes virtuais. Além disso, esse modelo prevê que as regras de interação de um ambiente possam sofrer modificações em tempo de execução.

O autor do modelo afirma que os ambientes virtuais devem ser condicionados pela natureza evolutiva das pessoas. Assim, o principal objetivo de seu trabalho é definir um modelo que assegure a coordenação e regulação de interação social em ambientes virtuais de interação, de modo que a regulação de interação de cada ambiente deve ser adaptável dinamicamente, refletindo as necessidades de interação dos seus usuários e flexibilizando o ambiente de forma a garantir seu sucesso.

Esse modelo se baseia em metáforas do teatro para ser composto: representação, peça, atores, palco e roteiro:

> O teatro, em particular a representação, pode ser visto como um modelo de interação regulado, dado que os atores interagem num ambiente segundo um conjunto de normas definidas de forma a atingir um objetivo em comum: por em cena a peça de teatro. Assim, o modelo de interação do teatro é caracterizado pelos atores que desempenham um ou mais papéis em palco segundo regras de interação definidas no guião [roteiro] da peça teatral. [...] Os Teatros Sociais são resultado da aplicação do modelo de interação do teatro a situações do cotidiano, em ambiente virtuais de interação social.

Os palcos desses teatros são os chamados Espaços Sociais, os ambientes virtuais propriamente ditos.

Esse modelo possui quatro componentes definidos: papéis, fluxo de interação, regras e ações.

Os papéis definem a função que é desempenhada por um utilizador num Espaço Social. Eles não são simples etiquetas, que associadas aos usuários, definem entidade de interação dos Espaços Sociais. Ou seja, 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 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 (6) 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 sequências das ações a serem realizadas pelos atores. A organização dessas ações é efetuada com base no fluxo de interação que atua de similar a um *workflow*, em particular nas ferramentas de groupware, encadeando as ações a serem desempenhadas por cada ator. Assim, o fluxo garante a organização e coordenação das ações de interação dos Espaços Sociais.

Um fluxo de interação é definido por: (1) um conjunto de ações; (2) um conjunto de transações, guardadas por condições; e (3) um conjunto opcional de fluxo de dados.

As regras são protocolos que garantem a regulação nos Espaços Virtuais. Uma regra é genericamente definida como uma expressão lógica que combina informação sobre a estrutura do Espaço Virtual e estado de execução, definindo se uma ação pode ser executada num estado. Além disso, as regras dos Espaços Sociais são adaptáveis em tempo de execução.

O quarto componente do Modelo dos Teatros Sociais são as Ações. Contudo, Paredes não as coloca explicitamente como um componente do modelo. O modelo, na verdade, age sobre essas ações. Elas são ações previamente definidas no Espaço Social, com o objetivo de concretizar a interação entre os participantes do ambiente.

As ações de interação são atividades que tem por objetivo promover serviços de conversação, compartilhamento de objetos e colaboração. Paredes dá um exemplo do cotidiano, uma conferência de impressa:

> Nessa situação há primeiramente uma declaração, apresentando o motivo da conferência de imprensa, seguida de uma sessão de questões. Podemos, num ambiente simplificado, identificar três ações nesta situação: "fazer uma declaração"; "por uma questão"; "responder à questão". [...] O fluxo de interação organiza a execução das ações definindo transições entre elas.

Além de ações de interação, há ainda ações que atuam diretamente no fluxo de interação, com o objetivo de poderem ser feitas alterações em tempo de execução à regulação. Para ambos os tipos de ações, há regras relacionadas.

Paredes ainda propõe uma arquitetura que provê um meio de implementar tal modelo em ambientes virtuais. Naquela arquitetura, há definições precisas de como especificar cada um dos componentes acima citados, através de configurações feitas com arquivos XML, que possuem campos específicos denotando o funcionamento desejado para cada um dos componentes.

Nota-se que as ações precisam estar previamente definidas para que os componentes de um Espaço Social regulem o ambiente. Paredes apenas trata da flexibilidade da regulação e execução de ações previamente estabelecidas. Não trata de flexibilidade para alteração dessas ações. Assim, um ambiente virtual flexível precisa de flexibilidade tanto em questões de regulação, quanto em questões estruturais e funcionais.

## 3.7 Conclusão do Capítulo

Neste capítulo, foram vistos os principais conceitos e ideias de sustentação deste trabalho.

Foi apresentado o conceito de "ambiente virtual" adotado neste trabalho. Os ambientes virtuais de interesse deste trabalho são sistemas web com um ambiente compartilhado com o objetivo de promover a interação entre as pessoas através do

computador. Os principais representantes dessa classe de aplicações são groupware e ambientes virtuais de aprendizagem baseados na Web, mas não se restringindo somente a esses. Foi visto no capítulo anterior a necessidade por flexibilidade nesse tipo de software.

Além disso, foi exposta uma ideia apresentada por Ellis (1998) que serve de base para a proposta do MOrFEu eXtendido, o conceito de "O Guardião do Artefato". Ellis aponta outras duas categorias de groupware: o Coordenador e o Comunicador. Essas categorias não são mutuamente exclusivas, mas classificam groupware segundo suas funcionalidades.

O maior interesse deste trabalho está na categoria do Guardião do Artefato. Esses groupware agem basicamente como repositórios e gerentes de dados compartilhados entre seus usuários, gerenciam os objetos que são manipulados pelo grupo e têm controle sobre as operações e os objetos compartilhados.

Ellis cita ainda uma quarta categoria de groupware, os Agentes de Times. Esses agentes realizam tarefas dentro do contexto de um groupware que não são primariamente relacionadas com a interação do grupo. É importante ressaltar que essa categoria não se aplica à ambientes virtuais, porém as outras três categorias sim (Guardião, Coordenador e Comunicador).

Dessa forma, neste capítulo, foram expostos os conceitos necessários em que foram embasados as ideias desta tese.

No próximo capítulo, são discutidos trabalhos que sob diferentes ângulos também tratam da questão da flexibilidade no desenvolvimento de groupware.

# 4 Abordagens de Desenvolvimento de Ambientes Virtuais Flexíveis

Neste capítulo, são discutidos alguns trabalhos que têm sido propostos para o desenvolvimento de ferramentas mais adequadas às necessidades específicas de cada grupo de usuário. Dentre elas, destacam-se abordagens baseadas em módulos, componentes, linhas de produto 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, 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, *et al.*, 2007).

Além dos ambientes virtuais citados nesse capítulo, existem uma diversidade de outros ambientes virtuais que não foram levados em consideração, pois não se propõem a ser ambientes virtuais flexíveis.

## 4.1 Módulos

Uma das principais técnicas que tem sido usada para o desenvolvimento de ambientes virtuais flexíveis é o desenvolvimento baseado em módulos, ou modular. Os ambientes virtuais desenvolvidos dessa maneira têm por objetivo a fácil integração

com novas funcionalidades apresentadas como módulos do sistema, de forma que seja possível a instalação fácil de novas funcionalidades.

Dentre os produtos comerciais de ambientes virtuais de aprendizagem, destaca-se o Moodle (DOUGIAMAS e TAYLOR, 2003), com mais de 80 mil sites registrados em cerca de 240 países, onde o Brasil é o terceiro país com maior número de sites (MOODLE.ORG, 2013). Esse ambiente virtual possui desenvolvimento modular, no modelo *open source*. Sua arquitetura modular permite que desenvolvedores criem módulos e adicionem funcionalidades. Esse é o principal objetivo dessa arquitetura, tornar o Moodle capaz de absorver novas funcionalidades e sempre evoluir para suprir as diversas necessidades de seus utilizadores.

No entanto, as modificações realizadas são válidas para todos os grupos e usuários do ambiente, ou seja, pequenos grupos não podem produzir ou alterar suas ferramentas sem que isso afete todos os usuários do ambiente.

O núcleo de serviços básicos do Moodle é indivisível, trata das funcionalidades de usuários, cursos e configurações dos módulos e do ambiente como um todo. As ferramentas de comunicação/interação são instaladas como módulos e incluídas nos cursos.

O Moodle possui o conceito de "curso". Um curso é um espaço virtual restrito para um grupo de usuários. Um ambiente virtual Moodle pode possuir vários cursos, onde cada usuário pode ser participante de um ou mais curso. Um curso é organizado e gerenciado por um "professor" (ou "tutor"). Esse usuário que possui esse papel especial num curso pode organizá-lo da maneira desejada. Outros participantes do curso são os "estudantes", utilizam as ferramentas disponibilizadas no curso pelo professor.

O professor organiza o curso através de tópicos, que também podem estar associados a semanas. Cada tópico pode conter uma lista de recursos e ferramentas.

Os recursos são conteúdos do curso, links para websites, páginas web estáticas, arquivos, etc. Tudo que o professor considerar importante para o curso, naquele tópico.

As ferramentas são destinadas à comunicação/interação entre os participantes do curso, por exemplo, fórum, chat, glossário, etc. Podem ser também enquetes, questionários de prova, entrega de tarefas, etc., ferramentas auxiliares num curso, além da interação entre os alunos e professor.

Associado a um curso, há ainda a funcionalidade de notas em atividades, onde determinadas ferramentas dão suporte a essa funcionalidade. Isso permite ao professor atribuir notas aos estudantes de acordo com a atividade realizada utilizando-se as ferramentas.

O núcleo do Moodle possui todas as funcionalidades de gerência de usuários e dos cursos, bem como a gerência de configurações também do ambiente como um todo. Além disso, possui um framework para instalação e execução dos módulos. O pacote básico de *download* do Moodle possui vários dessas ferramentas como padrão, porém outras mais podem ser encontradas no site do Moodle.

Para se desenvolver um módulo do Moodle é necessário se adequar ao seu framework. O trabalho de Santos (SANTOS, 2009) traz um exemplo de como utilizar o framework do Moodle para se construir um sub-ambiente virtual que se aproveita de toda a sua infraestrutura de serviços. Não é uma tarefa muito complicada, porém o desenvolvimento de uma ferramenta deve seguir o framework, de acordo com as funcionalidades e organização do Moodle.

Para se criar novos módulos ou se fazer alterações em módulos já existentes é necessário acesso ao código-fonte e privilégios como administrador do sistema.

Uma vez instalado, um módulo vale para todo o ambiente. Se alguma modificação for realizada nesse módulo, ela será válida para todo o ambiente, em todos os cursos. Ou seja, grupos pequenos não podem produzir ou alterar suas ferramentas sem que isso afete todos os usuários do ambiente.

Dessa forma, os ambientes baseados em desenvolvimento modular não são ambientes virtuais com grande flexibilidade, que embora presente na instalação de novos módulos, não consegue atender à demanda dos usuários que necessitam de ferramentas personalizadas às suas demandas.

## 4.2 Componentes e Linhas de Produto de Software

A técnica de desenvolvimento com componentes também é estudada para desenvolvimento de ambientes virtuais flexíveis. O princípio que norteia esses trabalhos é a montagem (*assembly*) de componentes pré-desenvolvidos de acordo com as necessidades de cada grupo em particular, gerando assim um ambiente virtual para suporte das atividades daquele grupo. No entanto, essa técnica depende de componentes pré-desenvolvidos e configuráveis para serem usadas em diversas ocasiões. Além disso, as Linhas de Produto de Software é um método mais sistemático para se desenvolver groupware com componentes.

Vários são os trabalhos que utilizam desenvolvimento baseado em componentes para se construir groupware e ambientes virtuais. Dentre eles pode-se citar os trabalhos de Roseman e Greenberg (1996), Farias *et. al* (2000), Roth e Unger (2000), Won *et al.* (2005), Gerosa *et al.* (2006).

Roseman e Greenberg (1996) propuseram um *toolkit* para desenvolvimento de aplicações de conferência síncrona e distribuída, chamado GroupKit. Eles desenvolveram um núcleo de rotinas de computação comuns requisitadas por quase todos os *groupware* notadas até aquele momento.

Farias *et. al* (2000) propuseram uma metodologia para desenvolvimento de *groupware* através do reuso e montagem de componentes, artefatos de software pré-fabricados, configuráveis e evoluídos independentemente na forma de blocos de construção. De acordo com esse processo, o desenvolvimento de uma aplicação é organizado usando quatro diferentes camadas de abstração. Em cada nível diferente,

visões são utilizadas para capturar aspectos estruturais, comportacionais e interacionais da aplicação em desenvolvimento.

Roth e Unger (2000) propuseram uma abordagem de desenvolvimento baseado em componentes partindo de abordagens para o desenvolvimento de aplicações *single-user*, resolvendo problemas para a versão colaborativa: componentes colaborativos têm que se comunicar com outros sites e múltiplos usuários, têm que gerenciar dados compartilhados, têm que reagir a eventos do grupo e têm que oferecer serviços de percepção de grupo.

Won *et al.* (2005) propuseram uma abordagem de desenvolvimento com componentes que possibilitasse *tailoring*, flexibilidade de alterações em tempo de execução por parte dos usuários-finais. No entanto, ele se baseia na ideia de que a integração de componentes é facilmente entendida tanto por programadores quanto pelos usuários. Propõem, assim, uma arquitetura de software que permite *tailoring* de aplicações, contudo as modificações passíveis de serem feitas são em base de configurações e composições de componentes pré-fabricados.

Gerosa *et al.* (2006) propuseram uma forma de organização de do desenvolvimento com componentes baseada no modelo de colaboração 3C. Sua proposta baseia-se na composição, recomposição e personalização de componentes, refletindo nas dinâmicas de colaboração.

Além desses, Gadelha *et al.* (2009) utilizam um método mais sistemático para se desenvolver groupware com componentes, chamado *Groupware Product Lines* (Linhas de Produto de Software). Esse método cobre todas as etapas de desenvolvimento do groupware, desde o levantamento de requisitos, até a montagem final do produto.

Utilizando abordagens particulares e modelos inspirados em diferentes teorias, todos eles utilizam os conceitos de montagem de componentes. Tal abordagem pode ser comparada a montar peças de Lego® para formar um artefato desejado: um ambiente virtual para dar suporte adequado a um grupo específico.

O reuso dos componentes é estratégia aplicada nessas montagens. Partindo-se do princípio de que os groupware partilham de muitas funcionalidades em comum, pode-se criar uma variedade desses groupware reusando os principais componentes montando-os com outros componentes necessários para cada grupo específico. Dessa forma são encapsuladas muitas das dificuldades técnicas no desenvolvimento de groupware, tais como gestão de login, envio de e-mails, gestão de arquivos, etc.

A característica de montagem dos componentes traz a sua principal limitação: é preciso ter certa variedade de componentes para serem montados. Nos trabalhos citados anteriormente, os componentes são ferramentas de comunicação/interação, tais como um chat ou um fórum. A ideia é montar um groupware com essas ferramentas, de tal maneira que o ambiente virtual montado seja personalizado para o grupo que irá utilizá-lo.

Isso é semelhante aos módulos em ambientes virtuais baseados em módulos (apesar de ser melhor organizado), como foi descrito na seção anterior, se o ambiente virtual for usado apenas por um grupo de usuários. O ambiente virtual deve ser configurado com as ferramentas de comunicação/interação (módulos ou componentes) necessárias para àquele grupo. No entanto, a abordagem baseada em componentes é um pouco mais flexível no sentido que são possíveis modificações um pouco mais profundas, podendo ser escolhidos componentes adequados para outro nível de funcionalidades no ambiente virtual, além das ferramentas de comunicação/interação.

Contudo, esse tipo de desenvolvimento não dá suporte adequado para modificações posteriores à geração do produto. É necessário fazer modificações no componente que se deseja alterar, possivelmente modificando sua interface de ligação com os outros componentes do sistema. Dessa forma, as demandas de grupos particulares podem não ser atendidas prontamente, necessitando de uma reformulação do ambiente, talvez voltando a ponto inicial do projeto.

## 4.3 Web Services e Agentes

Outros trabalhos atacam o problema de criação de groupware personalizados através da composição de serviços distribuídos na Web( *Web services*). O princípio é o mesmo das abordagens baseadas em componentes: o reuso e a composição de aplicações já existentes, montados de forma personalizada para cada grupo particular. Tendo características parecidas com o desenvolvimento com componentes, essa abordagem necessita de serviços pré-desenvolvidos para serem usados nos groupware criados.

Esses *web services* podem ser aplicações *stand alone* existentes na Web, tais como serviços de chat, fórum, arquivos, etc., que seguem o modelo de *web services* padronizados, através de um protocolo de comunicação bem definido, ou seja,uma API que é possível acessar suas operações. Pode-se citar dois trabalhos que utilizam essa técnica de montagem para criação de groupware personalizados, os trabalhos de Pessoa *et al.* (2002) e Medeiros (2005).

Pessoa *et al.* (2002) propuseram um *framework*, chamado FAmCorA, para desenvolvimento de aplicações a partir de reuso de provedores de serviço (*web services*). Eles propõe um protocolo de comunicação inter-aplicações que permite chamar aplicações externas apenas apontando para um *link* e ainda assim manter o controle e monitoramento das ações do usuário.

Medeiros (2005) propôs um ambiente virtual baseado na metáfora de Estações de Aprendizagem, através da composição de *web services* para formar esses ambientes.

A principal diferença entre a abordagem de *web services* e a de componentes reside no que é reusado. Com *web services* são reusadas aplicações completas que existem por si só, na Web, e os componentes reusam pequenos blocos de construção, e é necessário montar tais blocos através de elementos de programação, enquanto com *web services* é possível apenas montá-los através de configurações.

Pessoa *et al.* (2002) e Medeiros (2005) utilizam agentes para prover a interface de comunicação entre os *web services*. Dessa forma, é possível prover uma camada de *web services* para aplicações que não foram originalmente criadas para esse propósito, ou seja, não possuem uma API padrão de comunicação inter-aplicações. Portanto, é possível existir aplicações *stand alone* que podem ser reusadas dentro do contexto de um groupware que agrega várias dessas ferramentas, de acordo com as necessidades específicas do grupo para o qual se está desenvolvendo.

Isso encapsula diversas dificuldades no desenvolvimento de aplicações de comunicação/interação utilizadas no contexto de ambientes virtuais. No entanto, ficam ainda mais difíceis as modificações em algum desses componentes, pois possivelmente o desenvolvedor do ambiente virtual não é o mesmo desenvolvedor da aplicação que se está reusando, podendo estar distribuída em outro servidor na Web. Além de que a gerência de dados em bases locais e remotas não é trivial.

Logo, as modificações em tempo de execução ficam praticamente inviáveis. É possível trocar o serviço por outro, no entanto, todo o contexto de utilização anteriormente desenvolvido será perdido. Ou seja, um serviço carrega consigo todos os dados gerados por ele, tornando-os acessíveis apenas através das interfaces de comunicação definidas.

## 4.4 Conclusão do Capítulo

Essas abordagens tratam o ambiente virtual como uma aplicação concluída, onde 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, como discutido no capítulo anterior. No entanto, pontos positivos podem ser extraídos dessas abordagens.

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, inclusive em tempo de execução. No entanto, a modificação de funcionalidades de ferramentas já existentes é um tanto complicada, o que inviabiliza a obtenção de um ambiente personalizado para um grupo de usuários.

Uma característica interessante do desenvolvimento com componentes e linhas de produto de software é o reuso de código, que permite o desenvolvimento mais rápido de outras ferramentas semelhantes às já existentes. Em tempo de projeto, permite que sejam feitas modificações no ambiente virtual de acordo com suas necessidades.

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 nas 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.

Na Tabela 2, é apresentado um resumo comparativo das características de cada abordagem apresentada. Observa-se que algumas delas compartilham características. Observam-se ainda suas limitações em se obter ambientes virtuais personalizados.

**Tabela 2. Resumo das características das abordagens de desenvolvimento de ambientes virtuais flexíveis.**

| | Módulos | Componentes | Webservices/Agentes |
|---|---|---|---|
| Encapsulamento de dificuldades técnicas | Sim | Sim | Sim |
| Inclusão de novas funcionalidades | Em tempo de Execução, ferramentas;<br>Não para funcionalidades do núcleo | Tempo de Projeto | Tempo de Execução |
| Modificação de funcionalidades | Não | Tempo de Projeto | Não |
| Ambientes personalizados | Não | Tempo de Projeto | Sim, havendo serviços disponíveis |

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**: nenhuma das abordagens citadas explora tal requisito;

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 buscam dar suporte às funcionalidades que atendam a esse requisito;

Dessa forma, observa-se que o requisito R2, em especial, não está sendo contemplado. Ou seja, nenhuma das abordagens de desenvolvimento ambientes virtuais flexíveis lida adequadamente com o problema de serem feitas modificações nos ambientes virtuais em tempo de execução para atender às demandas de seus usuários.

No próximo capítulo, apresentamos uma abordagem inovadora, concebida para atender demandas da cooperação através da web, características do requisito R2.

# 5 MOrFEu: uma abordagem diferenciada para ambientes virtuais

Neste capítulo, são discutidas as ideias do MOrFEu. São feitos estudos do potencial desse modelo através de provas de conceito e estudo de caso, que também são apresentados neste capítulo. O resultado é um panorama das limitações do MOrFEu e ideias de evolução do mesmo, culminando na contribuição central deste trabalho, o MOrFEu eXtendido (MX), apresentado no capítulo seguinte.

Menezes *et al.* (2008) propõem uma maneira diferenciada de se pensar ambientes virtuais, a essa proposta foi dado o nome de MOrFEu, um acrônimo para "Multi-Organizador Flexível para Espaços Virtuais". A busca por essa nova abordagem está pautada nos princípios de plasticidade, 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 dessa abordagem do MOrFEu para se criar uma maneira de se especificar ambientes virtuais**.

Além de apresentar as ideias de apresentadas em (MENEZES, *et al.*, 2008), é feita uma exploração de suas capacidades, passando pela concretização das ideias em artefatos, implementações de ambientes virtuais que foram construídos segundo aquela perspectiva. Além da concepção de uma linguagem diagramática para especificação de documentos.

## 5.1 MOrFEu: Ideia geral

"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 3.4 para maiores detalhes). Esse artefato é chamado, no MOrFEu, de **documento**.

Dessa forma, um ambiente 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. O 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 3, está representado o funcionamento de um ambiente virtual segundo modelo conceitual 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.

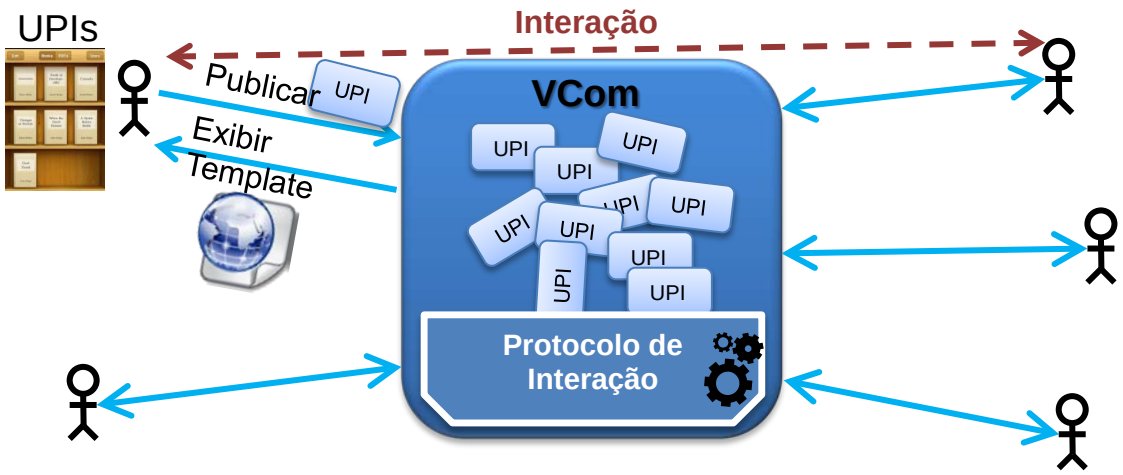


**Figura 3. Funcionamento de um VCom, dando suporte à interação de seus usuários.**

O MOrFEu não busca apenas a criação de ambientes inovadores construídos a partir de suas ideias, mas também é possível pensar os ambientes existentes atualmente da forma proposta.

O MOrFEu é um projeto multi-institucional que conta com pesquisadores das Universidades Federais do Amazonas (UFAM), do Espírito Santos (UFES) e Rio Grande do Sul (UFRGS). Há diversos trabalhos no âmbito do projeto, que exploram cada um de seus elementos e tentam desenvolver a ideia como um todo.

### 5.1.1 Autoria

Uma concepção relevante do modelo 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[33] 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.

No MOrFEu, a **Unidade de Produção Intelectual** (UPI), que é o artefato básico de suporte à autoria, destina-se a registrar as produções individuais. O tipo básico de uma UPI são textos escritos em HTML e, portanto, podem usar diferentes

[33] 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.

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.

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 3). Contrastando com a forma atual: quando uma pessoa utiliza uma ferramenta de comunicação/interação, sua produção fica atrelada a essa ferramenta (Figura 4).

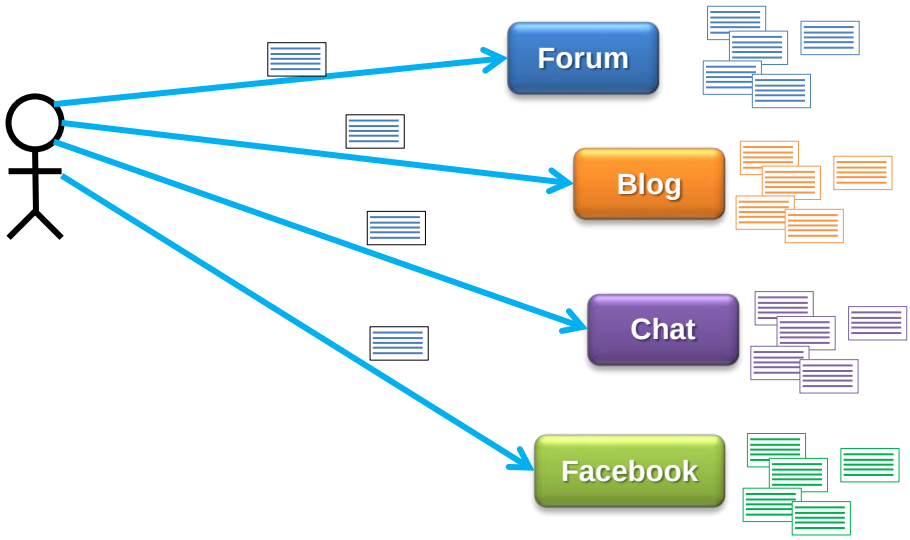


**Figura 4. Forma tradicional em que as produções são gerenciadas nos ambientes virtuais.**

### 5.1.2 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 colaboração. 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[34]) 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

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

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.

### 5.1.3 O que o MOrFEu não é

O MOrFEu é um modelo conceitual de ambientes virtuais para a Web, ele não trata de outras classes de ambientes virtuais, como os ambientes de 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.

## 5.2 Arquitetura de Desenvolvimento de VComs

A primeira ação de avaliação da proposta MOrFEu consistiu na concepção e desenvolvimento de um meta-ambiente virtual que gerencia outros ambientes virtuais nele contidos, os Veículos de Comunicação (VCom). Nesta seção, propomos uma arquitetura para VComs, de maneira que eles atendam aos requisitos de flexibilidade (requisito R2).

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, *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 espera-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.

Primeiramente, são expostos os conceitos gerais dos 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 5, 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.

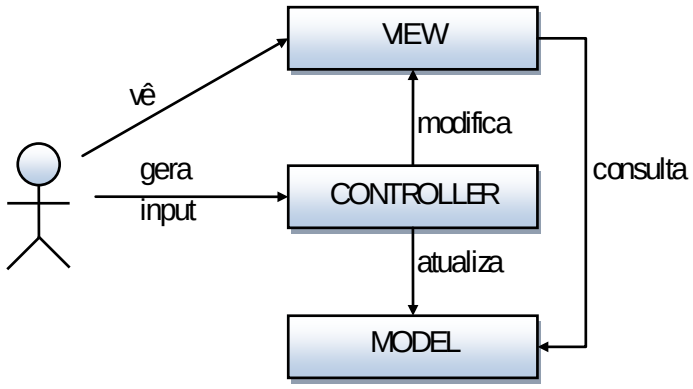


**Figura 5. Esquema do padrão de arquitetura MVC.**

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

### 5.2.1 Model

Um dos princípios do MOrFEu é a flexibilidade, 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[35] e Sedna[36]; e os baseados em documentos, como CouchDB[37], MongoDB[38] e SimpleDB[39].

[35] http://exist.sourceforge.net/
[36] http://www.sedna.org/
[37] http://couchdb.apache.org/
[38] http://www.mongodb.org/
[39] http://aws.amazon.com/simpledb/

### 5.2.2 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.

### 5.2.3 Controller

O controller é o 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, *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 de ações CRUDP (*Create, Read, Update, Delete and Publish*).

### 5.2.4 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.

### 5.2.5 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.

### 5.2.6 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 6, é apresentado um diagrama com o resumo das principais funcionalidades de cada um dos componentes da arquitetura e as relações entre eles.

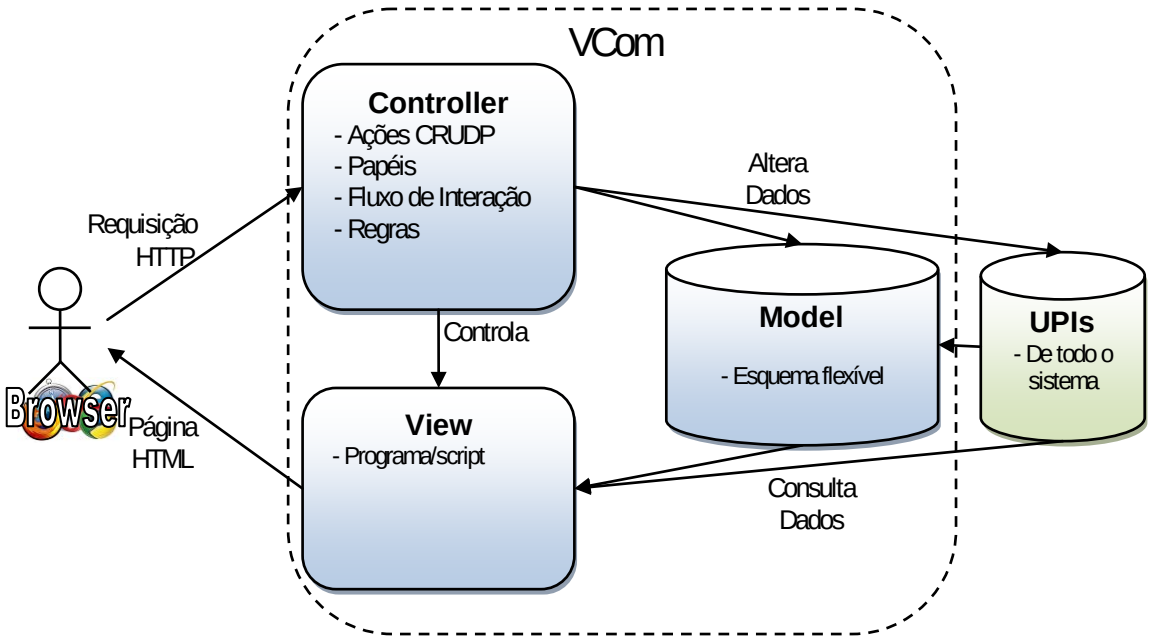


**Figura 6. Arquitetura MVC dos Veículos de Comunicação do MOrFEu.**

## 5.3 Meta-ambiente para Desenvolvimento de Veículos de Comunicação

Uma vez descrita a arquitetura, foi então 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 7, 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.

Diferente do que foi sugerido na seção que fala sobre o Model, o esquema de dados utilizado foi um tipo tradicional, implementado em um banco de dados relacional. Apesar de parecer contrário ao que foi dito anteriormente, trata-se da concepção do meta-ambiente virtual que não necessitará de modificações em suas funcionalidades. Esse meta-ambiente foi concebido para auxiliar no desenvolvimento e na execução dos ambientes virtuais. Esses últimos sim, foram concebidos para serem passíveis de modificações em tempo de execução.

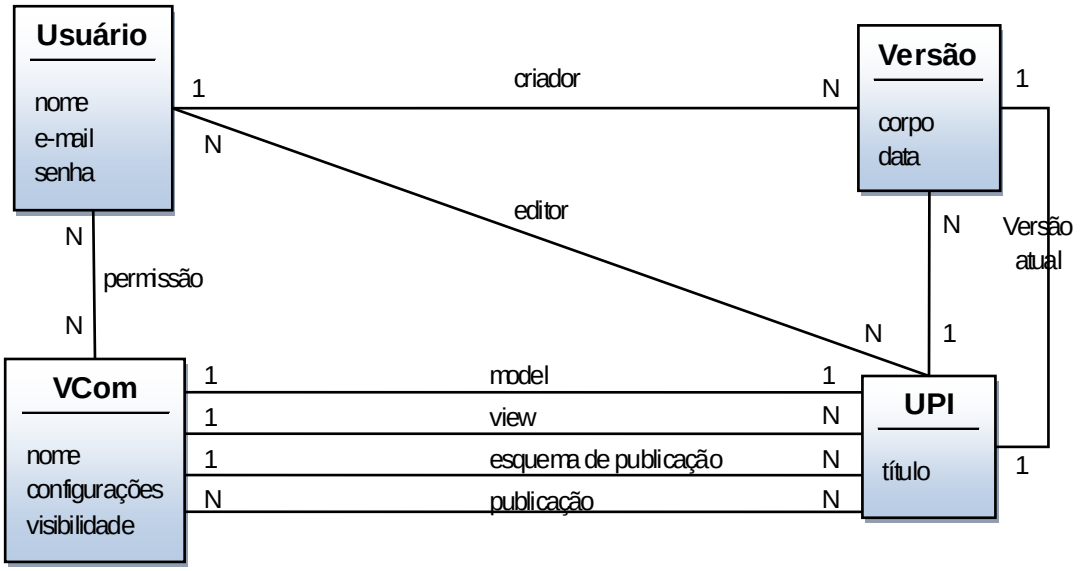


**Figura 7. Esquema de dados do protótipo do MOrFEu.**

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-fonte base 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 meta-ambiente virtual 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. Dentre as alternativas, 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 desses 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 estão apresentadas na Figura 8.

Não foram implementadas as funcionalidades relacionadas com o protocolo de interação: papéis, regras e fluxo de interação. No entanto, alguns VCom

implementados necessitavam dessas funcionalidades, e elas foram implementadas com controladores nas Views e armazenando dados no Model.

Os papéis básicos existentes são os autores e os leitores de um VCom. Os 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 9, 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 10. 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.

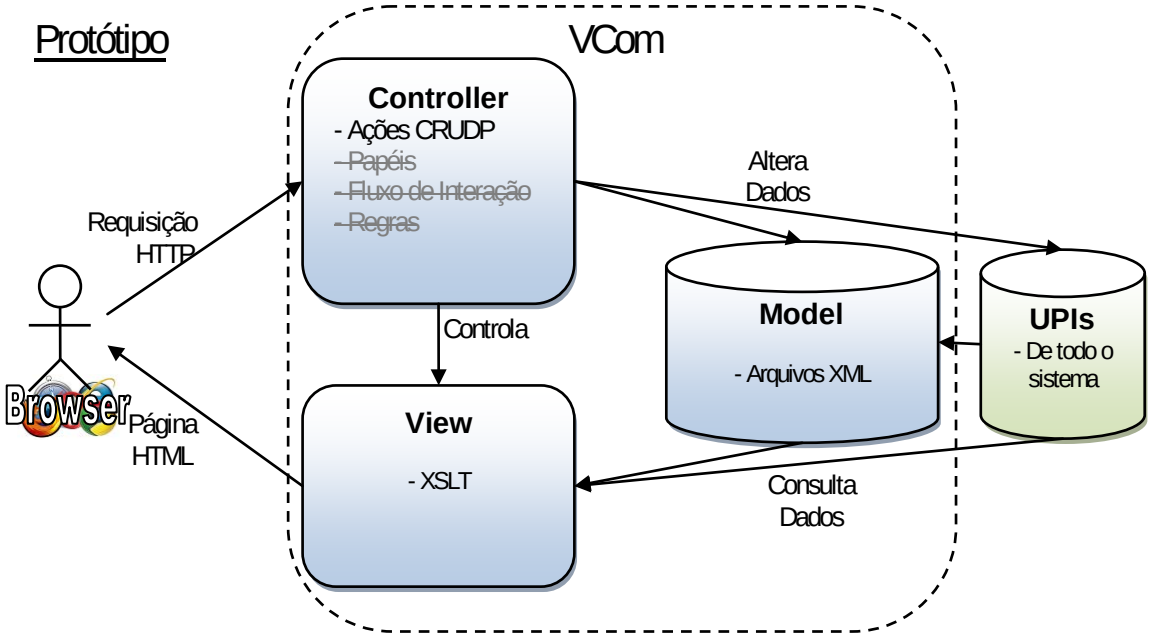


**Figura 8. Arquitetura MVC de Protótipo MOrFEu.**

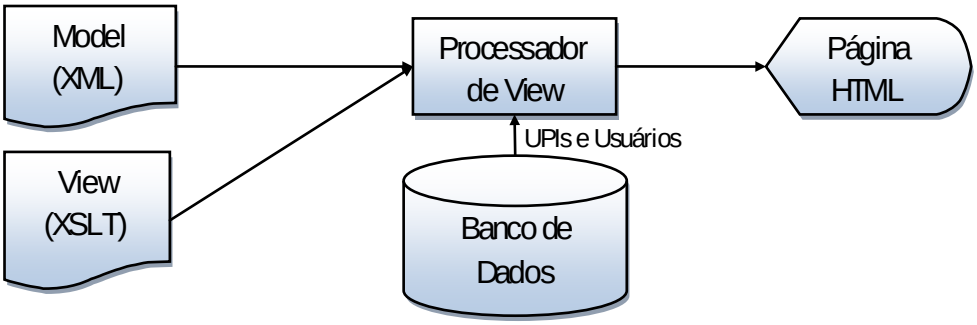


**Figura 9. Processamento de uma view.**

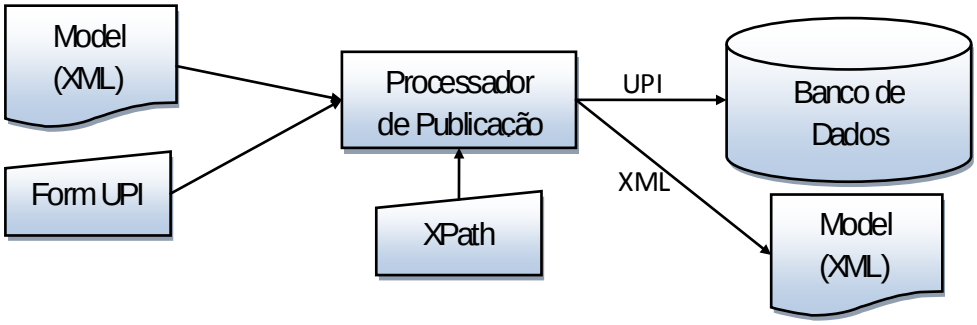


**Figura 10. Processamento de uma publicação.**

### 5.3.1 Ilustrando o Ciclo de Desenvolvimento de um VCom

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, num desenvolvimento em espiral.

Será utilizado como exemplo o desenvolvimento de uma ferramenta de apoio à arquitetura pedagógica "Debate de Teses" (NEVADO, DALPIAZ e MENEZES, 2009) (ver Seção 2.2.3). Esta ferramenta possui muitas das funcionalidades que esse protótipo MOrFEu disponibilizou. Exemplos de outras ferramentas serão mostrados para completar a descrição de suas funcionalidades.

#### 5.3.1.1 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, porque além de se poder criar 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 11, é apresentado um diagrama no estilo JSP e uma instância de documento XML equivalente ao diagrama.

Na Figura 12, é apresentado o diagrama em JSP do esquema de dados da ferramenta de apoio à arquitetura pedagógica Debate de Teses, 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.

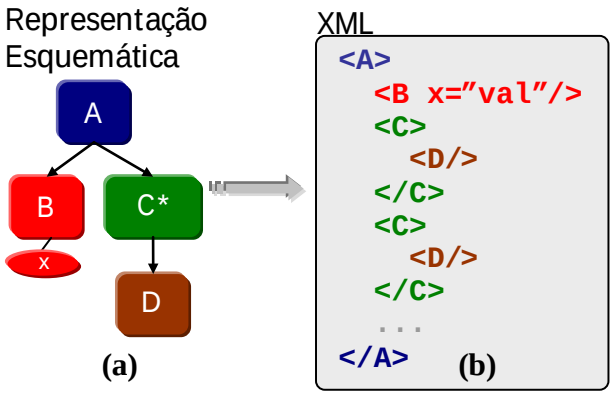


**Figura 11. (a) Representação Gráfica da Estrutura de um VCom. (b) Tradução da linguagem gráfica para XML.**

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 12, esse esquema é representado com linhas pontilhadas delimitando cada esquema de publicação.

No quadro de discussão (ver Figura 12), 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.

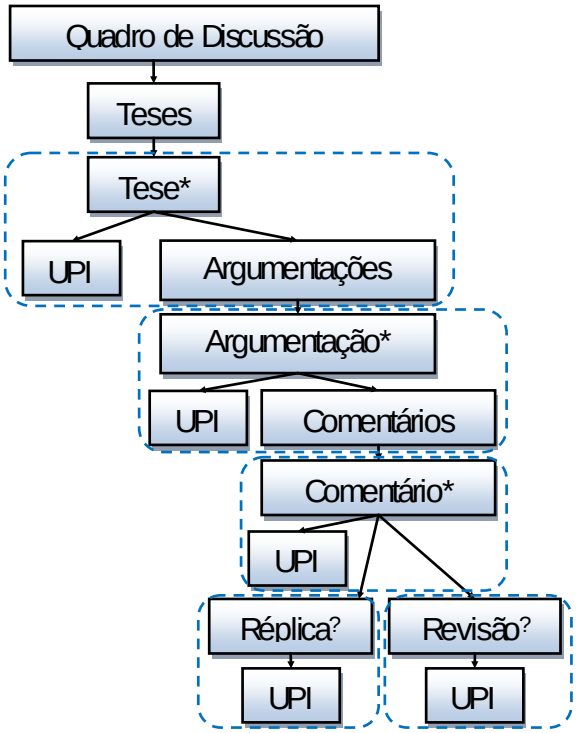


**Figura 12. Diagrama do esquema de dados da ferramenta de apoio à arquitetura pedagógica Debate de Teses usando diagrama JSP.**

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 13, 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 14).

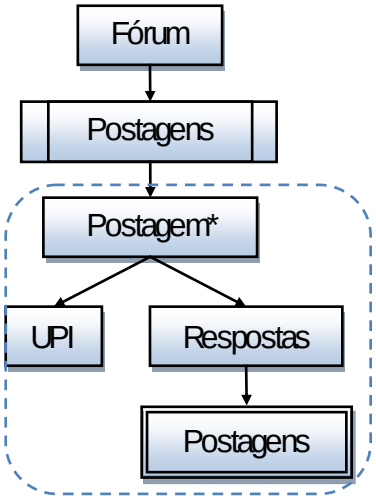


**Figura 13. Diagrama do esquema de dados de uma ferramenta de fórum.**

No VCom para apoio básico a projetos de aprendizagem (Figura 15), uma das ferramentas posteriormente implementadas, 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.

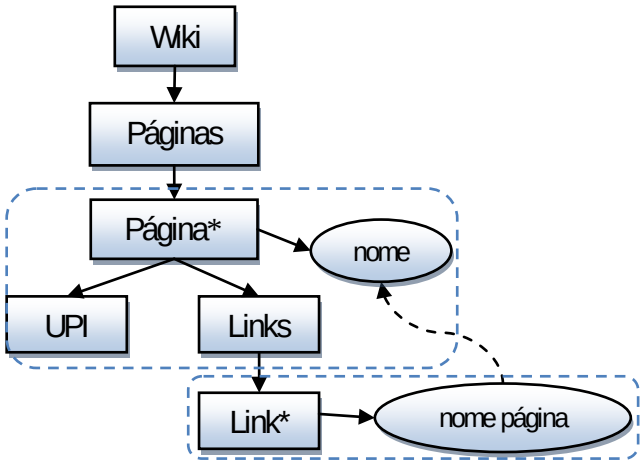


**Figura 14. Diagrama do esquema de dados de uma ferramenta de wiki.**

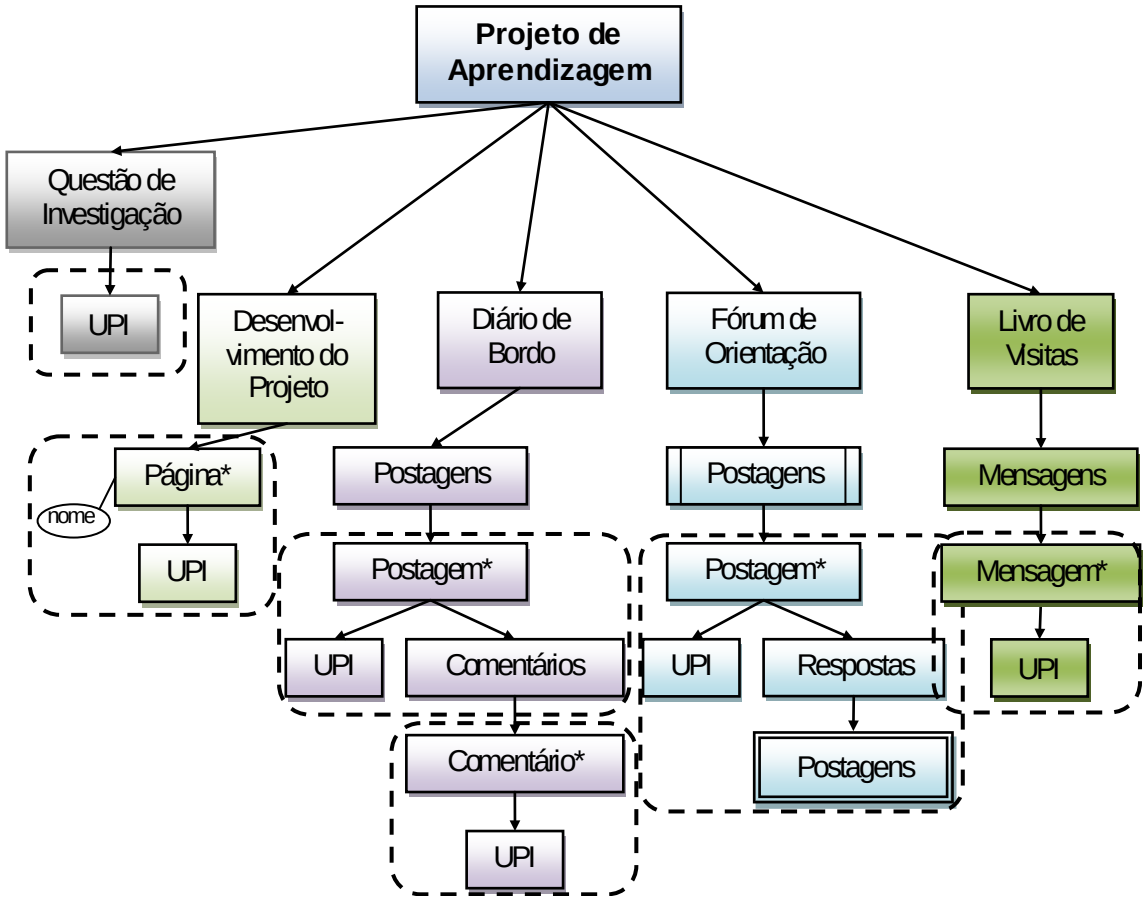


**Figura 15. Diagrama do esquema de dados de uma ferramenta de apoio a projetos de aprendizagem, destacando os sub-VComs.**

#### 5.3.1.2 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 16. Assim, sabe-se que é preciso criar três views, uma para cada página, nessa modelagem.

#### 5.3.1.3 Views

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

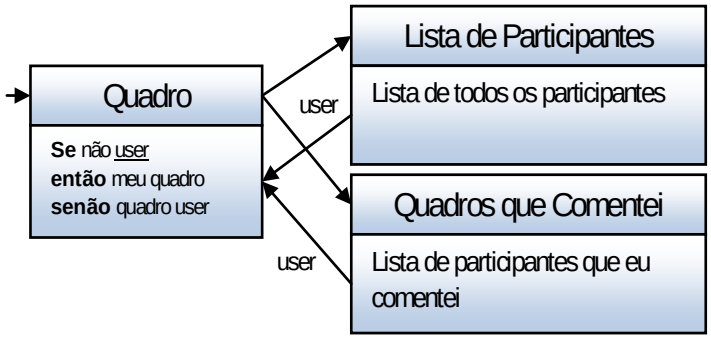


**Figura 16. Diagrama de navegação do quadro de discussão.**

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 gera-los, 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 15) 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.

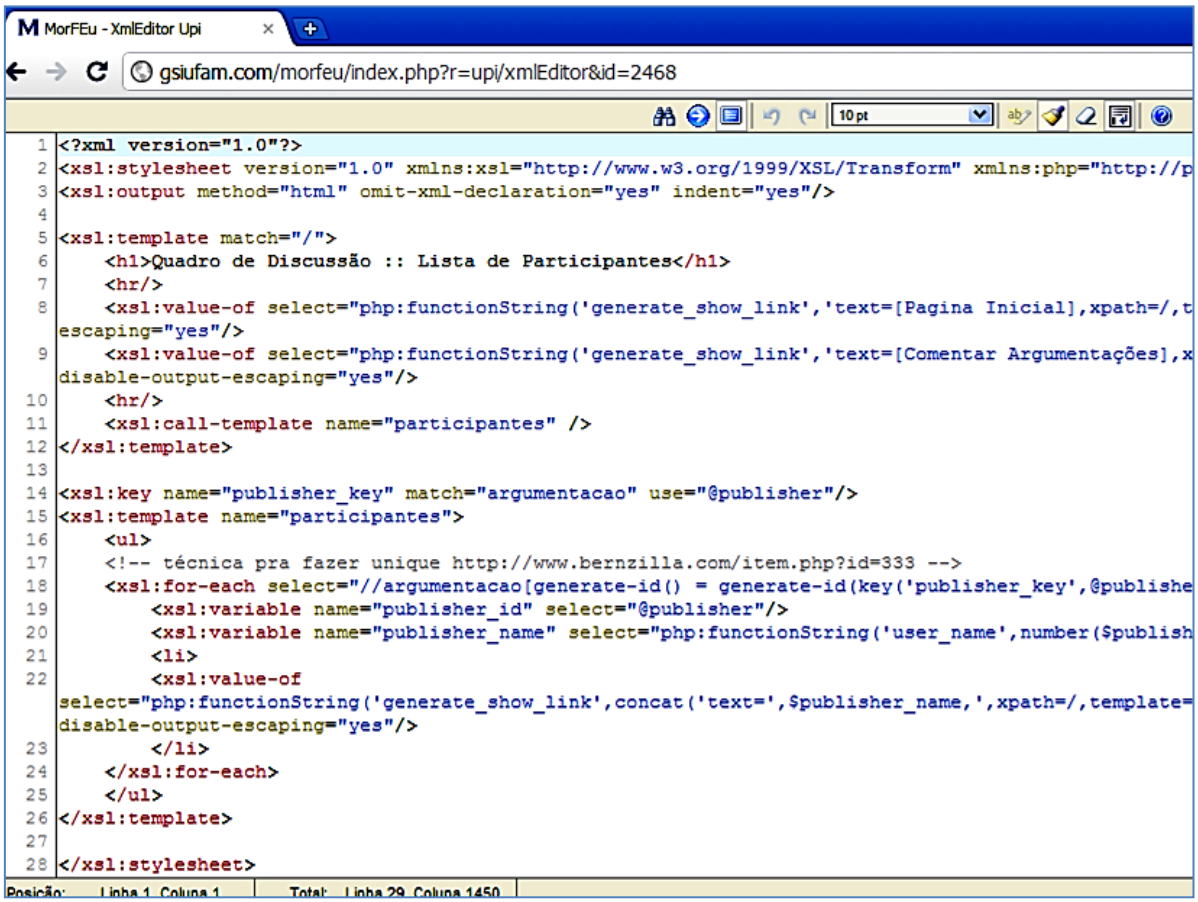


**Figura 17. View em XSLT da página Lista de Participantes do VCom.**

Na Figura 17, 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:

- 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.

#### 5.3.1.4 Alterando o VCom

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 Debate de Teses, 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 18, 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 19, 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, 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.

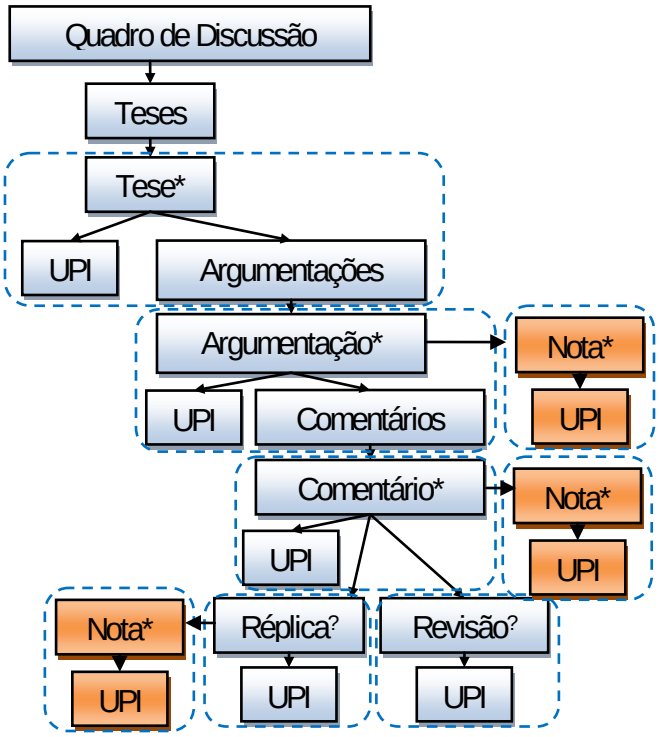


**Figura 18. Diagrama do esquema de dados da ferramenta de apoio à arquitetura pedagógica Debate de Teses estendido para contemplar notas de texto do professor.**

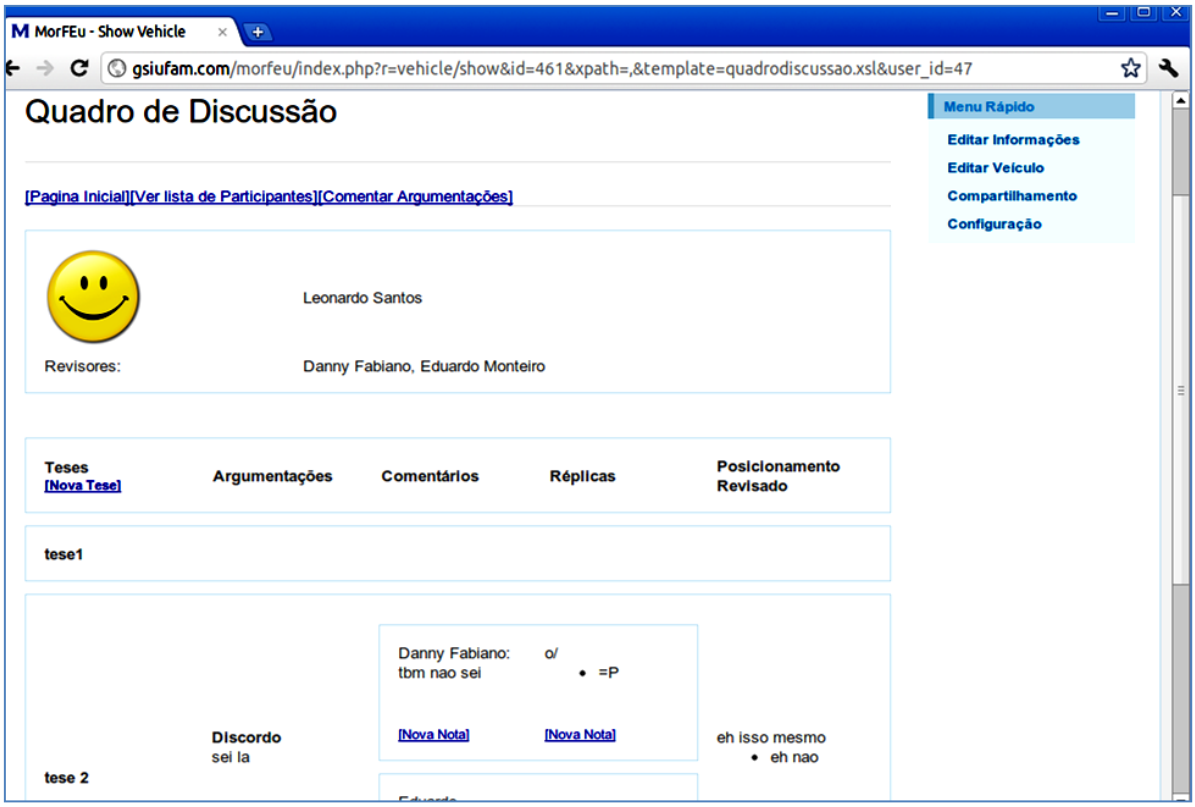


**Figura 19. Tela do protótipo funcionando, um quadro de discussão da arquitetura pedagógica debate de teses.**

## 5.4 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 propostos.

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.

Desejava-se avaliar também as questões da flexibilidade dos VComs. Para isso, se realizou um estudo de caso com usuários reais. Um estudo de caso é a metodologia mais adequada devido ao fato do fenômeno das demandas por flexibilidade e a

capacidade do MOrFEu de atender essas demandas estar muito ligada ao contexto que elas surgem, não havendo limites claramente evidentes.

### 5.4.1 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.

#### 5.4.1.1 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.

#### 5.4.1.2 Fórum da Controvérsia Acadêmica

Um fórum modificado para apoio à prática pedagógica **Controvérsia Acadêmica** realizada em ambientes virtuais (MENDONÇA, *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. Na 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.

#### 5.4.1.3 CARTOLA

Uma ferramenta web para o **objeto de aprendizagem CARTOLA** (ALVES, *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 dava suporte a UPIs que sejam imagens.

#### 5.4.1.4 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.

#### 5.4.1.5 Debate de Teses

Um quadro para a arquitetura pedagógica **Debate de Teses** (NEVADO, DALPIAZ e MENEZES, 2009) descrito anteriormente na Seção 2.2.3 e exemplificado na Seção 5.3.1.

#### 5.4.1.6 Projetos de Aprendizagem

Um ambiente para apoio básico a **Projetos de Aprendizagem** (FAGUNDES, *et al.*, 2005), descrito anteriormente na seção 5.3.1.

#### 5.4.1.7 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 "Debate de Teses", com revisão por pares, mas numa versão mais simples.

#### 5.4.1.8 Curso

Uma versão básica do **"Curso" do Moodle** (Moodle, 2011). Trata-se da página inicial de um curso, dividida em tópicos, onde em 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 15).

#### 5.4.1.9 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.

Esse ambiente virtual é um pouco diferente dos demais, pois em sua essência não é um ambiente virtual como foi apresentado. Ele foi utilizado como exemplo para verificar a capacidade do protótipo de implementar ambientes diversos.

#### 5.4.1.10 Conclusão

Os 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 de desenvolvimento flexível 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.

### 5.4.2 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 caso é parte do curso de formação de professores e multiplicadores do Projeto UCA – Um Computador por Aluno (MINISTÉRIO DA EDUCAÇÃO, 2011). Foram 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, indo o curso na modalidade a distância. E os tutores desse curso foram professores do Instituto de Computação da Universidade

Federal do Amazonas. Todos 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) que fizeram uso apenas de material convencional (papel e caneta). No fim processo, todos os participantes (três participantes efetivos e sete observadores) manifestaram 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 as atividades. 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 *bugs* simples e 11 eram para melhor acomodar preferências pessoais no VCom utilizado. 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 seriam mais adequadas 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, o qual 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, *et al.*, 2007).

## 5.5 Conclusão do Capítulo

A exploração da proposta MOrFEu, suas características e de ambientes implementados segundo esse modelo possibilitou identificar dificuldades e possibilidades na transposição dos elementos conceituais para ferramentas utilizadas em situações reais, bem como uma análise de seu escopo e escalabilidade, essenciais para o desenvolvimento do modelo proposto neste trabalho.

Identificou-se também a necessidade de um elemento de execução de procedimentos de forma automática, o que não é contemplado no MOrFEu, porque pressupõe que toda a construção do Documento é feito por usuários. Isso também é incorporado no modelo apresentado no próximo capítulo, através de agentes.

As Hipóteses H1 e H2 foram confrontadas através de provas de conceito. No entanto, um estudo de caso revelou uma limitação de modelagem de ferramentas com os conceitos previstos pelo MOrFEu. Assim, essa análise levou à invalidade da Hipótese H1 e subsequente formulação da Hipótese H1.1 de que Os elementos conceituais apresentados inicialmente pelo MOrFEu **acrescidos de alguns elementos** possibilitam a especificação de qualquer instância de ambiente virtual

utilizado em certo domínio de aplicação. No próximo capítulo é apresentado uma extensão do MOrFEu, buscando validar a Hipótese H1.1.

# 6 O Modelo Conceitual "MOrFEu eXtendido"

Partindo-se da exploração apresentada no capítulo anterior – identificação das dificuldades e na transposição dos elementos conceituais do MOrFEu para ferramentas utilizadas em situações reais – e das ideias de Ellis e Wainer (2000) – buscando obter aplicações do tipo Guardiãs de Artefato especificáveis, neste capítulo apresenta-se uma abordagem para especificação de ambientes virtuais web de interação humano-computador-humano.

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 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**, que visualiza as aplicações Coordenadoras e

Comunicadoras também como Guardiães, tornando-as concretas e modificáveis em tempo de execução.

A abordagem MX organiza de melhor forma os conceitos propostos no MOrFEu e elenca os elementos dos ambientes virtuais, de forma que é possível descrever precisamente a estrutura, funcionalidades e navegação de um ambiente virtual. Nessa abordagem, um ambiente virtual possui três elementos básicos: 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. Além disso, foi adicionado um novo tipo de elemento para tratar estados e permissões: os agentes. Os agentes são responsáveis por execução de procedimentos automáticos e execução de serviços necessários para alguns ambientes virtuais.

Além das definições e especificações de cada componente do modelo conceitual MX, é apresentado também a arquitetura dos ambientes virtuais construídos segundo o mesmo, bem como os algoritmos que executam essas aplicações, reunindo esses componentes num framework, possibilitando a implementação de ambientes virtuais flexíveis.

A seguir são descritos os elementos do MOrFEu Extendido bem como a modelagem dos ambientes virtuais que foram posteriormente implementados.

## 6.1 Definição do Modelo

O MX reúne os elementos do MOrFEu original, e acrescenta outras características. Definições mais específicas e mais palpáveis foram feitas de seus componentes. As pessoas, o Documento, o Protocolo de Interação e as Interfaces são os componentes essenciais do modelo conceitual MX.

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 contém 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 foi chamada de Unidade de Produção Intelectual (UPI). Uma postagem num fórum, uma mensagem num chat, uma página num 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** para os elementos gerais do MX. Algumas convenções na notação foram utilizadas:

- Letras gregas maiúsculas são utilizadas para conjuntos;
- Letras gregas minúsculas são utilizadas para relações, por exemplo, $\varphi \subseteq \Psi \times \Xi \times \Pi$;
- Letras romanas maiúsculas são usadas para componentes em tuplas, por exemplo, $E = \langle D, I, P \rangle$; e
- Letras romanas minúsculas para elementos de conjuntos.

**Definição 1.** Existe um conjunto Π de usuários do ambiente.

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

$$E = \langle D, I, P \rangle$$

onde $D$ é um documento, $I$ é um protocolo de interação e $P$ 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 Ψ de todas as UPIs de todos os usuários.

**Definição 5.** Existe um conjunto ρ 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:

$$\rho = \{\langle u, p \rangle | \forall u \in \Psi \wedge \exists p \in \Pi\}$$

**Definição 6.** Existe um conjunto Ξ de rótulos, que são strings e indicam qual a natureza da publicação da UPI ou da relação entre as UPIs.

**Definição 7.** Existe um conjunto Γ de condições, que são funções lógicas.

### 6.1.1 Documento

O Documento armazena as 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 porquê 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 mesmo 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 estar associada 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 o Documento são necessárias **operações**. Essas operações podem ser de **leitura** (consulta ou leitura dos 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 são fixas, em um conjunto finito e bem definido, em contraste com as Pesquisas e Atividades, que são especificáveis.

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

O resultado de uma operação de consulta ainda passa por um a filtragem: apenas as UPIs e relações que possuem permissões de leitura, definidas no Protocolo de Interação, são retornadas.

Uma **Pesquisa** é uma 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 geral do documento, mas não necessariamente modifica as UPIs já publicadas.

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

**Definição 8.** 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; e um conjunto de UPIs publicadas e grupos. Tal que o documento pode ser representado como $\mathsf{D} = \langle \varphi, \alpha, \beta, \gamma \rangle$ em que:

- $\varphi$ é um conjunto de UPIs publicadas, os vértices do grafo, onde $\varphi \subseteq \Psi \times \Xi \times \Pi$, UPIs publicadas por usuários com um determinado rótulo, tal que:
  - $\Psi$ é o conjunto de UPIs
  - $\Pi$ é o conjunto de usuários

- $\alpha \subseteq \varphi \times \varphi \times \Xi \times \Pi$ 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;
- $\beta \subseteq \varphi \times \Xi \times \Pi$ é um conjunto de relações entre UPIs publicadas e usuários, denotando que outros usuários, além do autor, podem ter relação com essa UPI; e
- $\gamma \subseteq \varphi \times \Xi \times \Theta$ é um conjunto de relações entre UPIs publicadas e grupos, denotando que grupos podem ter relação com essa UPI, onde o conjunto de grupos $\Theta$ faz parte do protocolo de interação $I$.

**Definição 9.** O conjunto $\Delta$ refere-se ao conjunto de todos os possíveis documentos, tal que:

$$\Delta = (\Psi \times \Xi \times \Pi) \times (\varphi \times \varphi \times \Xi \times \Pi) \times (\varphi \times \Xi \times \Pi) \times (\varphi \times \Xi \times \Theta)$$

Esse conjunto será usado como domínio e contra-domínio em funções de operações de alteração de um documento.

**Definição 10.** As operações de consulta no documento são funções que mapeiam o documento a um conjunto de resultados, um conjunto de UPIs. O conjunto $\Upsilon$ é o conjunto das funções de operações de consulta, composto pelas funções:

- $consultaUPI: \Gamma \times \Delta \rightarrow \varphi$ é uma função que mapeia um documento num conjunto de UPIs do documento que satisfaçam uma dada condição:

$$consultaUPI(condicao, D) = \left\{ u \middle| \begin{matrix} \mathrm{D} = \langle \varphi, \alpha, \beta, \gamma \rangle \\ \wedge\, u \in \varphi \\ \wedge\, condicao(u) \end{matrix} \right\}$$

- $upisRelComUPI: \varphi \times \Gamma \times \Gamma \times \Delta \rightarrow \varphi$ é uma função que mapeia um documento num conjuntos de UPIs do documento que estejam relacionadas com uma dada UPI, satisfaçam uma dada condição de seleção da UPI e uma condição de seleção da relação:

$$upisRelComUPI(upi, condUPI, condRel, D) = \left\{ u \middle| \begin{matrix} \mathrm{D} = \langle \varphi, \alpha, \beta, \gamma \rangle \\ \wedge\, u \in \varphi \\ \wedge\, condUPI(u) \\ \wedge\, rel \in \alpha \\ \wedge\, rel = \langle u, upi, r, a \rangle \\ \wedge\, condRel(rel) \end{matrix} \right\}$$

- $upisRelComUsuario: \Pi \times \Gamma \times \Gamma \times \Delta \rightarrow \varphi$ é uma função que mapeia um documento num conjuntos de UPIs do documento que estejam relacionadas com um dado Usuário, satisfaçam uma dada condição de seleção da UPI e uma condição de seleção da relação:

$$upisRelComUsuario(usuario, condUPI, condRel, D) = \left\{ u \left| \begin{array}{c} \mathrm{D} = \langle \varphi, \alpha, \beta, \gamma \rangle \\ \wedge\, u \in \varphi \\ \wedge\, condUPI(u) \\ \wedge\, rel \in \beta \\ \wedge\, rel = \langle u, r, usuario \rangle \\ \wedge\, condRel(rel) \end{array} \right. \right\}$$

- $upisRelComGrupo: \Theta \times \Gamma \times \Gamma \times \Delta \rightarrow \varphi$ é uma função que mapeia um documento num conjuntos de UPIs do documento que estejam relacionadas com um dado Usuário, satisfaçam uma dada condição de seleção da UPI e uma condição de seleção da relação:

$$upisRelComGrupo(grupo, condUPI, condRel, D) = \left\{ u \left| \begin{array}{c} \mathrm{D} = \langle \varphi, \alpha, \beta, \gamma \rangle \\ \wedge\, u \in \varphi \\ \wedge\, condUPI(u) \\ \wedge\, rel \in \gamma \\ \wedge\, rel = \langle u, r, grupo \rangle \\ \wedge\, condRel(rel) \end{array} \right. \right\}$$

**Definição 11.** As operações de alteração mapeiam um documento em outro documento, fazendo inserções ou remoções de UPIs publicadas ou relações. O conjunto $\Phi$ é o conjunto das funções de operações de alteração, composto pelas funções:

- $publicar: \Psi \times \Xi \times \Pi \times \Delta \rightarrow \Delta$ é um função que mapeia um documento, uma UPI e um rótulo num novo documento, com essa UPI publicada:

$$publicar(upi, rotulo, publicador, \langle \varphi, \alpha, \beta, \gamma \rangle) = \langle \varphi \cup \langle upi, rotulo, publicador \rangle, \alpha, \beta, \gamma \rangle$$

- $relacionarUpis: \varphi \times \varphi \times \Xi \times \Pi \times \Delta \rightarrow \Delta$ é um função que mapeia um documento, duas UPIs e um rótulo num novo documento, com essas duas UPIs possuindo uma relação entre elas:

$$relacionarUpis(upiA, upiB, rotulo, publicador, \langle \varphi, \alpha, \beta, \gamma \rangle) = \langle \varphi, \alpha \cup \langle upiA, upiB, rotulo, publicador \rangle, \beta, \gamma \rangle$$

- $relacionarUpiUsuario: \varphi \times \Xi \times \Pi \times \Delta \to \Delta$ é um função que mapeia um documento, uma UPI, um usuário e um rótulo num novo documento, com essa UPI possuindo uma relação com o usuário:

$$relacionarUpiUsuario(upi, rotulo, usuario, \langle \varphi, \alpha, \beta, \gamma \rangle) = \langle \varphi, \alpha, \beta \cup \langle upi, rotulo, usuario \rangle, \gamma \rangle$$

- $relacionarUpiGrupo: \varphi \times \Xi \times \Theta \times \Delta \to \Delta$ é um função que mapeia um documento, uma UPI, um usuário e um rótulo num novo documento, com essa UPI possuindo uma relação com o usuário:

$$relacionarUpiGrupo(upi, rotulo, grupo, \langle \varphi, \alpha, \beta, \gamma \rangle) = \langle \varphi, \alpha, \beta, \gamma \cup \langle upi, rotulo, grupo \rangle \rangle$$

- $despublicar: \Psi \times \Xi \times \Pi \times \Delta \to \Delta$ é um função que mapeia um documento, uma UPI e um rótulo num novo documento, com essa UPI despublicada:

$$despublicar(upi, rotulo, publicador, \langle \varphi, \alpha, \beta, \gamma \rangle) = \langle \varphi - \{\langle upi, rotulo, publicador \rangle\}, \alpha, \beta, \gamma \rangle$$

- $desrelacionarUpis: \varphi \times \varphi \times \Xi \times \Pi \times \Delta \to \Delta$ é um função que mapeia um documento, duas UPIs e um rótulo num novo documento, com retirando a relação entre essas duas UPIs:

$$derelacionarUpis(upiA, upiB, rotulo, publicador, \langle \varphi, \alpha, \beta, \gamma \rangle) = \langle \varphi, \alpha - \{\langle upiA, upiB, rotulo, publicador \rangle\}, \beta, \gamma \rangle$$

- $desrelacionarUpiUsuario: \varphi \times \Xi \times \Pi \times \Delta \to \Delta$ é um função que mapeia um documento, uma UPI, um usuário e um rótulo num novo documento, com essa UPI não mais possuindo essa relação com o usuário:

$$desrelacionarUpiUsuario(upi, rotulo, usuario, \langle \varphi, \alpha, \beta, \gamma \rangle) = \langle \varphi, \alpha, \beta - \{\langle upi, rotulo, usuario \rangle\}, \gamma \rangle$$

- $desrelacionarUpiGrupo: \varphi \times \Xi \times \Theta \times \Delta \rightarrow \Delta$ é um função que mapeia um documento, uma UPI, um usuário e um rótulo num novo documento, com essa UPI não mais possuindo essa relação com o usuário:

$$desrelacionarUpiGrupo(upi, rotulo, grupo, \langle \varphi, \alpha, \beta, \gamma \rangle) = \langle \varphi, \alpha, \beta, \gamma - \{\langle upi, rotulo, grupo \rangle\} \rangle$$

### 6.1.2 Protocolo

O **Protocolo de Interação** (ou apenas Protocolo para simplificar) define como o veículo de comunicação pode ser utilizado. Ou seja, trata principalmente das permissões de operações a 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 têm permissão para serem executadas. Algo parecido acontece para as **pesquisas** (função com várias operações de consulta ao Documento), somente as UPIs e Relações que possuem permissão das operações de consulta são retornadas, ou seja, uma pesquisa pode retornar vazio se nada for permitido ser consultado.

O processo de inferência sobre as permissões é regido por um sistema de **negação por falha**, ou seja, se não houver algo explicitamente permitido de ser feito, então isso não é permitido.

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

**Definição 12.** Um protocolo de interação $I$ é formado por **seis elementos**, de forma que $I = \langle \mathrm{P}, \Theta, \Sigma, \Sigma', \Lambda, \kappa \rangle$, onde:

- $\mathrm{P}$ é um conjunto de papéis de usuários, de forma que $r \in \mathrm{P} = \langle Rotulo, \Omega \rangle$, onde $Rotulo$ é uma string e $\Omega \subseteq \Pi$;
- $\Theta$ é um conjunto de grupos de usuários, de forma que $g \in \Theta = \langle Rotulo, \Omega \rangle$, onde $Rotulo$ é uma string e $\Omega \subseteq \Pi$;
- $\Sigma$ é um conjunto de estados;
- $\Sigma'$ é um conjunto de estados atualmente ativos, de forma que $\Sigma' \subseteq \Sigma$;

- $\Lambda$ é um conjunto de permissões;
- $\kappa$ é um conjunto de relações entre estados e permissões, denotando quais permissões são aplicáveis em cada estado, de forma que $\kappa \subseteq \Sigma \times \Lambda$;
- A é um conjunto de atividades;
- B é um conjunto de pesquisas;

**Definição 13.** Uma **permissão** $l \in \Lambda$ é formada por quatro elementos, de forma que $l = \langle r, o, s, \varepsilon \rangle$, onde:

- $r$ é um papel, de forma que $r \in$ P;
- $o$ é uma operação sobre o Documento, ou sobre o Protocolo, ou sobre a Interface, de forma que $o \in (\Upsilon \cup \Phi)$;
- $s$ é um estado, de forma que $s \in \Sigma$;
- $\varepsilon$ é uma função de $condicao(u, o)$, que mapeia para o conjunto $\{Verdadeiro, Falso\}$, onde $u \in \Psi$ é o usuário requisitando a permissão e $o$ é a operação requisitada.

**Definição 14.** Uma **pesquisa** é uma função que utiliza operações de consulta para mapear um documento em um subconjunto das UPIs publicadas. De forma que:

Uma pesquisa $p \in$ B, tal que $p: \Delta \rightarrow \varphi$

**Definição 15.** Uma **atividade** é uma lista, uma sequência finita, de operações de alteração. De forma que:

Uma atividade $a \in$ A, tal que $a = \Phi^* = [o_0, o_1, o_2, \dots, o_n]$, onde $o_i \in \Phi$ e $i \in \mathbb{N}$

As transições entre esses estados são executados na forma de **operações** do Protocolo. São duas as operações: ativar e desativar um estado, que inclui ou exclui um estado do conjunto dos estados atuais, respectivamente.

**Definição 16.** As operações do protocolo mapeiam um conjunto de estados atuais em outro conjunto de estado atuais, fazendo inserções ou remoções de estados. O conjunto X é o conjunto das funções de operação do protocolo, composto pelas funções:

- *ativarEstado*: $\Sigma \times \{\Sigma'\} \rightarrow \{\Sigma'\}$ uma função que mapeia um estado e um conjunto de estados atuais em outro conjunto de estados atuais, inlcuindo esse estado:

$$ativarEstado(estado, estadosAtuais) = estadosAtuais \cup \{estado\}$$

- *desativarEstado*: $\Sigma \times \{\Sigma'\} \rightarrow \{\Sigma'\}$ uma função que mapeia um estado e um conjunto de estados atuais em outro conjunto de estados atuais, excluindo esse estado:

$$desativarEstado(estado, estadosAtuais) = estadosAtuais - \{estado\}$$

### 6.1.3 Interface

A Interface descreve a apresentação (gráfica) do sistema para 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 é 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, uma página pode estar associada uma atividade diferente para cada par estado-papel.

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 seus próprios templates 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; permissões de exibição de páginas de conteúdo; e permissões de execução de páginas de atividades.

**Definição 17.** A **interface** $W$ de um VCom é formado por cinco componentes, de forma que $W = \langle \Omega_c, \Omega_a, \mathrm{T}, \upsilon, \zeta \rangle$, onde:

- A página de conteúdo $p_c \in \Omega_c = \langle Rotulo \rangle$ possui um rótulo do nome da página, e o template que gera seu conteúdo é definido por $Y$;
- A página de atividade $p_a \in \Omega_a = \langle Rotulo \rangle$ possui um rótulo do nome da página, e a atividade que é executada por essa página é definida por $Z$;
- $\mathrm{T}$ é um conjunto de templates de forma que seus elementos são pesquisas que fazem consultas no Documento e geram strings no formato HTML;
- $\upsilon \subseteq \Sigma \times \mathrm{P} \times \Omega_c \times \mathrm{T}$ é um conjunto de permissões de página que define "qual o conteúdo que será exibido para um usuário, dependendo do estado do VCom e do papel que ele interpreta", onde $\Sigma$ é o conjunto de estados e $R$ é o conjunto de papéis.
- $\zeta \subseteq \Sigma \times \mathrm{P} \times \Omega_a \times \mathrm{A}$ é um conjunto que define "qual atividade que será executada para um usuário, dependendo do estado do VCom e do papel que ele interpreta", onde $\Sigma$ é o conjunto de estados, $\mathrm{P}$ é o conjunto de papéis e $\mathrm{A}$ é o conjunto de atividades.

## 6.2 Agentes

Nem todos os ambientes virtuais podem ser implementados simplesmente montando os componentes anteriormente citados, como um Guardião de Artefato.

Uma funcionalidade comum, como uma busca por palavra-chave, não pode ser implementada eficientemente nesse modelo. Uma solução comum para tal esse tipo de problema é o uso de um serviço externo de indexação e busca dos itens presentes no ambiente virtual. Esses tipos de serviços podem ser caracterizados como agentes de software.

Nesta seção apresenta-se um componente do MX que prevê o uso dessas funcionalidades externas ao VCom, ou seja, procedimentos automatizados que não são executados por usuários normais, os agentes. Eles podem atuar na exibição de páginas, na composição do Documento e na regulação das permissões, executando operações sobre o Protocolo.

Um agente é um programa de computador que interage em certo ambiente (não necessariamente virtual ou na web). Um **agente** é capaz de perceber seu **ambiente** por meio de **sensores** e de agir sobre esse ambiente por intermédio de **atuadores** (RUSSEL e NORVIG, 2004, p. 33). Dentre os vários tipos de agentes destacam-se os agentes reativos simples – que respondem diretamente a percepções –, agentes reativos baseados em modelos – que mantêm estados internos para controlar aspectos do mundo que não estão evidentes na percepção atual –, agentes baseados em objetivos – que agem para alcançar seus objetivos – e os agentes baseados em utilidade – que tentam maximizar sua própria "felicidade" esperada.

Ellis (1998) fala de tais agentes, chamando-os de Agentes de Time. São processos automáticos que realizam funções especializadas dentro de um contexto de groupware, construídos para executar sub-tarefas específicas, e não globais.

Em ambientes virtuais MX, os agentes agem como um usuário normal agiria: percebe o ambiente através de pesquisas e operações de leitura e atua sobre o ambiente com atividades e operações de escrita. Obviamente, também interpretam papéis e estão sujeitos às permissões pertinentes. Sendo assim, os agentes fazem o que um usuário comum poderia fazer.

No entanto, o propósito de um agente é realizar tarefas que reduziriam a carga cognitiva dos usuários do sistema. Por exemplo, esses auxiliares podem atuar na mudança automática de estados do ambiente, prover serviços de busca, atuar na interface de modo a melhorar a percepção dos usuários do sistema (SPÓSITO, CASTRO e CASTRO, 2008), etc.

Os **agentes gatilho** são agentes reativos. Os gatilhos (*triggers*) são atividades especiais que são realizadas automaticamente quando uma condição é satisfeita. Por exemplo, uma **troca de estado** é um tipo de gatilho, onde se espera alguma condição ser atendida e promove-se uma troca do estado atual para outro, ou seja, o agente gatilho executa uma atividade com operações sobre o Protocolo: 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.

Exemplos desse tipo de agente são os buscadores: por exemplo, eles podem indexar o conteúdo das UPIs publicadas, utilizando pesquisas, e respondem a buscas por palavras-chave com uma lista de UPIs. As requisições a esses agentes devem ser feitas no código dos templates.

Outro exemplo são os tradutores, que podem trabalhar traduzindo mensagens dos participantes num contexto de colaboração multinacional.

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 das decisões tomadas, podendo 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 apresentação de possíveis agentes atuadores em VCom não é restritiva a somente essas classes. Trata-se apenas de uma organização de possíveis tipos de agentes que podem atuar em VComs.

Dessa forma, é possível prever o uso de agentes em ambientes virtuais, que realizam ações rotineiras ou que dão um suporte inteligente a seus utilizadores. Em sua essência, são programas de computador que realizam tarefas marginais ao VCom, muitas vezes podendo atuar como funcionalidades do Guardião do Artefato.

**Definição 18.** Um **agente** é um usuário, de forma que ele pertence ao conjunto Π de usuários do ambiente. De forma que um agente é subordinado ao Protocolo, Pesquisas e Atividades da mesma forma que um usuário normal. Sua diferença pode estar em um conjunto diferenciado de papéis e permissões que pode se dar a um agente. No entanto, um agente pode ser uma ferramenta de software que realiza procedimentos automatizados.

$$agente \in \Pi$$

## 6.3 Modelagem de Ambientes Virtuais com o MX

Nesta seção são apresentados quatro exemplos de modelagem de ferramentas através do MX. São eles: o Fórum, o Diário de Resolução de Problemas de Matemática, o Debate de Teses e o Jogo de Dominó. Esses ambientes virtuais farão parte dos exemplos de implementação do próximo capítulo.

Vários outros exemplos de modelagem de ambientes virtuais com o MX podem ser encontrados no Apêndice A. Os exemplos implementados têm por objetivo contribuir como prova de conceito dos elementos de modelagem do MX, de modo análogo aos descritos na Seção 5.4.1.

Os quatro ambientes apresentados neste capítulo foram escolhidos pela sua diversidade que, em conjunto, cobrem a quase totalidade das características que o modelo conceitual MX se propõe a descrever. Esses foram implementados em protótipos descritos no próximo capítulo, segundo um framework desenvolvido.

O Fórum é um ambiente virtual bem simples e bastante utilizado. O Diário de Resolução de Problemas é um ambiente virtual um pouco mais complexo, mas que já possui características com relação a papéis e permissões. O Debate de Teses possui um maior grau de complexidade que os dois primeiros, onde é feito uma espécie de revisão por pares, com estados e permissões bem definidas.

O Jogo de Dominó, por sua vez, não é um ambiente virtual de uso em atividades pedagógicas. Trata-se de um jogo jogado em duplas, com alto grau de complexidade e jogadas que necessitam de boa dose de raciocínio e de memória de seus jogadores. A escolha desse ambiente virtual se deu pela sua alta complexidade no Protocolo de Interações, com permissões em nível de operações, diferente dos outros três, que somente necessitavam de permissões em nível de páginas e atividades.

Não foram encontrados, na literatura, ambientes virtuais de apoio pedagógico que possuíssem grau de complexidade de Protocolo de Interação parecido com o Jogo de Dominó, possivelmente, devido ao fato de ainda não existirem ferramentas para autoria desse tipo de ambiente.

### 6.3.1 Fórum

Um fórum é uma ferramenta de discussão de um tema específico. 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[40] 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.

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 resposta a outro post já publicado no VCom. Na Figura 20, está o modelo do documento desse tipo de VCom, utilizando-se caixas para representar as UPIs e setas para representar as relações entre as UPIs.

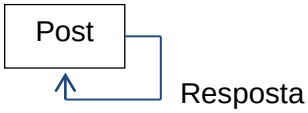


**Figura 20. Diagrama do modelo de documento de um fórum.**

A modelagem da Figura 20 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.

Algumas atividades desse VCom são:

- Incluir Post-Raiz: publica uma UPI com o rótulo "Post";

$$[publica(upi, \text{"Post"}, publicador, D_0)]$$

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

$$\begin{bmatrix} publica(upi, \text{"Post"}, publicador, D_0), \\ relacionaUpis(upi, \langle upi, \text{"Post"}, publicador \rangle, \lambda, publicador, D_1) \end{bmatrix}$$

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:

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

- Os participantes podem consultar o Documento inteiro

$$\forall r \in \mathrm{P}, \forall o \in \Upsilon \ \langle r, o, inicial, Verdadeiro \rangle$$

- Participantes podem publicar Posts

$$\langle participante, publicar(upi, \text{"Post"}, publicador, D), inicial, Verdadeiro \rangle$$

- Participantes podem publicar relações de Resposta entre Posts

$$\begin{pmatrix} participante, relacionarUpis(\langle \_, \text{"Post"}, \_ \rangle, \langle \_, \text{"Post"}, \_ \rangle, \text{"Resposta"}, publicador, D), \\ inicial, \\ Verdadeiro \end{pmatrix}$$

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 21, 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 "bonecos magros" representam a quais papéis o template está associado.

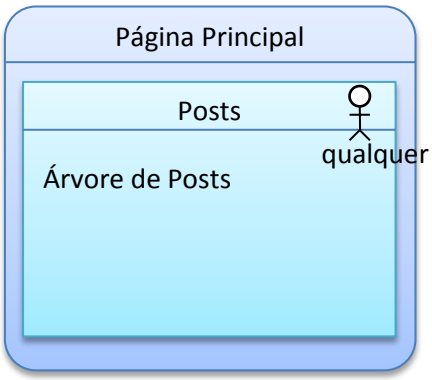


**Figura 21. Diagrama de navegação da interface de um fórum.**

### 6.3.2 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 "Debate de Teses", 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, a etapa de revisões é iniciada. Cada estudante deve comentar a solução de pelo menos outros dois estudantes. Serres e Basso (2009) comentam que 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 “Debate de Teses”, 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 cada 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 22, está representado o modelo desse Documento, muito similar ao do Blog, apresentado na Figura 73. Nesse diagrama são utilizadas caixas para representar as UPIs e setas para representar as relações entre as UPIs


**Figura 22. 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”

$$[publica(upi, \text{"Post"}, publicador, D_0)]$$

- Incluir Comentário: publica uma UPI com o rótulo “Comentário” e publica a relação desse comentário com um Post.

$$\begin{bmatrix} publica(upi, \text{"Comentario"}, publicador, D_0), \\ relacionaUpis(\langle upi, \text{"Comentario"}, publicador\rangle, post, \lambda, publicador, D_1) \end{bmatrix}$$

Algumas pesquisas desse VCom são:

- Pesquisa o diário de um usuário

$$diario(usuario, D) = consultaUPI\big((u = \langle \_, \text{"Post"}, usuario\rangle), D\big)$$

- Pesquisa comentários de um diário

$$comentarios(diario, D) = upisRelComUPI(diario, Verdadeiro, Verdadeiro, D)$$

- Pesquisa todos os diários

$$todosDiarios(usuario, D) = consultaUPI\big((u = \langle \_, \text{"Post"}, \_\rangle), D\big)$$

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

$$\langle participante, consultaUPI(u, D), inicial, (u = \langle \_, \text{"Post"}, usuario\rangle)\rangle$$

- O professor pode consultar todo o Documento

$$\forall o \in \Upsilon\ \langle professor, o, inicial, Verdadeiro\rangle$$

- Um participante pode publicar uma UPI Post, se ele já não tiver publicado nenhuma UPI Post anteriormente;

$$\begin{pmatrix} participante, publicar(upi, \text{"Post"}, publicador, D), inicial, \\ (diario(publicador, D) = \emptyset) \end{pmatrix}$$

- 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 23, 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.

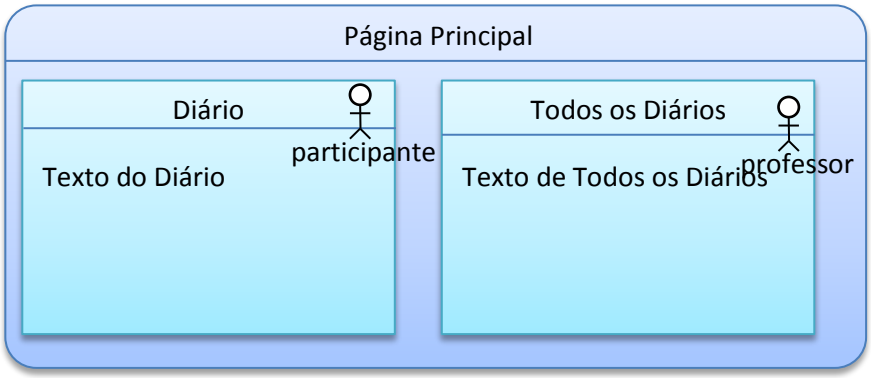


**Figura 23. Diagrama de navegação da interface de um diário de resolução de problemas, no estado inicial.**

**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;

$$\forall r \in \mathrm{P}, \forall o \in \Upsilon \ \langle r, o, revisao, Verdadeiro \rangle$$

- Um participante ou professor pode publicar Comentários, e publicar relações desse comentário com um Post;

$$\langle participante, publicar(upi, \text{"Comentario"}, publicador, D), revisao, Verdadeiro \rangle$$

$$\forall r \in [participante, professor]$$

$$\left( \begin{array}{c} r, \\ relacionarUpis(\langle \_, \text{"Comentario"}, \_ \rangle, \langle \_, \text{"Post"}, \_ \rangle, \lambda, publicador, D), \\ revisao, \\ Verdadeiro \end{array} \right)$$

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 24, é apresentado o diagrama de navegação para esse estado de revisões.

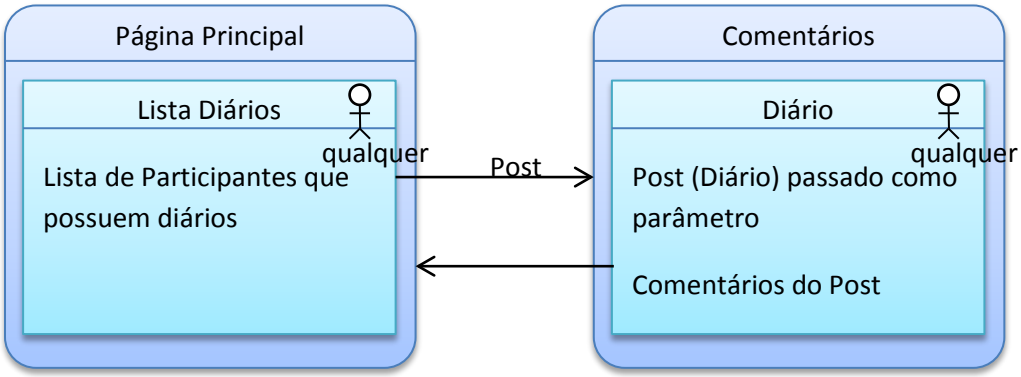


**Figura 24. Diagrama de navegação da interface de um diário de resolução de problemas, no estado de revisões.**

### 6.3.3 Debate de Teses

A arquitetura pedagógica “Debate de Teses”, também chamada de “Construindo Conceituações”, 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 possível inclusive a inserção de novas etapas”. Como resultado, foi elaborada uma atividade com seis 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 25.

| **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 25. Exemplo de quadro da arquitetura pedagógica Debate de Teses.**

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 26, pode-se ver uma tela de um desses debates de teses funcionando num wiki.

Dessa forma, pode-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 27.

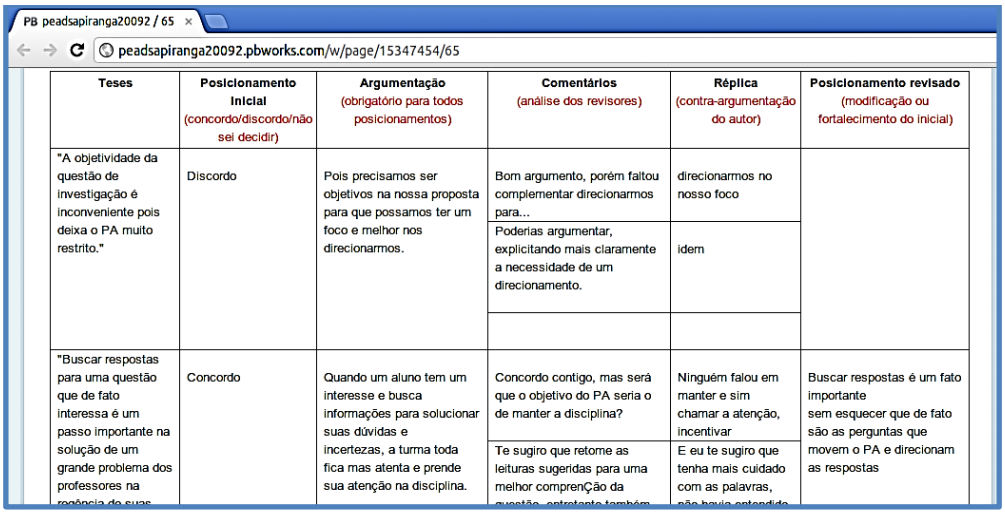


**Figura 26. Tela de um debate de teses funcionando num wiki.**

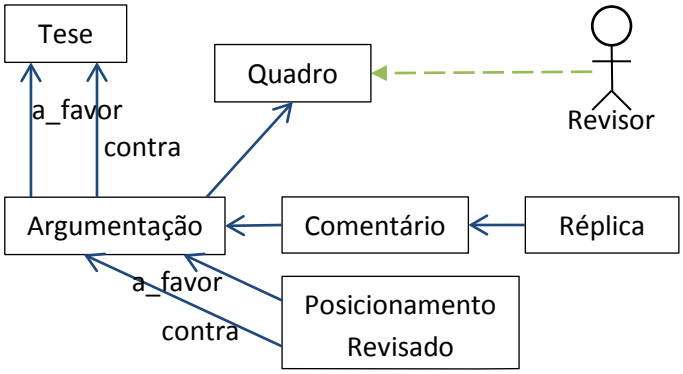


**Figura 27. Diagrama do modelo de documento de um quadro de debate de teses.**

Algumas atividades desse VCom são:

- Incluir Tese: publica uma UPI Tese;

$$[publica(upi, \text{"Tese"}, publicador, D_0)]$$

- 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;

$$\left[\begin{array}{c} publica(upi, \text{"Argumentacao"}, publicador, D_0), \\ relacionaUpis(\langle upi, \text{"Argumentacao"}, publicador\rangle, tese, pos, publicador, D_1), \\ publica(\lambda, \text{"Quadro"}, publicador, D_2), \\ relacionaUpis\left(\begin{array}{c}\langle upi, \text{"Argumentacao"}, publicador\rangle, \langle \lambda, \text{"Quadro"}, publicador\rangle, \\ \lambda, publicador, D_3\end{array}\right) \end{array}\right]$$

- Incluir Comentário de uma Argumentação: publica uma UPI Comentário e relaciona essa UPI com a Argumentação;

$$\left[\begin{array}{c} publica(upi, \text{"Comentario"}, publicador, D_0), \\ relacionaUpis(\langle upi, \text{"Comentario"}, publicador\rangle, argumentacao, \lambda, publicador, D_1) \end{array}\right]$$

- Incluir Réplica de um Comentário: publica uma UPI Réplica e relaciona essa UPI com o Comentário;

$$\left[\begin{array}{c} publica(upi, \text{"Replica"}, publicador, D_0), \\ relacionaUpis(\langle upi, \text{"Replica"}, publicador\rangle, comentario, \lambda, publicador, D_1) \end{array}\right]$$

- Incluir Posicionamento Revisado: publica uma UPI Posicionamento Revisado e relaciona essa UPI com a Argumentação Inicial com o rótulo do posicionamento.

$$\left[\begin{array}{c} publica(upi, \text{"PosRev"}, publicador, D_0), \\ relacionaUpis(\langle upi, \text{"PosRev"}, publicador\rangle, argumentacao, \lambda, publicador, D_1) \end{array}\right]$$

### 6.3.4 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 sempre que tenham pontas iguais. O objetivo do jogo é fazer mais pontos que a outra dupla. Os pontos são feitos apenas 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[41]. Mas infelizmente ele foi descontinuado no ano de 2004[42] e atualmente a sua página inicial apresenta somente a frase "EM CONSTRUÇÃO". É 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).

[41] http://www.dominonet.com.br
[42] http://web.archive.org/20030603041953/http://www.dominonet.com.br/

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. 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 por 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 6 da pedra $\langle 2|6 \rangle$ e o 5 da pedra $\langle 1|5 \rangle$. Excepcionalmente, a carroça inicial pode receber até quatro encaixes.

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 10 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 com o jogador que bateu na jogada anterior, e ele deve jogar uma carroça. Se ele não possuir nenhuma carroça para jogar, então ele passa e cede a saída ao próximo jogador.


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

Um jogador faz pontos quando joga uma pedra na mesa e a soma das pontas da mesa é um múltiplo de 5. Por exemplo, na Figura 28, o jogador não marcou 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 apresentado um diagrama do documento desse VCom.

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.

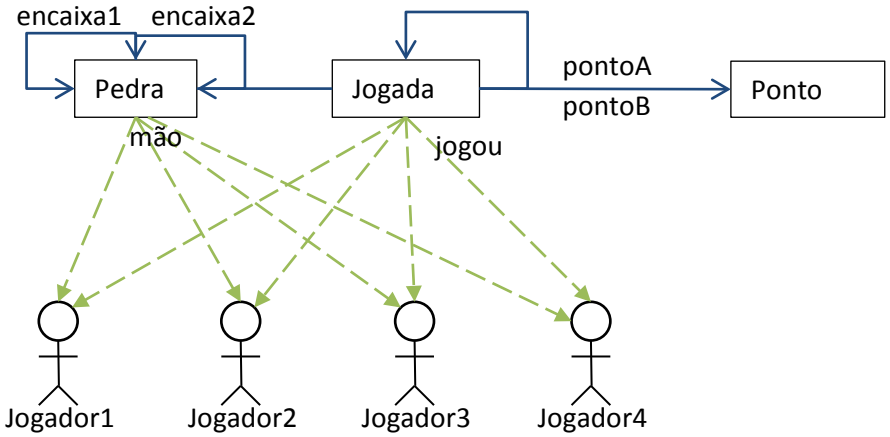


**Figura 29. Diagrama do modelo de documento de um jogo de dominó.**

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;

$$Pedras = consultaUPI\big((u = \langle \_, \text{"Pedra"}, \_ \rangle), D\big) \wedge M1 \cup M2 \cup M3 \cup M4 = Pedras$$
$$\wedge M1 \cap M2 = \emptyset \wedge M1 \cap M3 = \emptyset \wedge M1 \cap M4 = \emptyset$$
$$\wedge M2 \cap M3 = \emptyset \wedge M2 \cap M4 = \emptyset \wedge M3 \cap M4 = \emptyset$$
$$\forall p \in M1, [relacionarUpiUsuario(p, \text{"mao"}, jogador1, D]$$
$$\forall p \in M2, [relacionarUpiUsuario(p, \text{"mao"}, jogador2, D]$$
$$\forall p \in M3, [relacionarUpiUsuario(p, \text{"mao"}, jogador3, D]$$
$$\forall p \in M4, [relacionarUpiUsuario(p, \text{"mao"}, jogador4, D]$$

- 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.

$$\left[\begin{array}{c} relacionaUpis(pedra, pedraPonta, encaixe, publicador, D_0), \\ publica(n, \text{"Jogada"}, publicador, D_1), \\ relacionaUpis\left(\begin{array}{c} \langle n, \text{"Jogada"}, publicador\rangle, \langle (n-1), \text{"Jogada"}, publicador\rangle, \\ \lambda, publicador, D_2 \end{array}\right), \\ relacionaUpis(\langle \lambda, \text{"Jogada"}, publicador\rangle, pedra, \lambda, publicador, D_3) \end{array}\right]$$

O Protocolo de Interação desse VCom é bem complexo, pois não pode dar margem a jogadas erradas, em momentos errados, 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 definir 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.

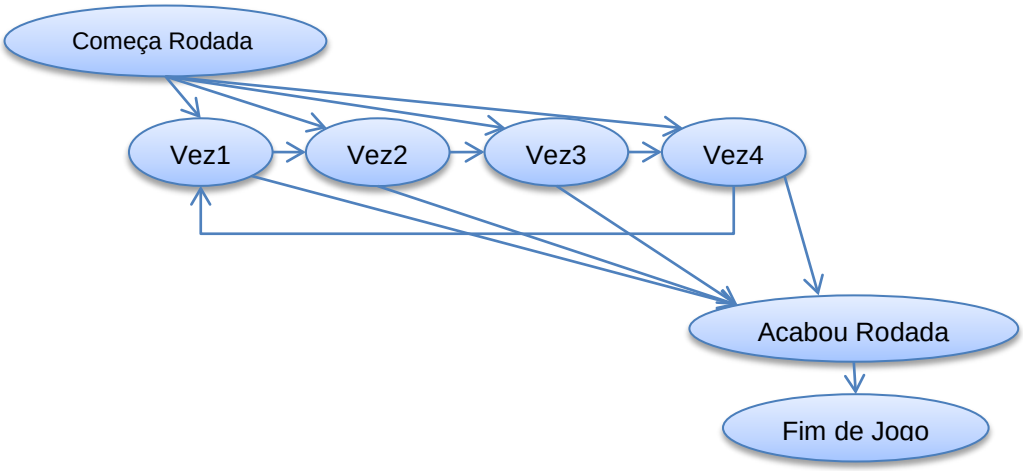


**Figura 30. Diagrama de estados de um jogo de dominó.**

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 todas as UPIs Pedra;

$$\forall r \in \mathrm{P}, \forall s \in \Sigma \ \langle r, consultaUPI(u, D), s, (u = \langle \_, \text{"Pedra"}, usuario \rangle) \rangle$$

- Os jogadores podem consultar suas mãos, ou seja, consultar as relações UPI-Usuário entre UPIs Pedra e ele mesmo;

$$\forall s \in \Sigma \ \langle jogador, upisRelComUsuario(usuario, rel, D), s, (r = \langle \_, \text{"mao"}, usuario \rangle) \rangle$$

- Os jogadores podem consultar todas as jogadas, as relações entre as jogadas, e as relações entre jogadas e pedras;

$$\forall r \in \mathrm{P}, \forall s \in \Sigma \ \langle p, upisRelComUPI(ua, ub, rel, D), s, (ua = \langle \_, \text{"Jogada"}, \_ \rangle) \rangle$$
$$\forall r \in \mathrm{P}, \forall s \in \Sigma \ \langle p, upisRelComUPI(ua, ub, rel, D), s, (rel = \langle \_, \_, \text{"encaixe1"}, \_ \rangle) \rangle$$
$$\forall r \in \mathrm{P}, \forall s \in \Sigma \ \langle p, upisRelComUPI(ua, ub, rel, D), s, (rel = \langle \_, \_, \text{"encaixe2"}, \_ \rangle) \rangle$$

- 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 da regra anterior para a carroça inicial, que pode ter até quatro encaixes (dois encaixes ao centro e dois às pontas), 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 10 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.

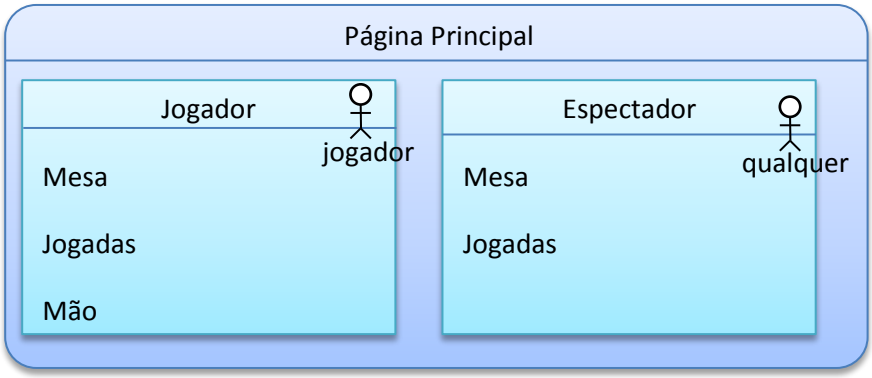


**Figura 31. Diagrama de navegação da interface de um jogo de dominó.**

O Jogo de Dominó não é um ambiente virtual de uso em atividades pedagógicas. Esse ambiente virtual possui uma alta complexidade no Protocolo de Interações, com permissões em nível de operações, e bastante controle de *workflow*.

Esse tipo de jogo se assemelha a dinâmicas pedagógicas realizadas em sala de aula, por isso o Dominó se torna interessante. Por exemplo, o jogo "batata-quente", onde um grupo de crianças senta em um círculo, enquanto passam uma bola (representando a "batata quente") para a criança imediatamente ao lado. Em uma de suas variações, cada vez que uma criança possuir a bola em suas mãos, ela deve falar rapidamente uma coisa sobre o tema que se está tratando, por exemplo, deve falar um tipo de mamífero. Essa criança não deve demorar muito tempo para falar na sua vez, senão perde, pois a "batata" lhe "queima".

Observe que, se esse tipo de jogo (e tantos outros) possuísse uma versão on-line, deveria ter bastante controle de workflow, de tempo e de condições para se ganhar o jogo, etc., de forma muito parecida como ocorre com o Dominó. Esse controle é externo aos atores, produções e veículos envolvidos, ou seja, esse controle deve ser provido pelos **agentes**.

## 6.4 Avanços possibilitados pelo MX

Nesta seção, são analisadas as semelhanças e acréscimos que o MX apresenta ao MOrFEu.

Na Tabela 3, é apresentado um resumo das características que compõe o modelo conceitual MX, 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.

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

**Tabela 3. Resumo dos acréscimos do MX em relação ao 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 |

Qualitativamente, o MX amplia o escopo de ambiente virtuais que o MOrFEu podia representar (modelar). 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 alguns dos ambientes virtuais que estão servindo de objeto de estudo desta tese. Eles foram escolhidos possuírem características que representam bem o universo dos ambientes virtuais, ou por se tratarem de ambientes virtuais que são ferramentas de atividades cooperativas 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.

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

- Estados;
- 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 explicitada e descrita de maneira precisa.

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. Na Seção 0, 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 sejam flexíveis. Isso possibilita a edição e criação de funcionalidades do ambiente. Esse é um aspecto muito importante quando de trata de aplicações que estão em evolução constante, como ocorre com o Facebook, e para aplicações que precisam ser personalizadas em cada instância, 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.

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

| Ambientes Virtuais | MOrFEu | MX |
|---|---|---|
| **Fórum** | Sim | Sim |
| **Diário de Resolução de Problemas** | 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 | |
| **Jogo de Dominó** | Não | Sim |
| | Precisa de estados | |
| | Precisa de protocolo sofisticado | |
| **Blog** | Sim | Sim |
| **Enquete** | Sim, com artifícios | Sim |
| | Precisa de estados | |
| **Wiki** | Sim, incompleto | Sim |
| | Falta funcionalidades exercidas pelos agentes | |
| **Glossário** | Sim | Sim |
| **Chat** | Sim, incompleto | Sim |
| | Falta funcionalidades exercidas pelos agentes | |
| **Mural** | Sim | Sim |
| **CARTOLA** | Sim | Sim |
| **Controvérsia Acadêmica** | Sim, com artifícios | Sim |
| | Precisa de estados | |
| | Precisa de comportamento condicional das páginas | |
| **Júri Simulado** | Sim, com artifícios | Sim |
| | Precisa de estados | |
| **Projeto de Aprendizagem** | 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 | |
| **Apresentação de Artigos** | Sim, com artifícios | Sim |
| | Precisa de estados | |
| **Estudo Bíblico** | Sim | Sim |
| **Interpretador Online de Código-fonte** | Sim, com artifícios | Sim |
| | Precisa de funcionalidades exercidas pelos agentes | |
| **Bolão Copa do Mundo** | Sim, com artifícios | Sim |
| | Precisa de estados | |
| | Precisa de comportamento condicional das páginas | |

## 6.5 Conclusão do Capítulo

O modelo conceitual MX se baseia nas ideias do MOrFEu para criar um método de construção de ambientes virtuais, que parte desde sua modelagem até a descrição precisa de suas funcionalidades.

Com o MX, foi criada uma estrutura para representação dos elementos de um ambiente virtual (VCom) e foram mostrados exemplos de como descrevê-los precisamente. Foi definido também um esquema diagramático para a modelagem de VComs (caixas e setas representando UPIs e Relações respectivamente) e para a modelagem de interface, com a representação de páginas e templates e uma exibição condicional das mesmas baseado no estado atual e no papel interpretado pelo usuário.

O MX consegue modelar e descrever algumas aplicações que não eram possíveis de serem modeladas precisamente com o MOrFEu e dispõe de uma maneira mais precisa de descrever todas as funcionalidades de um ambiente virtual.

Como prova de conceito, foram modelados diversos ambientes virtuais conhecidos, desde aqueles possíveis de serem implementados com o MOrFEu, passando por aqueles que houveram limitações na sua modelagem, e chegando nos ambientes virtuais inusitados que necessitam de um alto grau de controle de protocolo de interação, como o Jogo de Dominó.

Isso evidencia que o grau de modelagem, descrição e detalhamento de ambientes virtuais web foi significativamente ampliado. Isso corrobora a Hipótese H1.1, o que sustenta de maneira firme o modelo apresentado.

No próximo capítulo, é apresentado como processar a linguagem de representação de VComs proposta pelo MX, no sentido de concretizar tais veículos, buscando evidenciar a Hipótese H2. Esse processamento é apresentado em termos de uma arquitetura e de algoritmos de funcionamento desses VCom derivados das especificações descritas neste capítulo.

Além disso, serão apresentadas provas de conceito de alterações em tempo de execução, buscando evidenciar a Hipótese H3.

# 7 Concretizando de Ambientes Virtuais MX

Neste capítulo, é apresentada uma arquitetura de ambientes virtuais construídos segundo o modelo conceitual MX e os algoritmos responsáveis pela execução desses ambientes virtuais. Também são apresentadas provas de conceitos do modelo conceitual MX, através de implementações de ambientes virtuais completos, ou partes deles. O objetivo desses protótipos é demonstrar a factibilidade da proposta e de como ela se adequa aos requisitos levantados.

As primeiras implementações foram feitas com a linguagem Prolog[43], que além de verificar a exequibilidade das definições, teve o intuito de refinar os detalhes do modelo conceitual. Depois disso, foi desenvolvido um framework que aplica os elementos do MX para criar ambientes virtuais na Web, utilizando tecnologias tradicionais na Web como a linguagem PHP e banco de dados MySQL.

Ambos os tipos de implementação seguem a arquitetura e os algoritmos apresentados. No entanto, os protótipos com Prolog não implementam a Interface. Enquanto isso, o protótipo web implementa uma versão mais simples do Protocolo, mas consegue construir ferramentas completamente funcionais.

Foram desenvolvidos três ambientes virtuais em Prolog, compartilhando um código de núcleo comum: Fórum, Debate de Teses e Jogo de Dominó. Além de um Protocolo complexo, o Jogo de Dominó possui uma característica peculiar: ele necessita que sejam executados procedimentos externos, na forma de gatilhos (*triggers*). Foram implementados dois gatilhos: gatilho de troca de estados depois que um jogador faz sua jogada; e um gatilho de contagem de pontos.

Os gatilhos foram utilizados como auxiliares que reduzem a carga cognitiva imposta aos jogadores. Num estágio final dessa análise, eles são agentes reativos

[43] Utilizando-se o ambiente de programação SWI-Prolog.

simples, que agem quando determinada condição é percebida no ambiente, realizando tarefas auxiliares e periféricas ao objetivo central, que é cumprido pelos usuários humanos do ambiente virtual. Podendo ser classificados também como Agentes de Time (ELLIS, 1998).

Os ambientes virtuais MX são aplicações web. São acessíveis através de URLs. Na Figura 32, é apresentada a arquitetura dos ambientes virtuais MX, os VComs. A forma como os usuários interagem com o ambiente virtual é através de URLs na Web.

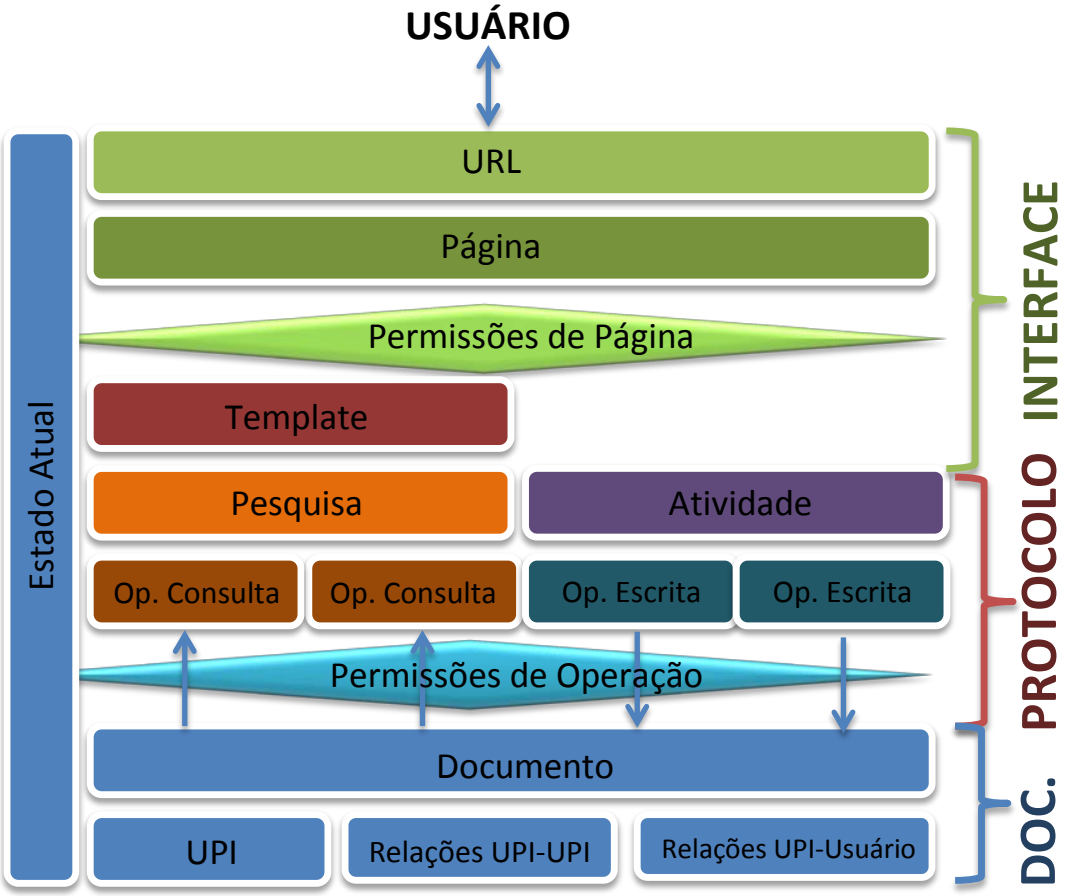


**Figura 32. Arquitetura de um VCom no MX.**

Essas URLs dão acessos à Interface do VCom (Definição 17). As URLs acessam diretamente as Páginas do VCom. Essas Páginas podem ser de dois tipos: de

Conteúdo ou de Atividades, de acordo com o que foi especificado nas Permissões de Páginas. Se nenhum Template ou Atividade for atribuído a uma Página, então essa Página é inacessível.

Além disso, as permissões de página definem quem pode ver essa página, de acordo com os estados atuais ativados do sistema e do papel exercido pelo usuário que a está acessando.

As Páginas de Conteúdo executam Templates específicos. O Algoritmo 1 especifica como é feita a execução de uma página desse tipo. Em linhas gerais, se o usuário que está solicitando a apresentação da Página puder acessá-la no estado atual, então executa o Template associado a ela e exibe o resultado para o usuário.

Um Template é um algoritmo que utiliza Pesquisas para apresentar um determinado conteúdo na Página que foi executado. Ou seja, os Templates proporcionam visualizações dos dados contidos no Documento.

As Páginas de Atividades possuem comportamento bastante parecido, como é apresentado no Algoritmo 2: Se o usuário que está solicitando a Página tiver permissão para fazê-lo no estado atual, então executa a atividade.

As Pesquisas são formadas por uma ou mais operações de consulta no Documento. E as Atividades por operações de escrita no Documento. Ambas as pesquisas e as Atividades são definidas pelo desenvolvedor do VCom.

As operações (de consulta ou escrita no Documento) são em um número limitado e bem definidas, e possuem associado mais um nível de permissões definidas no Protocolo (Definição 10 e Definição 11), pelo desenvolvedor.

Dessa forma, as operações de consulta são filtradas pelas permissões definidas. Ou seja, o resultado de uma operação é filtrado e é retornado apenas as UPIs ou Relações que é permitido ao usuário consultar naquele estado atual (Algoritmo 3).

Em uma Pesquisa, são realizadas operações de consulta. Essas Pesquisas são definidas pelo desenvolvedor. Como as operações de consulta podem retornar respostas vazias, uma Pesquisa também pode fazê-lo.

De forma semelhante, as operações de escrita também possuem um limitador, regido pelas permissões de operações de escrita, também definidas pelo desenvolvedor do VCom. Se uma operação de escrita pode ser executada, condicionada pelos parâmetros que são passados para ela, então ela é executada, senão ela simplesmente não é executada (Algoritmo 4).

Em uma Atividade, se alguma das operações não for possível de ser executada, então nenhuma das operações da Atividade é executada.

**Algoritmo 1. Função de executar uma página de conteúdo com um template.**

```
função executa_pagina_conteudo(usuario, pagina, parametros):
    υ ← permissoes_de_pagina_template()
    Σ′ ← estados_atuais()
    P_u ← papeis_usuario(usuario)
    para cada (s,r,p,t) ∈ υ faça:
        se (s ∈ Σ′) ∧ (r ∈ P_u) ∧ (p = pagina) então:
            retorna executa_template(t, parametros);
    retorna "Não permitido";
```

**Algoritmo 2. Função de executar uma página de atividade.**

```
função executa_pagina_atividade(usuario, pagina, parametros):
    υ ← permissoes_de_pagina_atividade()
    Σ′ ← estados_atuais()
    P_u ← papeis_usuario(usuario)
    para cada (s,r,p,a) ∈ υ faça:
        se (s ∈ Σ′) ∧ (r ∈ P_u) ∧ (p = pagina) então:
            executa_atividade(a, parametros)
            retorna Verdadeiro
    retorna Falso
```

**Algoritmo 3. Função de executar uma operação de consulta.**

**função** *executa_operacao_consulta*($usuario$, $operacao$, $parametros$):
    $\Sigma' \leftarrow$ *estados_atuais*()
    $P_u \leftarrow$ *papeis_usuario*(*usuario*)
    $\psi \leftarrow$ *roda_consulta*($operacao$, $parametros$)
    $Resposta \leftarrow \emptyset$
    **para cada** $u \in \psi$ **faça**:
        **se** *pode_consultar*($\Sigma'$, $P_u$, u) **então**:
            $Resposta \leftarrow Resposta \cup \{u\}$
    **retorna** *Resposta*

**Algoritmo 4. Função de executar uma operação de escrita.**

**função** *executa_operacao_escrita*($usuario$, $operacao$, $parametros$):
    $\Sigma' \leftarrow$ *estados_atuais*()
    $P_u \leftarrow$ *papeis_usuario*(*usuario*)
    **se** *pode_realizar_operacao*($\Sigma'$, $P_u$, $operacao$, $parametros$) **então:**
        *Documento.operacao_escrita*($operacao$, $parametros$)
        **retorna** Verdadeiro
    **senão:**
        **retorna** Falso

## 7.1 Protótipos em Prolog

O primeiro movimento para concretizar um ambiente virtual com o MX foi através da linguagem de programação de **Prolog**, interpretada com o software SWI-Prolog[44]. Os protótipos buscaram implementar elementos referentes ao Documento e Protocolo de Interação. Como a maioria dos conceitos do MX foi expressa em notação semi-formal, a tradução para o Prolog foi feita de maneira natural, resultando em protótipos funcionais, porém sem Interface.

Esses protótipos ajudaram diretamente na definição e descrição dos elementos do modelo conceitual MX. Durante o desenvolvimento deles foi possível refinar e ajustar os pontos desconexos apresentados nas primeiras versões do modelo.

[44] http://www.swi-prolog.org/

Foram escolhidos três ambientes virtuais para serem construídos em Prolog: Fórum, Debate de Teses e Jogo de Dominó. Essa escolha se deu pelos diferentes níveis de complexidade de tais ambientes: baixa, média e alta complexidade. Além disso, em conjunto esses ambientes virtuais cobrem a totalidade das características do MX, com exceção da Interface.

Um Fórum é um ambiente virtual comum e relativamente simples, segundo os conceitos do MX. O Debate de Teses necessita de um ambiente virtual mais complexo, com um Documento com maior variedade de tipos de UPIs e relações entre elas, com várias etapas e um Protocolo de mais complexo que um Fórum.

Com o protótipo do Debate de Teses foi também explorada a capacidade de modificação do ambiente em tempo de execução, com inclusão de funcionalidades no decorrer da execução do ambiente.

O Jogo de Dominó se destaca pelo alto grau de complexidade do Protocolo de Interação. As características desse protocolo se assemelham muito às dinâmicas de grupo usadas no contexto da educação, por isso o Jogo de Dominó foi um exemplo interessante de ser estudado.

Outra característica interessante do Jogo de Dominó é a necessidade de utilização de gatilhos (*triggers*) para o correto funcionamento do jogo. Isso foi feito implementando-se um agente reativo simples que faz o trabalho auxiliar e periférico ao jogo.

O código-fonte possui uma seção fixa, pré-definida, e uma parte dinâmica, ambas partem das bases de conhecimento no Prolog. Na parte fixa é implementada a arquitetura dos ambientes virtuais MX e os algoritmos apresentados no capítulo anterior. A parte dinâmica é escrita exclusivamente para cada ambiente virtual, nela são descritas as características do VCom, implementando suas funcionalidades.

Esses protótipos foram implementados utilizando três bases de conhecimento:

- Base de Conhecimento do Documento (**KB-Doc**): uma base de conhecimento que armazena dados sobre o Documento do VCom;

- Base de Conhecimento do framework do MX (**KB-MX**): uma base de conhecimento sobre VComs em geral, com implementações da arquitetura e dos algoritmos;
- Base de Conhecimento do VCom Específico (**KB-VCom**): possui o conhecimento específico sobre o VCom que se deseja implementar, a descrição de seu funcionamento, ou seja, a especificação do Protocolo do VCom.

### 7.1.1 Bases de Conhecimento Gerais

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. Essa base possui o conhecimento especificado pela **Definição 6**.

A base de conhecimento sobre VComs é estática, não necessitando de alterações, podendo ser reutilizada para a construção de qualquer VCom. São implementados os algoritmos que gerenciam o funcionamento do VCom de acordo com as especificações contidas na base de conhecimento do VCom que se está implementando.

#### 7.1.1.1 KB-Doc: Base de Conhecimento do Documento

Além dos dados (fatos) do Documento, a KB-Doc 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 usuário 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 previstos pelo framework do MX, no entanto, esses outros tipos de relações não foram necessárias para o funcionamento dos protótipos apresentados neste capítulo.

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 um **Documento inicial**, que é replicado para cada nova instância desse VCom. Os VCom Fórum e Debate de Teses não necessitam de UPIs previamente publicadas, portanto sua KB-Doc inicial é vazia. Já o Jogo de Dominó precisa que suas 28 peças estejam previamente presentes no Documento.

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

#### 7.1.1.2 KB-MX: Base de Conhecimento do Funcionamento do MX

Nessa base de conhecimento são implementadas as operações de consulta e escrita no Documento.

Um exemplo de operação de escrita no Documento é a operação de publicar uma UPI. Na Figura 33, 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. Depois dessas condições serem satisfeitas, um novo conhecimento é incluído na base de conhecimento e salvo no arquivo da KB-Doc, para assegurar a persistência desses dados. Esse trecho de programa é baseado no Algoritmo 4.

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

**Figura 33. Código em Prolog da operação de publicar uma UPI.**

A regra de permissão de publicar UPI `canPublishUPI` deve ser obrigatoriamente implementada na KB-VCom. Se isso não for feito, será assumido uma negação por falha. Ou seja, só são permitidas operação que foram explicitamente permitidas pelo desenvolvedor do ambiente.

Além da operação de publicar uma UPI, há também a operação 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, respectivamente fazendo *assert* dos fatos `relationUPI/4` e `relationUPIUser/4`.

A KB-MX também disponibiliza operações de consulta ao Documento. Na Figura 34, 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 às quais são permitidas 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-VCom, senão essa permissão é assumida como falsa. Esse trecho de programa foi baseado no Algoritmo 3.

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

**Figura 34. 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 na base de conhecimento KB-MX. Essa base de conhecimento possui seu código-fonte estático, o desenvolvedor de ambientes virtuais não deve alterá-la. As implementações das funcionalidades de um ambiente virtual específico são feitas nas bases de conhecimento do tipo KB-VCom.

A seguir são descritas as KB-VCom para cada um dos ambientes virtuais implementados.

### 7.1.2 Base de Conhecimento do VCom Específico

Na terceira base de conhecimento, são expressos o conhecimento sobre o VCom específico, o ambiente virtual que se deseja desenvolver. Nela são expressos, os papéis, os estados, as permissões, as atividades e as pesquisas do Protocolo.

#### 7.1.2.1 KB-VCom-Forum: Base de Conhecimento do Fórum

Nesta base de conhecimento foram implementadas as permissões, as atividades e as pesquisas do Fórum, conforme modelado na Seção 6.3.1.

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.

Além disso, o desenvolvedor de um VCom ainda teria que programar as páginas e templates, que não foram alvos de estudo nesse protótipo, onde buscou-se analisar propriedades do Documento e do Protocolo.

O Fórum possui apenas um estado e apenas um papel, o de participante do fórum.

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 35, é apresentado a regra `canQueryUPI/3`, que define as permissões para consulta de UPIs no Documento. Logo abaixo, em comentário, há a definição de outra permissão possível, que permite apenas consultar as UPIs do nível raiz, ou seja, aquelas que não possuem nenhuma outra UPI relacionada com elas.

```
%Qualquer um pode consultar o documento inteiro a qualquer momento
canQueryUPI(_State,_User,_UPI).

%TESTE: pode consultar só as UPIs raizes
%canQueryUPI(_State,User,UPI):-
%       queryUPIRelatedBy(UPI,User,R),
%       R=[].
%
```

**Figura 35. Código Prolog para as permissões de consulta do Documento de um Fórum.**

As permissões de escrita são apresentadas na Figura 36. São apresentadas duas regras. A primeira define que qualquer participante pode publicar UPIs com a label "`post`". E a segunda descreve que qualquer participante pode relacionar dois posts.

```
%Participantes podem publicar posts
canPublishUPI(_State,(_Content,Label,Publisher)):-
        Label = 'post',
        actor(participant,Publisher).

%Participantes podem relacionar post como resposta a outro post
canRelateUPIs(_State,(_ContentA,LabelA,_PublisherA),(_Content,LabelB,_PublisherB), _Label, Publisher) :-
        actor(participant,Publisher),
        LabelA = 'post',
        LabelB = 'post'.
```

**Figura 36. Código Prolog para permissões de escrita no Documento de um Fórum.**

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 37, são apresentadas duas atividades do Fórum. A atividade `publishRootPost` publica um post raiz, aquele que não possui nenhum "pai". Essa atividade apenas executa a operação `publishUPI`. Como as operações estão condicionadas às permissões definidas anteriormente, essa atividade somente será executada se as operações forem possíveis de serem executadas.

A atividade `publishPost` executa duas operações: primeiro publica uma UPI Post e em seguida associa essa UPI recém publicada a outra UPI, como "reposta", a outra UPI, passada por parâmetro.

```
publishRootPost(Content,Publisher):-
        publishUPI((Content,'post',Publisher)).

publishPost(Content,UPI1,Publisher):-
        publishUPI((Content,'post',Publisher)),
        relateUPIs((Content,'post',Publisher), UPI1, '', Publisher).
```

**Figura 37. Código Prolog de atividades do Fórum.**

São usadas duas pesquisas para montar a árvore de post de um Fórum, apresentadas na Figura 38. A pesquisa `queryRootPost` retorna todas as UPIs de post raiz, aquele que não possui nenhum "pai", ou seja, não há nenhuma outra UPI que faz relação a essa UPI. E a pesquisa `queryChildrenPost` retorna todos os "filhos" de um post passado por parâmetro, ou seja, todas as UPIs das quais a UPI passada por parâmetro possui uma relação.

```
queryRootPost(User,R):-
        queryUPI(User,AllUPIs),
        all(
                X,
                (
                        member(X,AllUPIs),
                        queryUPIRelatedBy(X,User,R1),
                        R1=[]
                ),
                R
            ).

queryChildrenPost(UPI,User,R):-
        queryUPIRelate(UPI,User,R).
```

**Figura 38. Código Prolog de pesquisas do Fórum.**

Na Figura 39, é apresentado um teste de uso do protótipo funcionando. Nele são apresentados exemplos de como operar sobre a base de conhecimento do Fórum feita em Prolog. Depois de alguns exemplos de uso de Atividades e Pesquisas, algumas

alterações são feitas no código-fonte da base de conhecimento, nas ações numeradas 14 a 20. Isso demonstra a capacidade de adaptação em tempo de execução do protótipo construído.

Depois de todas essas interações com o VCom, a base de conhecimento do Documento desse VCom ficou como apresentado na Figura 40.

```
1:  //Carregando o arquivo
    $ swipl -f forum_v1.pl

2:  //faz uma pesquisa
    ?- queryRootPost('Leo',R).
    R = [].

3:  //executa uma atividade e publica o primeiro primeiro post
    ?- publishRootPost('Primeiro Post','Leo').
    true.

4:  //faz uma pesquisa dos posts raízes
    ?- queryRootPost('Leo',R).
    R = [ ('Primeiro Post', post, 'Leo')].

5:  //faz outro post
    ?- publishRootPost('Segundo Post','Leo').
    true.

6:  //verifica se ele foi publicado
    ?- queryRootPost('Leo',R).
    R = [ ('Primeiro Post', post, 'Leo'), ('Segundo Post', post, 'Leo')].

7:  //publica uma resposta ao primeiro post
    ?- publishPost('Resposta1', ('Primeiro Post', post, 'Leo'), 'Nardo').
    true.

8:  //verifica que ele não foi postado como um post raiz
    ?- queryRootPost('Leo',R).
    R = [ ('Primeiro Post', post, 'Leo'), ('Segundo Post', post, 'Leo')].

9:  //ele foi postado como filho de outro post
    ?- queryChildrenPost(('Primeiro Post', post, 'Leo'), 'Nardo', R).
    R = [ ('Resposta1', post, 'Nardo')].

10: //publica uma resposta à primeira resposta
    ?- publishPost('Resposta1.1', ('Resposta1', post, 'Nardo'), 'Leo').
    true.

11: //veririca que ele foi publicado
    ?- queryChildrenPost(('Resposta1', post, 'Nardo'), 'Nardo', R).
    R = [ ('Resposta1.1', post, 'Leo')].

12: //não é permitido publicar uma mesma UPI.
    ?- publishPost('Resposta1.1', ('Resposta1', post, 'Nardo'), 'Leo').
    false.

13: //saindo
    ?- halt.

14: //Fazendo modificações em tempo de execução
    //retirando: Qualquer um pode consultar o documento inteiro a qualquer momento
    //%canQueryUPI(_State,_User,_UPI).
    //pode consultar só as UPIs raízes
    //canQueryUPI(_State,User,UPI):-
    //      queryUPIRelatedBy(UPI,User,R),
    //      R=[].

15: //recarregando
    $ swipl -f forum_v1.pl

16: ?- queryChildrenPost(('Primeiro Post', post, 'Leo'), 'Nardo', R).
    R = [].
    //nada é retornado

17: ?- halt.

18: //Fazendo modificações em tempo de execução (2)
    //retirnado a permissão: Participantes podem relacionar post como resposta a outro post
    //%canRelateUPIs(_State,(_ContentA,LabelA,_PublisherA),(_Content,LabelB,_PublisherB), _Label,
    Publisher) :-
    //%     actor(participant,Publisher),
    //%     LabelA = 'post',
    //%     LabelB = 'post'.
    //Somente pode publicar root posts.

19: //recarregando
    $ swipl -f forum_v1.pl

20: ?- publishPost('Resposta2', ('Primeiro Post', post, 'Leo'), 'Nardo').
    ERROR: relateUPIs/4: Undefined procedure: canRelateUPIs/5
```

**Figura 39. Teste de uso e alteração em tempo de execução do protótipo do Fórum.**

```
:- dynamic currentState/1.
currentState(s0).
:- dynamic publishedUPI/1.
publishedUPI(true).
publishedUPI(('Primeiro Post', post, 'Leo')).
publishedUPI(('Segundo Post', post, 'Leo')).
publishedUPI(('Resposta1', post, 'Nardo')).
publishedUPI(('Resposta1.1', post, 'Leo')).
publishedUPI(('Resposta2', post, 'Nardo')).
:- dynamic relationUPI/4.
relationUPI(true, true, true, true).
relationUPI(('Resposta1', post, 'Nardo'), ('Primeiro Post', post, 'Leo'), '', 'Nardo').
relationUPI(('Resposta1.1', post, 'Leo'), ('Resposta1', post, 'Nardo'), '', 'Leo').
```

**Figura 40. Resultado Final da KB-Doc depois das interações de teste.**

### 7.1.2.2 KB-VCom-ThesesDebate: Base de Conhecimento do DT

Nesta base de conhecimento foram implementadas as permissões, as atividades e a as pesquisas do Quadro de Discussão do Debate de Teses, conforme modelado na Seção 6.3.3.

Os papéis são dois: tutor e participante. Para esse protótipo, existe um tutor chamado '`leo`' e todos os outros usuários são participantes, conforme exibido na Figura 41.

```
%leo é um tutor
actor(tutor,leo).
%todo mundo é participant
actor(participant,_User).
```

**Figura 41. Definição de atores para o protótipo do Debate de Teses.**

Existe um conjunto de permissões para cada estado. No primeiro estado, apenas o tutor tem interação, ele insere quais as teses que serão discutidas. Na Figura 42, é apresentado o código que dá permissão para um usuário `tutor` publicar uma UPI do tipo '`thesis`'. Nesse estado ainda é permitido ao tutor trocar de estado para o estado seguinte.

```
%tutor pode publicar teses
canPublishUPI(s1,(_Content,Label,Publisher)):-
        Label = 'thesis',
        actor(tutor,Publisher).
```

**Figura 42. Permissão para o tutor publicar teses no primeiro estado do Debate de Teses.**

No segundo estado, as permissões mudam. Os participantes podem publicar argumentações para a tese, se ainda não o tiverem feito, ou seja, cada participante só

pode fazer uma única publicação para cada tese. Isso é descrito na Figura 43, com as duas permissões, uma permissão de publicação de UPI de argumentação e outra permissão de relação dessa UPI com outra UPI de tese. Pode-se observar que são utilizadas operações de consulta para se definir as permissões, como por exemplo, a operação `queryUPIRelate_LabelPublisher`, que faz a consulta de UPIs relacionadas com alguma UPI passada por parâmetro, filtrando também por rótulo e usuário publicador, ambos também passados por parâmetro.

No terceiro estado são publicados os comentários das argumentações. Nesse momento os participantes devem comentar os comentários feitos por outros participantes. Três restrições são aplicadas: um participante não pode publicar sua própria argumentação; um participante não pode comentar mais de uma vez uma argumentação; e só pode comentar argumentações que tenham um ou nenhum comentário, pois uma argumentação não pode ter mais de dois comentários. Essas permissões estão descritas na Figura 44.

No quarto estado, os participantes fazem réplicas aos comentários feitos às suas argumentações. Na Figura 45, é descrita essa permissão em código Prolog.

No quinto e último estado, os participantes podem publicar revisões de suas argumentações iniciais, como descrito nas permissões listadas na Figura 46.

E essas formam o conjunto de permissões do Debate de Teses, um pequeno conjunto para cada estado. E a troca de estado é tarefa do tutor.

```
%participant pode publicar argumentacao
canPublishUPI(s2,(_Content,Label,Publisher)):-
        Label = 'argumentation',
        actor(participant,Publisher).

canRelateUPIs(s2,(_ContentA,LabelA,_PublisherA),(ContentB,LabelB,PublisherB), _Label, Publisher) :-
        actor(participant,Publisher),
        LabelA = 'argumentation',
        LabelB = 'thesis',
        %não há argumentações do Publisher relacionadas com a tese
        queryUPIRelate_LabelPublisher((ContentB,LabelB,PublisherB),'argumentation',Publisher,Publisher,R),
        R=[].
```

**Figura 43. Permissões de publicar argumentações no segundo estado do Debate de Teses.**

```
%participant pode publicar comentarios de argumentacoes
canPublishUPI(s3,(_Content,Label,Publisher)):-
        Label = 'comment',
        actor(participant,Publisher).

canRelateUPIs(s3,(_ContentA,LabelA,_PublisherA),(ContentB,LabelB,PublisherB), _Label, Publisher) :-
        actor(participant,Publisher),
        LabelA = 'comment',
        LabelB = 'argumentation',
        %Um participante não pode comentar a própria argumentacao
        PublisherB \= Publisher,
        %Não há comentários do Publisher relacionadas com essa argumentação
        queryUPIRelate_LabelPublisher
((ContentB,LabelB,PublisherB),'comment',Publisher,Publisher,R),
        R=[],
        %Há apenas um ou nenhum comentário relacionado com essa argumentacao (É permitido apenas 2
comentários por argumentação)
        queryUPIRelate_Label((ContentB,LabelB,PublisherB),'comment',Publisher,R2),
        (R2=[];R2=[_Head]).
```

**Figura 44. Permissões de publicação de comentários para o terceiro estado do Debate de teses.**

```
%participant pode publicar replicas a comentários feitos feitos por ele
canPublishUPI(s4,(_Content,Label,Publisher)):-
        Label = 'replica',
        actor(participant,Publisher).

canRelateUPIs(s4,(_ContentA,LabelA,_PublisherA),(ContentB,LabelB,PublisherB), _Label, Publisher) :-
        actor(participant,Publisher),
        LabelA = 'replica',
        LabelB = 'comment',
        %um participant não pode fazer uma replica de um próprio comentário
        PublisherB \= Publisher,
        %o comentário deve ser para uma argumentação do Publisher
        queryUPIRelatedBy_Label((ContentB,LabelB,PublisherB),'argumentation',Publisher,
[(_ContentR,'argumentation',Publisher)]),
        %deve haver somente uma replica para cada comentário
        queryUPIRelate_Label((ContentB,LabelB,PublisherB),'replica',Publisher,[]).
```

**Figura 45. Permissões de publicação de réplicas para o quarto estado do Debate de Teses.**

```
%participante pode publicar revisões para seu posicionamento inicial
canPublishUPI(s5,(_Content,Label,Publisher)):-
        Label = 'revision',
        actor(participant,Publisher).

canRelateUPIs(s5,(_ContentA,LabelA,Publisher),(ContentB,LabelB,PublisherB), _Label, Publisher) :-
        actor(participant,Publisher),
        LabelA = 'revision',
        LabelB = 'argumentation',
        %um participant faz uma revisão para sua argumentação
        PublisherB = Publisher,
        %só pode haver somente uma revisão para cada argumentação
        queryUPIRelate_Label((ContentB,LabelB,PublisherB),'argumentation',Publisher,[]).
```

**Figura 46. Permissões de publicação de revisões do posicionamento inicial para o quinto estado do Debate de Teses.**

As Atividades são aquelas que serão executadas pelos usuários, publicar teses, argumentações, comentários, réplicas e revisões, como codificado na Figura 47. Elas são compostas de uma sequência de operações de escrita, tais como `publishUPI`, que publica uma nova UPI, e `relateUPIs`, que relaciona duas UPIs já publicadas.

As Pesquisas são funções utilizadas para consultar o Documento e montar o template. Observe que nas permissões anteriormente mencionadas não haviam permissões de leitura do Documento, mas foi codificada uma permissão que permite que qualquer um possa consultar todo o Documento. Na Figura 48, são exibidos os códigos das pesquisas para o Debate de Teses.

Na Figura 49, é apresentado um teste de uso do protótipo funcionando. Nele são apresentados exemplos de utilização seguindo todos os estados.

Nas ações numeradas 14 a 19 são realizadas e testadas alterações em tempo de execução. É incluída a funcionalidade de escrever Notas às produções dos participantes, com intuito de tornar o Tutor mais participante da atividade. Essa funcionalidade é descrita no código da Figura 50, que descreve as permissões e a atividade para se fazer Notas. Essas permissões definem que o tutor pode fazer notas a argumentações, comentários, revisões, réplicas ou a outras notas. E que participantes podem responder notas com outras notas.

```
publishThesis(Content,Publisher):-
        publishUPI((Content,'thesis',Publisher)).

publishArgumentation(Content,Position,UPI1,Publisher):- %Position = concordo ou discordo
        publishUPI((Content,'argumentation',Publisher)),
        relateUPIs((Content,'argumentation',Publisher), UPI1, Position, Publisher).

publishComment(Content,UPI1,Publisher):-
        publishUPI((Content,'comment',Publisher)),
        relateUPIs((Content,'comment',Publisher), UPI1, '', Publisher).

publishReplica(Content,UPI1,Publisher):-
        publishUPI((Content,'replica',Publisher)),
        relateUPIs((Content,'replica',Publisher), UPI1, '', Publisher).

publishRevision(Content,UPI1,Publisher):-
        publishUPI((Content,'revision',Publisher)),
        relateUPIs((Content,'revision',Publisher), UPI1, '', Publisher).
```

**Figura 47. Atividades do Debate de Teses em Prolog.**

```
queryThesis(User,R):-
        queryUPI_Label(User,'thesis',R).

queryArgumentations(User,Publisher,UPIThesis,R):- %somente a argumentação para um usuário
específico para um tese
        queryUPIRelate_LabelPublisher(UPIThesis,'argumentation',Publisher,User,Argumentations),
        all(%inclui qualquer tipo de relação
                (Content,Label,Publisher,RelationLabel),
                (
                        member((Content,Label,Publisher),Argumentations),
                        queryRelationsUPIs((Content,Label,Publisher),UPIThesis,User,
[(RelationLabel,Publisher)|_])
                ),
                R
            ).

queryComments(User,UPIArgumentation,R):- %todos os comentários de uma argumentação
        queryUPIRelate_Label(UPIArgumentation,'comment',User,R).

queryReplica(User,UPIComment,R):- %todas as replicas de um comentário
        queryUPIRelate_Label(UPIComment,'replica',User,R).

queryRevision(User,UPIArgumentation,R):- %todas as revisões de uma argumentação
        queryUPIRelate_Label(UPIArgumentation,'revision',User,R).
```

**Figura 48. Pesquisas do Debate de Teses em Prolog.**

```
1:  $ swipl -f dt_v1.pl
2:  ?- publishThesis(t1,tutor).
    false.
    //tutor não é um usuário, é um papel
3:  //publica duas teses, leo é um tutor.
    ?- publishThesis(t1,leo).
    true.
4:  ?- publishThesis(t2,leo).
    true.
5:  ?- publishArgumentation(a1,(t1, thesis, leo),'concordo',user1).
    false.
    //não pode fazer nada nesse estado.
6:  ?- publishState(state2,leo).
    true.
    //muda de estado
7:  ?- publishArgumentation(a1,(t1, thesis, leo),'concordo',user1).
    false.
    //Só muda de estado realmente quando carrega de novo (limitacao técnica)
8:  ?- halt.
    $ swipl -q -f dt_v1.pl
9:  //os dois usuários publicam suas argumentações das duas teses
    ?- publishArgumentation(a1,(t1, thesis, leo),'concordo',user1).
    true.
10: ?- publishArgumentation(a2,(t2, thesis, leo),'discordo',user1).
    true.
11: ?- publishArgumentation(a3,(t1, thesis, leo),'concordo',user2).
    true.
12: ?- publishArgumentation(a4,(t2, thesis, leo),'discordo',user2).
    true.
13: ?- halt.
14: //teste de alteração em tempo de execução
    //arquivo alterado para adicionar as funcionalidade de notas sobre os argumentos (notes)
    $ swipl -q -f dt_v1.pl
15: //o tutor leo resolve fazer algumas considerações sobre a argumentação a1
    ?- publishNote(n1, (a1, argumentation, user1), leo).
    true.
16: ?- publishNote(n2, (n1, note, leo), user1).
    false.
    //A definição feita no no código fonte estava errada.
    //corrigido, faltava dar permissão para publicar note. Tinha dado só para relate.
17: ?- halt.
    $ swipl -q -f dt_v1.pl
18: ?- publishNote(n2, (n1, note, leo), user1).
    true.
    //agora funcionou
19: //o user1 retruca
    ?- publishNote(n3, (n2, note, user1), leo).
    true.
20: //trocando de estado
    ?- publishState(state3,leo).
    true.
21: ?- halt.
    $ swipl -q -f dt_v1.pl
22: //continuando as interações, publicando comentários
    ?- publishComment(c1,(a3, argumentation, user2), user1).
    true.
23: ?- publishComment(c2,(a4, argumentation, user2), user1).
    true.
24: ?- publishComment(c3,(a1, argumentation, user1), user2).
    true.
25: ?- publishComment(c4,(a2, argumentation, user1), user2).
    true.
```

```
26: ?- publishState(state4,leo).
    true.
27: $ swipl -q -f dt_v1.pl
28: //pode-se verificar o que pode ser feito para cada pessoa:
    //o que o tutor leo pode fazer?
    ?- currentState(S), canPublishUPI(S,(_C,Label,leo)).
    S = s3,
    Label = note ;
    S = s3,
    Label = note ;
    S = s3,
    Label = comment ;
    S = s3,
    Label = state4.
29: //o que o user1 pode fazer?
    ?- currentState(S), canPublishUPI(S,(_C,Label,user1)).
    S = s3,
    Label = note ;
    S = s3,
    Label = comment ;
    false.
30: //Mesmo 'leo' sendo tutor, por definição ele pode ser um participant:actor(participant,_User).
    //por isso ele pode adicionar comentários
    ?- actor(R,leo).
    R = tutor ;
    R = participant.
31: ?- publishReplica(r1,(c3, comment, user2),user1).
    true.
32: ?- publishReplica(r2,(c4, comment, user2),user1).
    true.
33: ?- publishReplica(r3,(c1, comment, user1),user2).
    true.
34: ?- publishReplica(r4,(c2, comment, user1),user2).
    true.
35: ?- publishState(state5,leo).
    true.
36: ?- halt.
    $ swipl -q -f dt_v1.pl
37: ?- publishRevision(rev1,(a1, argumentation, user1),user1).
    true.
38: ?- publishRevision(rev2,(a2, argumentation, user1),user1).
    true.
39: ?- publishRevision(rev3,(a3, argumentation, user2),user2).
    true.
40: ?- publishRevision(rev4,(a4, argumentation, user2),user2).
    true.
41: //Fazendo umas pesquisas
    ?- queryThesis(leo,R).
    R = [ (t1, thesis, leo), (t2, thesis, leo)].
42: ?- queryArgumentations(leo,user1,(t1, thesis, leo),R).
    R = [ (a1, argumentation, user1, concordo)].
43: ?- queryComments(leo,(a1, argumentation, user1), R).
    R = [ (c3, comment, user2)].
44: ?- queryReplica(leo,(c3, comment, user2),R).
    R = [ (r1, replica, user1)].
45: ?- queryRevision(leo,(a1, argumentation, user1),R).
    R = [ (rev1, revision, user1)].
46: //Todas as argumentações de uma tese.
    ?- queryArgumentations(leo,_User,(t1, thesis, leo),R).
    R = [ (a1, argumentation, user1, concordo), (a3, argumentation, user2, concordo)].
```

**Figura 49. Teste de uso e alteração em tempo de execução do protótipo do Debate de Teses.**

```
%tutor pode publicar notas em qualquer estado
canPublishUPI(_S,(_Content,Label,Publisher)):-
        Label = 'note',
        actor(tutor,Publisher).

canRelateUPIs(_S,(_ContentA,LabelA,_PublisherA),(_ContentB,LabelB,_PublisherB), _Label,
Publisher) :-
        actor(tutor,Publisher),
        LabelA = 'note',
        member(LabelB , ['argumentation', 'comment', 'revision', 'replica', 'note'] ).

%participantes podem responder notas com notas
canPublishUPI(_S,(_Content,Label,Publisher)):-
        Label = 'note',
        actor(participant,Publisher).
canRelateUPIs(_S,(_ContentA,LabelA,_PublisherA),(_ContentB,LabelB,_PublisherB), _Label,
Publisher) :-
        actor(participant,Publisher),
        LabelA = 'note',
        LabelB = 'note'.

%ATIVIDADE
publishNote(Content,UPI1,Publisher):-
        publishUPI((Content,'note',Publisher)),
        relateUPIs((Content,'note',Publisher), UPI1, '', Publisher).
```

**Figura 50. Alterações feitas para acrescentar a funcionalidade de Notas em tempo de execução ao Debate de Teses.**

#### 7.1.2.3 KB-VCom-Dominoes: Base de Conhecimento do Jogo de Dominó

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.

Os fatos sobre os papéis são representados apenas com uma lista de papéis, como apresentado na Figura 51.

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

**Figura 51. Código em Prolog para a representação de papéis do Jogo de Dominó.**

A associação entre usuários e os papéis produz a relação de atores, que devem ser listados da mesma forma. Na Figura 52, 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 52. Código em Prolog para a definição de atores do Jogo de Dominó.**

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.

Na Figura 33 e na Figura 34, podem-se observar duas verificações de permissões na KB-MX: `canPublishUPI` e `canQueryUPI`. As duas operações somente são realizadas se essas permissões forem verdadeiras na base de conhecimento, no caso a KB-VCom-Dominoes.

Na Figura 53, 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 53. Código Prolog que define que todos os usuários podem a qualquer momento consultar todas as 28 pedras do Jogo de Dominó.**

```
%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 54. Código Prolog de permissão de jogar uma pedra na mesa no Jogo de Dominó.**

Na Figura 54, 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.

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.

Na Figura 55, 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 55. Código Prolog de uma Pesquisa do Jogo de Dominó.**

Uma Atividade é uma sequência de operações de alteração do Documento. Na Figura 56, 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 encaixada 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 56. Código Prolog de uma Atividade do Jogo de Dominó.**

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 57, 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.

Há a possibilidade de implementar outros tipos de gatilhos. Na Figura 58, 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 58, 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.

```
%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 57. Código Prolog para um gatilho de troca de estado no Jogo de Dominó.**

```
%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 58. Código Prolog para um gatilho de contagem de pontos do Jogo de Dominó.**

## 7.2 Um Framework de Desenvolvimento Web para Ambientes Virtuais

Nesta seção será apresentado 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. Apenas algumas características de permissões em nível de operações exploradas, tendo elas sido exploradas nos protótipos em Prolog.

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-fonte base 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. 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. 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 partes **fixas**: no Documento, as operações de leitura e escrita, e partes que devem ser **desenvolvidas** pelo criador do VCom:

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

### 7.2.1 Protótipo Web do Diário de Resolução de Problemas

A seguir, 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 (SERRES e BASSO, 2009). Esse VCom possui um Documento simples, um Protocolo também simples, mas que podem 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.

A partir do "esqueleto" do framework, o desenvolvedor deve então preenchê-lo com as definições de seu VCom. O desenvolvedor deve escrever as funções de Atividades e Pesquisas sobre o Documento. Nas Figura 59 e a Figura 60 são apresentados 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 de dados exibido na Figura 22 não é explicitamente representado. As atividades são as responsáveis por gerar as relações entre as UPIs. Nas Figura 59 e Figura 60 são apresentados trechos de códigos para a Atividade de publicar comentários e a Pesquisa de um diário.

Após isso, o desenvolvedor deve escrever a função de retornar o papel de um dado usuário. O código apresentado na Figura 61 é bem simplista: o usuário que possui o *username* igual a "professor" tem o papel de professor, senão 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 de dados.

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

Os Actions do Controller no framework Yii fazem a funcionalidade de Páginas. Observa-se na Figura 62 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 63, é apresentada uma página de outro tipo, ela executa atividades e depois redireciona para outra página.

Na Figura 64, 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
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 59. 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 60. Código da pesquisa por um diário de um usuário.**

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

**Figura 61. Código da função de papel de um usuário.**

```
//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 62. 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 63. Código de uma Página de execução de Atividade.**

```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 64. Código do Template "todosDiarios".**

Pode-se observar que a estrutura desse protótipo segue as orientações da arquitetura proposta na Figura 32. Depois de o usuário acessar uma URL, é executado uma página (Figura 62). Dependendo das permissões de página, é escolhido um template para ser exibido (Figura 64). Nesse template, são realizadas pesquisas (Figura 60). As pesquisas são compostas por operações de consulta ao Documento, que recuperam UPIs e relações do banco de dados.

Na Figura 65, é apresentada a página de um participante do Diário Virtual, chamado de "estudante1". Seguindo as orientações do diagrama da Figura 23, para um participante comum é mostrado apenas o seu diário, nesse estado inicial, e para um professor é mostrada uma lista de todos os diários.

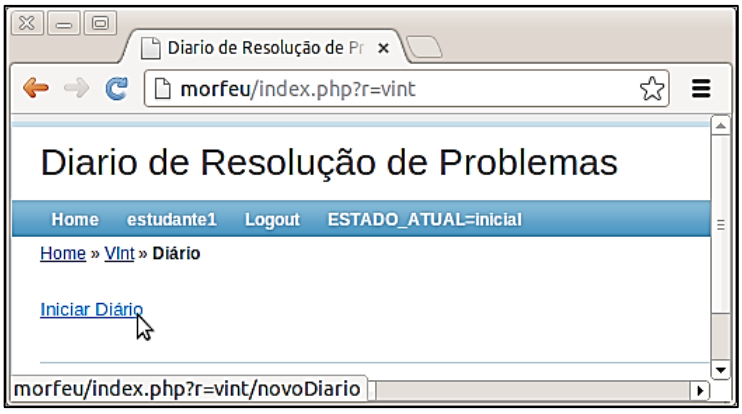


**Figura 65. Página Inicial de um participante do Diário Virtual.**

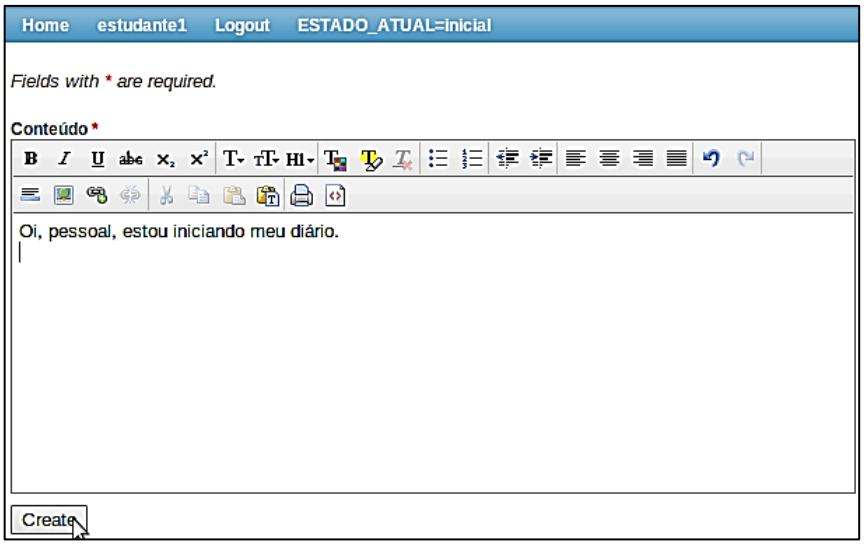


**Figura 66. Editando uma UPI.**

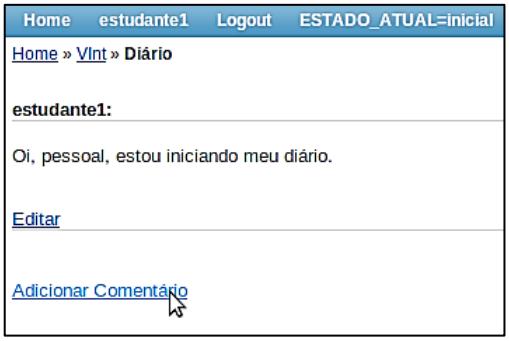


**Figura 67. Página Inicial de um participante do Diário Virtual, depois de escrever no seu diário.**

O usuário ainda não publicou nada, por isso o link "Iniciar Diário". Ao clicar nesse link, ele é direcionado para a página exibida na Figura 66, onde edita o conteúdo de uma UPI e a publica no VCom, resultando a página exibida na Figura 67. Nesse momento, se o "estudante1" clicar no link "adicionar comentário" receberá a mensagem "operação não permitida".

Ainda no estado inicial, a página que um "professor" pode ver é diferente de um participante comum. Na Figura 68, é apresentada a página de todos os diários para um professor.

Ao mudar para o estado de "revisões", conforme orientação do diagrama da Figura 24, qualquer usuário pode ver todos os diários. Assim, a página inicial do "estudante1" muda quando o estado é mudado para o estado de "revisões", conforme pode ser observado na Figura 69.

Na Figura 70, é apresentada a tela do “estudante1” visitando o diário do “estudante2” e fazendo um comentário.

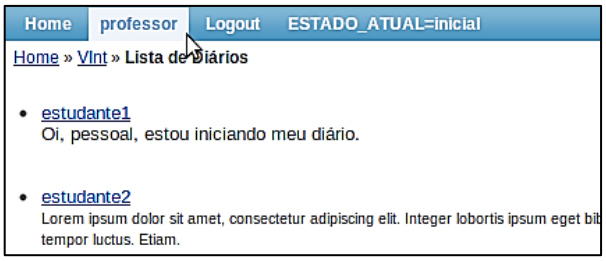


**Figura 68. Página Inicial de um professor no Diário Virtual.**

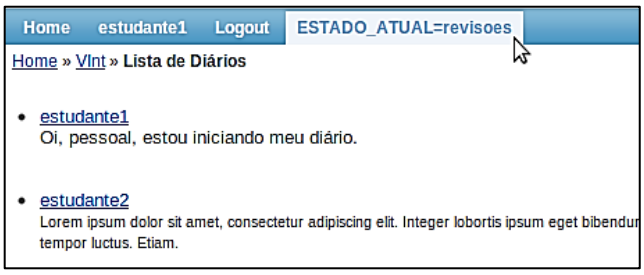


**Figura 69. Página inicial de um participante do Diário Virtual, na etapa de revisões.**


**Figura 70. Comentário num Diário Virtual.**

### 7.2.2 Modificações em Tempo de Execução

Foram feitas algumas simulações de alterações em tempo de execução. Foi acrescentada mais uma etapa, a etapa de conclusão. As alterações no protótipo foram realizadas em tempo de execução, sem perder as interações já realizadas.

Nessa nova etapa, os autores de seus diários, depois de realizados os comentários por seus pares, devem fazer um texto conclusivo da atividade. Essa etapa não é

contemplada pelo relato inicial de Serres e Basso (2009), ela é simplesmente ilustrativa, e baseada na etapa final do Debate de Teses.

O primeiro passo é modificar as modelagens realizadas, para dar suporte ao desenvolvimento das alterações. A primeira modificação foi no esquema do documento, como pode ser visto na Figura 71, foi adicionado um novo tipo de UPI e uma relação com a UPI Post, chamada de "Conclusão".

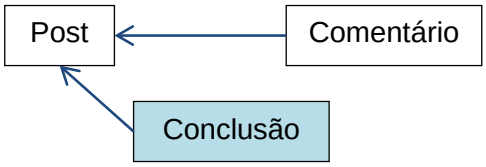


**Figura 71. Diagrama de Documento do Diário de Resolução de Problema alterado, adicionando mais um tipo de UPI, "Conlusão".**

Uma modificação deve ser feita na interface, no Template de um Diário, adicionando um link para a página de atividade de criar uma "Conclusão". As permissões em nível de página são muito parecidas com a do estado inicial, mas somente é permitido, nesse estado, executar a atividade "adicionar conclusão", que deve ser desenvolvida: criar uma nova UPI e relacioná-la com o post.

Além disso, depois da conclusão inserida é preciso exibi-la, na página do Diário. Para isso é preciso também criar uma pesquisa, que consulta o documento e recupera a conclusão relacionada com o post do diário.

Na Figura 72, é apresentada uma tela com um exemplo de resultado final de um Diário, como uma Conclusão sendo exibida.

Dessa forma, mostra-se a capacidade do protótipo de acomodar alterações em tempo de execução. Essas modificações foram feitas de maneira muito semelhante às modificações em tempo de execução realizadas nos protótipos em Prolog. Sendo assim mais uma prova de conceito de flexibilidade dos VComs construídos segundo o modelo MX.


**Figura 72. Conclusão em um Diário Virtual.**

### 7.2.3 Características Futuras

Esse protótipo não contempla as permissões em nível de operações, somente as permissões em nível de páginas, templates e atividades. Ou seja, o Algoritmo 3 e o Algoritmo 4 não foram completamente implementados, não foram feitas as verificações de permissões. Assim, para implementar tais algoritmos é necessário implementar as funções de "pode_consultar" e "pode_realizar_operacao".

Como é visto no Algoritmo 3, e como foi visto nos protótipos em Prolog, a função "pode_consultar" é uma função que recebe como parâmetro três elementos: um estado, um papel do usuário que está acessando, e uma UPI publicada. Essa função deve então retornar verdadeiro se pode consultar, e falso se não pode consultar.

As permissões de realizar operações são mais específicas por operação. Recebem qual a operação que se está querendo executar e seus parâmetros, assim como o estado atual e o papel do usuário que quer realizar a operação. Essa função deve então decidir se o usuário pode ou não realizar tal operação.

Portanto, as permissões em nível de operações, tanto de leitura quanto de escrita, são possíveis de serem implementadas, assim como essas foram nos protótipos em Prolog.

## 7.3 Conclusão do Capítulo

Neste capítulo, foi mostrado que a proposta do MX é implementável. Os protótipos desenvolvidos possuem, em conjunto, a totalidade das características elencadas pelo MX. Dessa forma, ficou evidenciada a adequabilidade do MX para desenvolvimento de ambientes virtuais.

Os protótipos em Prolog eram parcialmente funcionais, cumprindo o propósito de construção computacional dessas ferramentas, porém não tinham interface para o usuário. Assim, foi desenvolvido um framework de desenvolvimento web que implementava os elementos do MX. Com esse framework foi desenvolvido um protótipo do Diário de Resolução de Problemas. Tal protótipo é funcional e pronto para utilização por usuário final.

Assim, demonstra-se a factibilidade da proposta do MX em construir ambientes virtuais de suporte a atividades pedagógicas. Explorando com alguns exemplos os elementos descritos no modelo conceitual MX.

Observando os códigos de ambos os tipos de implementações, nota-se que a quantidade de código é bem reduzida, devido ao fato de ser necessário apenas descrever as características do ambiente virtual que se deseja implementar, fazendo um tipo de programação declarativa. Os Templates, no entanto, possuem quantidade de código similar a formas tradicionais de desenvolvimento, por se tratar da construção do HTML das páginas.

Essas implementações demonstram que o MX é um modelo conceitual implementável, que sua arquitetura e algoritmos podem se tornar concretizados em ambientes virtuais.

Além disso, foram apresentados estudo de casos de modificações de funcionalidades em tempo de execução. Isso prova o conceito de que o MX suporta esse tipo de característica nos ambientes construídos segundo seus conceitos.

Os algoritmos e a arquitetura também foram implementados da maneira proposta, em ambos os tipos de protótipo. Isso demonstra uma proximidade muito grande entre os protótipos Prolog e o Web. Um corrobora as observações de outro. Ou seja, os protótipos foram feitos em Prolog apenas por seu desenvolvimento ser mais rápido.

O protótipo Web demonstrou que é possível implementar ambientes virtuais MX com tecnologias tradicionais de desenvolvimento Web: Apache+MySQL+PHP (AMP).

Vale ressaltar que um dos requisitos do MOrFEu era o uso de uma base de dados que possuísse esquema flexível, como um banco NoSQL, para dar suporte adequado a mudanças em tempo de execução (SANTOS, CASTRO e MENEZES, 2010), elemento que foi removido pela proposta do MX. Anulando assim a Hipótese H4.

A implementação dos ambientes virtuais que, em conjunto, possuem a totalidade das características do MX, corrobora a Hipótese H2.

Além disso, as alterações em tempo de execução em todos os ambientes virtuais implementados evidencia a Hipótese H3.

Dessa forma, foi possível mostrar a factibilidade da proposta MX com todas as suas características.

# 8 Conclusão

Neste capítulo, são discutidos os elementos centrais da solução proposta ao problema apontado, bem como as evidências que a estrutura conceitual e a plataforma de modelagem desenvolvida é adequada aos propósitos, apontando possíveis desdobramentos e trabalhos futuros.

Ainda são ressaltados os elementos que evidenciam a validade da tese proposta.

## 8.1 Problema

O problema tratado nesta tese foi a criação de ambientes virtuais web, de acordo com a necessidade de seus usuários, dotados de flexibilidade para modificações em tempo de execução, em particular no âmbito de suporte a atividades pedagógicas cooperativas.

## 8.2 Solução proposta

Diferente das propostas tradicionais, pensamos um ambiente virtual como o guardião de um artefato composto pelas interações de seus usuários. O conjunto dessas interações forma um documento, que é gerenciado pelo ambiente virtual. A partir dessa concepção, foi possível conceber um modelo conceitual para modelagem de ambientes através de uma descrição precisa dos elementos e funcionalidades de tais ferramentas, e implementá-los através dessas descrições.

Através do MX, os ambientes virtuais puderam ser descritos com elementos simples: UPI, Documento, Publicação, Relações, Permissões e Templates. A organização e a combinação de tais elementos caracteriza um ambiente virtual

específico. De forma que é possível descrever vários dos ambientes virtuais conhecidos através desses elementos elencados no modelo conceitual proposto como solução.

Com os elementos elencados pelo MX é possível modelar ambientes virtuais e descrevê-los precisamente, estabelecendo uma "linguagem franca" para descrição, comunicação e comparação de funcionalidades, mesmo quando os ambientes não forem implementados numa mesma plataforma.

A descrição precisa das permissões e funcionalidades de um ambiente pode facilitar o processamento sobre o conhecimento da construção de ambientes virtuais. Isso sugere que seja possível a criação de agentes inteligentes que auxiliem os desenvolvedores na tarefa de desenvolver ambientes virtuais, possível tema para continuidade da pesquisa.

A descrição possui uma tradução próxima às linguagens de programação, onde desenvolver um ambiente virtual passa a ser "descrever suas funcionalidades". No entanto, os elementos de interface ainda precisam ser programados de maneira semelhante à tradicional. Os ambientes virtuais construídos com o MX podem ser modificados em tempo de execução. Modificando sua descrição o ambiente passa a funcionar de maneira diferente, sem o prejuízo a dados já inclusos.

Outro ponto interessante acerca da descrição das funcionalidades do ambiente é a não dependência a um esquema de dados fixo e pré-concebido. Como o MX pressupõe que todos os ambientes virtuais são compostos pelos mesmos tipos de elementos, o esquema dos dados é preparado para dar suporte a esses elementos, como UPIs e relações entre UPIs. Ou seja, o esquema de dados é sempre o mesmo, não importando qual o ambiente virtual. O que é específico para cada ambiente são quais as permissões de criação e associação dos seus elementos (por exemplo, permissões sobre a quem é dada a possibilidade de publicar posts em um fórum, e como um post pode se relacionar com outro post).

## 8.3 Evidências de adequação ao propósito

O modelo proposto tem três etapas distintas para o desenvolvimento de ambientes virtuais: modelagem, especificação e implementação (concretização). Assim, buscou-se coletar evidências para cada uma dessas etapas.

Conforme descrito no Capítulo 5, nossa coleta de evidências iniciou com a construção de um framework para o MOrFEu, onde e a modelagem e implementação de ambientes virtuais corroborou sua relevância.

A prospecção de experiências a partir de situação real de uso também foi crucial para identificar e refletir sobre melhoramentos àquela proposta e caracterizar uma plataforma para concepção e modificação de ambientes virtuais incorporando tais melhoramentos (o MX).

Foi feita a modelagem de 19 ambientes virtuais com o MX, descritas no Capítulo 6 e no Apêndice A, a saber: Fórum, Diário Virtual de Resolução de Problemas de Matemática; Debate de teses; Jogo de Dominó; Blog; Enquete; Wiki; Glossário; Chat; Mural; CARTOLA; Controvérsia Acadêmica; Júri Simulado; Projeto de Aprendizagem; Facebook; Bolão de Apostas da Copa do Mundo; Apresentação de Artigos; Estudo Bíblico; e Interpretador de Código Haskell.

Esses ambientes incluem ferramentas tradicionais de comunicação, como o Fórum, Blog, Chat e Wiki; outros são representantes da classe de ambientes que este trabalho se propôs a explorar inicialmente (ferramentas de apoio a aprendizagem cooperativa através da Web), como foi o caso do Diário Virtual, Debate de Teses, Júri Simulado, e Projeto de Aprendizagem; outros fogem também desse escopo, mas têm características interessantes, como o Jogo de Dominó. Todos esses foram modelados buscando compor um cenário abrangente de ambientes virtuais. Suas modelagens serviram como provas de conceito da proposta.

Alguns desses ambientes foram escolhidos para serem implementados, utilizando abordagens distintas porém complementares. Primeiramente em Prolog, fazendo uso

de uma descrição alinhada à lógica proposicional, visando principalmente à análise da especificação das permissões e funcionalidades dos ambientes virtuais. Em seguida foram utilizadas ferramentas de desenvolvimento web tradicionais: PHP e MySQL. O protótipo implementado para a Web teve o objetivo de analisar a construção das interfaces. Os protótipos foram submetidos a alterações em tempo de execução, evidenciando essa característica nos ambientes construídos segundo o MX.

## 8.4 Evidências de Validade da Tese

A tese deste trabalho era a de que os elementos conceituais apresentados pelo MOrFEu possibilitam a concepção e desenvolvimento de ambientes virtuais flexíveis na Web.

Foram formuladas três hipóteses que, quando confirmadas, validam a tese deste trabalho.

A primeira hipótese (H1) era de que os elementos do MOrFEu são suficientes para especificação de qualquer instância de ambiente virtual utilizado em certo domínio de aplicação. No entanto, essa hipótese foi refutada, devido a inadequação do modelo MOrFEu de representar alguns ambientes virtuais.

Essa hipótese foi reformulada (H1.1): Os elementos conceituais apresentados inicialmente pelo MOrFEu acrescidos de alguns elementos possibilitam a especificação de qualquer instância de ambiente virtual utilizado em certo domínio de aplicação. Note que ainda são os elementos do MOrFEu que estão sendo usados, por isso ainda está de acordo com a tese principal. Não foi possível refutar essa hipótese com contraexemplo, mesmo depois de ter representado diversos ambientes virtuais. A amostra de tipos de ambientes virtuais utilizada é representativa, pois abrange os ambientes virtuais mais simples e comuns e também aqueles considerados mais complexos, dentro do domínio de aplicação estudado. Ou seja, a amostra utilizada

possui uma grande abrangência das funcionalidades dos ambientes virtuais conhecidos e desenvolvidos até a presente data.

A hipótese H2 afirmava que é possível implementar ferramentas que foram especificadas com os elementos do MOrFEu em ambientes virtuais. Essa hipótese também não foi refutada com contraexemplos. Foram implementados ambientes virtuais através de uma arquitetura de software desenhada segundo os preceitos do modelo proposto. A amostra de ambientes virtuais implementada é representativa, pois esses ambientes fazem uso, em conjunto, todos os elementos previsto pelo modelo. Ou seja, todos os elementos do modelo são implementáveis. Isso confirma a hipótese H2.

A hipótese H3 trata a respeito da flexibilidade das ferramentas implementadas e afirma que é possível definir esquemas de representação que permitam que modificações na especificação de um ambiente virtual produzam modificações correspondentes na ferramenta implementada. Foram feitas alterações em tempo de execução nos ambientes implementados, evidenciando assim essa hipótese.

Validando essas três hipóteses pode-se concluir que a tese principal deste trabalho também é válida.

## 8.5 Escopo e alocação no cenário de trabalhos relacionados

O modelo conceitual MX tem o objetivo de modelar ambientes virtuais de apoio a atividades pedagógicas, principalmente aquelas realizadas de maneira cooperativa. Num âmbito mais geral, se propõe a construir ferramentas web que auxiliem os usuários a interagirem uns com os outros. Tais ferramentas então funcionariam como um meio pelo qual a interação ocorreria e a ideia principal do modelo é do ambiente virtual como um organizador de um documento escrito coletivamente, composto das interações de seus usuários.

Dentre os ambientes modelados, há alguns que não se encaixam diretamente sob a definição convencional de um ambiente virtual. É o caso do Jogo de Dominó (seção 6.3.4). Ele não é uma ferramenta para comunicação direta entre os participantes do jogo. No entanto, há uma comunicação entre os participantes, uma vez que as pedras jogadas comunicam o que os jogadores desejam e fazem. E o documento formado por essa ferramenta é composto pelas interações desses usuários.

Outro ambiente modelado que foge da definição convencional de ambiente virtual é o Bolão da Copa (Seção 10.15, no Apêndice A). Trata-se de um sistema web onde os usuários cadastram "palpites" de jogos de futebol. O sistema é responsável por calcular a pontuação obtida por cada participante a partir dos palpites confirmados, e montar um ranking de usuários melhor pontuados. Como se pode notar, não há interação direta entre os usuários, eles apenas veem a pontuação e as apostas dos outros usuários, comparando-as com as dele próprio. Diferente do Jogo de Dominó, o documento que o Bolão da Copa compõe não é um conjunto de interações de usuários com outros usuários, no entanto possui um protocolo complexo, com diferentes restrições de horários e visualizações.

Sistemas web como o Bolão da Copa, são interessantes de serem modelados com o MX exatamente por essa característica: ser apenas um sistema web simples de cadastro, mas que possui um protocolo relativamente complexo com restrições de horários e visualização de conteúdo, mostrando que o MX pode se tornar uma boa opção quando o sistema for muito dinâmico ou quando possuir um protocolo de interação mais complexo.

Há sistemas que não são adequadamente modelados com o modelo MX, seja pela ausência de recurso para representação direta de contexto – como é o caso com sistemas que controlam outros sistemas ou dispositivos, pois não há maneira de representar a atuação de um sistema sobre outro sistema – quanto pela necessidade de estruturas de dados específicas – como é o caso com sistemas de busca, sistemas de recomendação ou sistemas de data warehouse.

**Tabela 5. 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 |

Na Tabela 5 é apresentado um resumo comparativo das abordagens para desenvolvimento de ambientes virtuais flexíveis onde nota-se a adequação do MX às características limitadas das outras abordagens listadas.

## 8.6 Contribuições

As contribuições iniciais deste trabalho são relativas à

- criação de ambientes virtuais que sejam alteráveis em tempo de execução
- definição mais precisa desses ambientes.

Os ambientes implementados com base no **modelo conceitual** proposto são alteráveis em tempo de execução, mantendo os dados que já haviam sido incluídos. Para que essas alterações sejam efetivadas em suas implementações, basta alterar a especificação desses ambientes. Essas alterações incluem mudanças de estrutura e das funcionalidades, podendo evoluir conforme as necessidades de seus usuários evoluem.

Dessa forma, espera-se uma **nova direção** no desenvolvimento de ambientes virtuais, combatendo o problema identificado na literatura da "adequação das atividades aos recursos disponíveis", passando para o foi defendido por Lévy (1999, p.

75) quando afirma que “os mundos virtuais irão multiplicar-se em quantidade e desenvolver-se em variedade” em proporção às facilidades disponibilizadas.

O modelo proposto neste trabalho inclui **recursos para descrição** de ambientes virtuais, que pode se constituir numa linguagem franca de comunicação entre desenvolvedores, uma vez que possibilita uma descrição precisa que pode ser usada na comparação com outros ambientes. Utilizando tais recursos, foi **modelado um conjunto de ambientes** virtuais típicos dos quais alguns também foram selecionados para prototipação.

Foi definida uma **plataforma de desenvolvimento** para ambientes virtuais e dado que a programação de um ambiente passa a ser a definição de suas estruturas e funcionalidades, o tempo de desenvolvimento é reduzido significativamente, gerando uma quantidade de código menor, facilitando o reuso e reduzindo as dificuldades técnicas frequentes na implementação de aplicações web.

## 8.7 Desdobramentos e Trabalhos Futuros

A investigação relatada nesta tese contribuiu com a pesquisa sobre flexibilidade em ambientes virtuais na web, abrindo um conjunto de novas questões e possibilidades para a exploração do tema. Um desdobramento natural deste trabalho é que a plataforma de desenvolvimento gerada para descrição e implementação de ambientes a partir do modelo MX, possa ser utilizada em larga escala por indivíduos e comunidades que necessitam de ferramentas diferenciadas para apoiar ações mediadas pela web. Para tanto, os recursos da plataforma precisam ser operadas diretamente pelos usuários, o que requer:

- Ferramentas para edição de especificações – baseadas nos esquemas diagramáticos do MX, facilitando a criação, organização e validação das especificações pelo próprio usuário.
- Ferramentas para descrição de *triggers* – em situações de interação menos convencionais.

- Mecanismos de geração de "esqueletos" – de código-fonte a partir dos diagramas produzidos nas ferramentas de edição citadas no item anterior
- Desenvolvimento de métodos para representação de Templates (visões do Documento) – ou adaptação de recursos de edição já desenvolvidos nas áreas da tecnologia web.

Com a integração de todos ou parte desses requisitos, a plataforma poderia ser utilizada massivamente tanto no domínio inicial considerado nesta tese – a Educação – quanto em outros domínios onde a autoria individual e cooperativa através da web sejam elementos centrais.

Os auxiliares de coordenação que foram objeto de investigação nesta tese, são agentes reativos, responsáveis principalmente pelo controle adicional de interações, estabelecendo gatilhos (*triggers*) de situações específicas. Entretanto, a representação explícita dos protocolos de interação e da estrutura de documentos e dos ambientes que os utilizam, sugere a possibilidade de uma camada de serviços entregues por agentes com maior autonomia e cognição, requerendo dentre outras coisas um protocolo de comunicação inter-agentes e elementos estruturantes de conhecimento (*eg.* ontologias), reduzindo o esforço cognitivo do usuário.

Foram identificados alguns itens relacionados com a evolução do modelo conceitual proposto, sendo o primeiro deles referente ao Protocolo de Interação, que pode ser aprofundado no quesito "papéis", em especial em questões como a forma e as permissões com que os usuários subscrevem a papéis.

Outro item que merece investigação mais aprofundada diz respeito ao conceito de "sub-VComs". Os ambientes virtuais de aprendizagem como o Moodle, usam várias ferramentas de comunicação/interação dentro do ambiente de um curso. Esses sub-VComs naturalmente podem ter suas próprias funcionalidades e permissões. Além disso, eles podem ser dinamicamente criados. O MX correntemente trata essas situações como visões diferenciadas de um mesmo documento, e todas as características e permissões de tais sub-VComs são parte do conjunto completo de permissões de todo o VCom.

## 8.8 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, *et al.*, 2008), avaliada com como "B2" pelo Qualis da CAPES;
- "MOrFEu: Criando Ambientes Virtuais Flexíveis na Web para Mediar a Colaboração", no Congreso Iberoamericano de Informática Educativa (IE 2010) (SANTOS, CASTRO e MENEZES, 2010), avaliada com como "B4" pelo Qualis da CAPES;
- "Flexible Virtual Environments for Teaching and Learning", na conferência Frontiers in Education (FIE 2012) (SANTOS, CASTRO e MENEZES, 2012), avaliada com como "B1" pelo Qualis da CAPES.

Ainda no período do desenvolvimento do curso de doutorado, o autor deste trabalho contribuiu no desenvolvimento dos seguintes artigos publicados:

- "Fleshing Out Clues on Group Programming Learning", na conferência International Conference on Enterprise Information Systems (ICEIS 2009) (CASTRO, *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, *et al.*, 2010).

# 9 Referências

AHMADI, N.; JAZAYERI, M.; LELLI, F.; REPENNING, A. 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.

ALLEN, C. Tracing the Evolution of Social Software. **Life With Alacrity**, 2004. Disponivel em: <http://www.lifewithalacrity.com/2004/10/tracing_the_evo.html>. Acesso em: Outubro 2013.

ALVES, E.; NUNES, C.; AXT, M.; THOMAZ, A. R.; ESSER, T. 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.; SILVA, A. C.; OTSUKA, J. L.; SILVA, C. G.; ROCHA, H. V. 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.; MEUNIER, R.; ROHNERT, H.; SOMMERLAD, P.; STAL, M. **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.; FUKS, H.; SANTOS, L.; JUNIOR, A. N. D. C. 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.

CASTRO, T.; FUKS, H.; SPOSITO, M.; CASTRO, A. N. The Analysis of a Case Study for Group Programming Learning. **Proceedings of 8th IEEE International Conference on Advanced Learning Technologies**, Santander, Espanha, 1, 2008. 850-854.

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.; CURBERA, F.; MEREDITH, G.; WEERAWARANA, S. 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. **The DICT Development Group**. Disponivel em: <http://www.dict.org/>. Acesso em: 27 Julho 2011.

DILLENBOURG, P.; MENDELSOHN, P.; JERMANN, P. Why spatial metaphors are relevant to virtual campuses? **Learning and instruction in multiple contexts and settings. Bulletins of the Faculty of Education**, 73, 1999. 61-71.

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.

EDUCAUSE EVOLVING TECHNOLOGIES COMMITTEE. Course Managment Systems (CMS), 2003.

ELLIS, C. A framework and mathematical model for collaboration technology. In: CONEN, W.; NEUMANN, G. **Coordination Technology for Collaborative Applications**. [S.l.]: Springer, 1998. p. 121-144.

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.

EZEQUIEL, V. D. C. Crítica da interação mediada por computador. **Revista de C. Humanas**, 9, n. 2, 2009. 265-277.

FAGUNDES, L. C.; NEVADO, R. A.; BASSO, M. V.; BITENCOURT, J.; MENEZES, C. S. 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.

FAGUNDES, L.; AZEVEDO, M.; ARAGON, R.; BITTENCOURT, J.; MENEZES, C. AMADIS - Um Ambiente Virtual para apoio ao Desenvolvimento de Projetos de Aprendizagem. **Anais do Simpósio Brasileiro de Informática na Educação**, 2005. 298-308.

FARIAS, C. R. G.; PIRES, L. F.; SINDEREN, M. A component-based groupware development methodology. **Proceedings of Enterprise Distributed Object Computing Conference**, 2000. 204-213.

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

FUKS, H.; RAPOSO, A.; GEROSA, M. A.; PIMENTAL, M.; LUCENA, C. J. P. 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.; NUNES, I.; FUKS, H.; LUCENA, C. J. P. An Approach for Developing Groupware Product Lines (GPL) based on the 3C. **CRIWG 2009, LNCS 5784**, Portugal, 2009. 328-343.

GEROSA, M. A.; PIMENTEL, M.; FUKS, H.; LUCENA, C. J. P. D. 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.

JENSEN, J. F. **'Interactivity':** Tracking a New Concept in Media and Communication Studies. [S.l.]: Aalborg University. Department of Mechanical Engineering, 1998. Disponível em http://www.nordicom.gu.se/common/publ_pdf/38_jensen.pdf.

JOHANSEN, R. **Groupware:** Computer Support for Business. [S.l.]: The Free Press, 1988.

JOHNSON-LENZ, P.; JOHNSON-LENZ, T. Consider the Groupware: Design and Group Process Impacts on Communication in the Electronic Medium. In: HILTZ, S.; KERR, E. **Studies of Computer Mediated Communications Systems:** A Synthesis of the Findings. Newark, New Jersey: Computerized Conferencing and Communications Center, New Jersey Institute of Technology, 1981.

JONES, S.; MARSH, S. Human-computer-human interaction: trust in CSCW. **SIGCHI BULLETIN**, 49, 1997. 36-40.

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.

LÉVY, P. **O que é o virtual?** Tradução de Paulo Neves. 2a. ed. São Paulo: 34, 1996. ISBN 978-85-7326-036-6.

LÉVY, P. **Cibercultura**. Tradução de Carlos Irineu da Costa. 1a. ed. São Paulo: 34, 1999. ISBN 85-7326-126-9.

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.

LITIU, R.; PRAKASH, A. Developing Adaptive Groupware Applications Using a Mobile. **Proceedings of the ACM Conference on Computer Supported**, 2000. 107-116.

MACLEAN, A.; CARTER, K.; LÖVSTRAND, L.; MORAN, T. 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.; GOMES, V.; RODRIGUES, A.; SANTOS, L.; CONTE, T. 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.; CASTRO, A. N.; SOUZA, F. F.; QUEIROZ, S. J. B.; BATISTA, V. O. Um Ambiente Telemático para mediar a Controvérsia Acadêmica. **Anais do Simpósio Brasileiro de Informática na Educação (SBIE)**, 2003.

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.

MENEZES, C. S.; NEVADO, R. A.; CASTRO, A. N.; SANTOS, L. N. 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.

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.; MENEZES, C.; NEVADO, R.; FAGUNDES, L. Ferramenta de Autoria e Interação para apoio ao desenvolvimento de Projetos de Aprendizagem. **Revista Novas Tecnologias na Educação (Renote)**, 3, n. 2, 2005.

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

MOODLE.ORG. Moodle Statistics. **Moodle.org**, 2013. Disponivel em: <http://moodle.org/stats/>. Acesso em: 16 Setembro 2013.

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.

NETO, F. A. D. A.; CASTRO, T. H. C. D.; CASTRO, A. N. D. Utilizando o Método Clínico Piagetiano para Acompanhar a Aprendizagem de Programação. **Anais do Simpósio Brasileiro de Informática na Educação**, 2006. 527-536.

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.

NEVADO, R.; BASSO, M. V.; BITTENCOURT, J. D. V. AMADIS: Ambiente de Aprendizagem a Distância para Formação Continuada de Professores. **Revista Informática na Educação – Teoria e Prática**, 4, n. 2, 2001.

OAKES, K. E-learning: LCMS, LMS - They're not just acronyms but powerful systems for learning. **Training & Development**, 56, n. 3, 2002. 73-75.

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.; MARIN, I.; GALLUD, J. A.; LOZANO, M. D.; TESORIERO, R. 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.

PRIBERAM. **Dicionário Priberam da Língua Portuguesa**. Disponivel em: <http://www.priberam.pt/DLPO/>. Acesso em: Outubro 2013.

PRIMO, A. Enfoques e desfoques no estudo da interação mediada por computador. **404NotFound**, n. 45, 2005. Disponível em http://www.facom.ufba.br/ciberpesquisa/404nOtF0und/404_45.htm.

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.; GIUNCHIGLIA, F.; VAN HARMELEN, F.; MARCHESE, M.; SABOU, M.; SCHORLEMMER, M.; SHADBOLT, N.; SIEBES, R.; SIERRA, C.; WALTON, C. **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.

ROSEMAN, M.; GREENBERG, S. Building real time groupware with GroupKit, a groupware toolkit. **ACM Transactions on Computer-Human Interaction**, 1996. 66-106.

ROTH, J.; UNGER, C. Developing synchronous collaborative applications with TeamComponents. **International Conference on the Design of Cooperative Systems**, 2000. 353-368.

RUSSEL, S.; NORVIG, P. **Inteligência Artificial**. Tradução de Vandenberg D. de Souza. 2a. ed. Rio de Janeiro: Elsevier, 2004. ISBN 85-352-1177-2.

SANTOS, L. N. **Desenvolvimento de um Multi-Organizador Flexível de Espaços Virtuais**. Universidade Federal do Amazonas. Manaus. 2009.

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. 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. N.; MENEZES, C. S. 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.

SLAGTER, R. J.; BIEMANS, M. C. M. Component Groupware: A Basis for Tailorable. **Proceedings of ICSC Conference on Intelligent Systems and Applications**, 2000.

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)**, 2008. 288-298.

TAIVALSAARI, A.; MIKKONEN, T.; INGALLS, D.; PALACZ, K. Web browser as an application platform: the lively Kernel experience. **SMLI TR-2008-175**, Mountain View, CA, USA, 2008.

THOMPSON, J. B. **A mídia e a modernidade:** uma teoria social da mídia. Petrópolis: Ática, 1998.

ULLRICH, C.; BORAU, K.; LUO, H.; TAN, X.; SHEN, L.; SHEN, R. 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.; DETERS, R.; GREER, J.; MCCALLA, G.; BULL, S.; KETTEL, L. Lessons from Deploying I-Help. **Proocedings Workshop Multi-Agent Architectures for Distributed Learning Environments, International Conference on AI and Education**, 2001. 3-11.

WAGNER, E. D. In support of a functional definition of interaction. **American Journal of Distance Education**, 8, 1994. 6-29.

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, 2005. 1-27.

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

# 10 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 modelo conceitual MX.

Foi feita a modelagem de 19 ambientes virtuais com o MX. Desde os mais simples e tradicionais, como o Fórum e Blog, até ambientes virtuais mais complexos, como aqueles que dão suporte às atividades pedagógicas, como por exemplo o Debate de Teses. Os primeiros são ambientes mais simples, em contraste com os ambientes de suporte a algumas atividades pedagógicas que demandam um grau de complexidade maior, com funcionalidades mais intricadas.

Alguns desses ambientes virtuais foram escolhidos por se tratarem de ferramentas tradicionais de comunicação, como o Fórum, Blog, Chat e Wiki. Outros por serem representantes da classe de ambientes virtuais que este trabalho se propõe a tratar, como o Diário Virtual, o Debate de Teses, Júri Simulado, Projeto de Aprendizagem, etc. Outros fogem desse escopo, mas têm características interessantes, como o Jogo de Dominó. Todos esses buscando compor um cenário abrangente de ambientes virtuais. Suas modelagens serviram como provas de conceito da proposta.

Outros ambientes modelados foram anteriormente apresentados na Seção 6.3.

## 10.1 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 73, um modelo de um VCom Blog utilizando-se caixas para representar as UPIs e setas para representar as relações entre as UPIs.

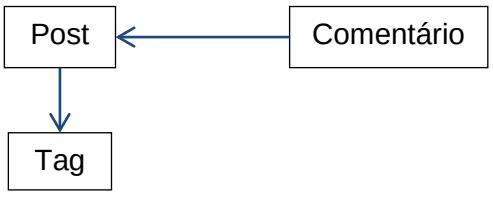


**Figura 73. Diagrama do modelo de documento de um blog.**

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 73 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 74, 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.

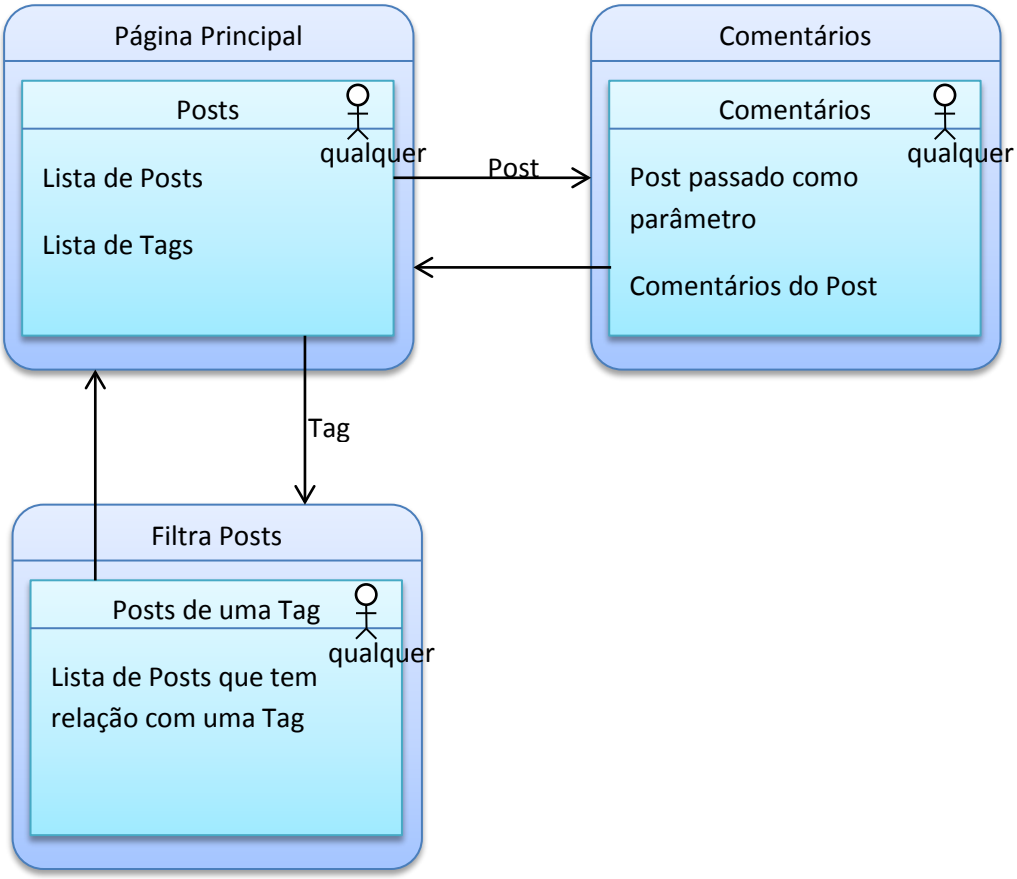


**Figura 74. Diagrama de navegação da interface de um blog.**

## 10.2 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 75, 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.

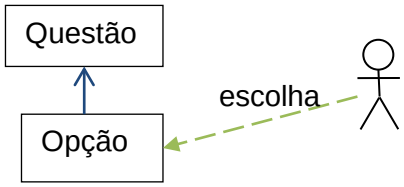


**Figura 75. Diagrama do modelo de documento de uma enquete.**

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 76:


**Figura 76. 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 77.

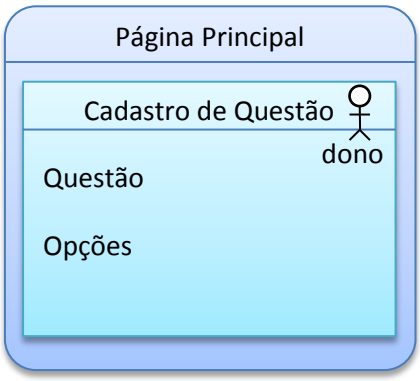


**Figura 77. Diagrama de navegação da interface de uma enquete em seu estado inicial.**

**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 78.

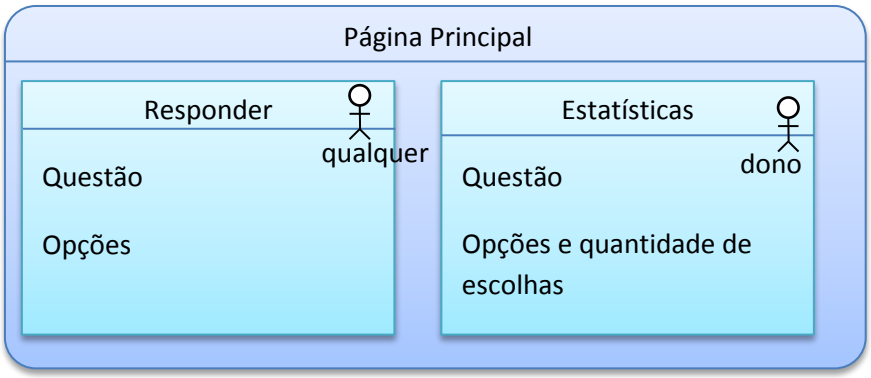


**Figura 78. Diagrama de navegação da interface de uma enquete em seu estado de responder.**

**Estado Fechada**. Nesse estado todos podem ver os resultados da enquete. O template de estatísticas é então exibido para todos (conforme Figura 79). A única permissão nesse estado é de que todos podem consultar o Documento inteiro.

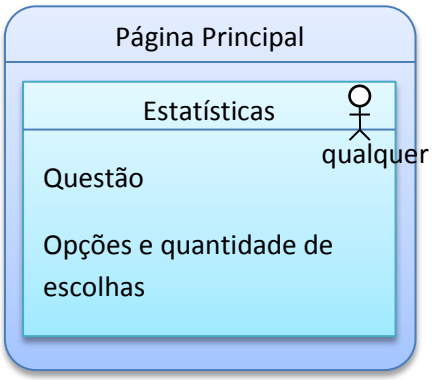


**Figura 79. Diagrama de navegação da interface de uma enquete quando ela se encontra fechada.**

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.

## 10.3 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 80 abaixo.

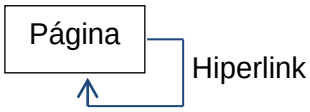


**Figura 80. Diagrama de um modelo de documento de um wiki.**

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.

## 10.4 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 80. 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.

## 10.5 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 81. Diferente das mensagens instantâneas do Facebook, essas mensagens não são direcionadas a um usuário específico, mas a todos os participantes do chat.

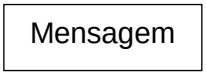


**Figura 81. Diagrama do modelo de documento de um chat.**

A atividade desse VCom é:

- Enviar Mensagem: publica uma UPI com o rótulo "Mensagem".

## 10.6 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 81.

A atividade desse VCom é:

- Incluir Postagem: publica uma UPI com o rótulo Mensagem.

## 10.7 CARTOLA

Trata-se de **objeto de aprendizagem** chamado **CARTOLA** (ALVES, *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 82.


**Figura 82. 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.

## 10.8 Controvérsia Acadêmica

Um fórum modificado para apoio à prática pedagógica **Controvérsia Acadêmica** realizada em ambientes virtuais (MENDONÇA, *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".

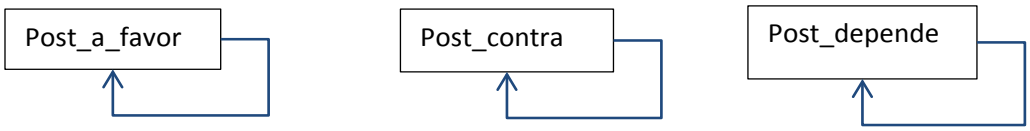


**Figura 83. Diagrama do modelo de documento de um fórum da Controvérsia Acadêmica.**

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.

## 10.9 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 84.

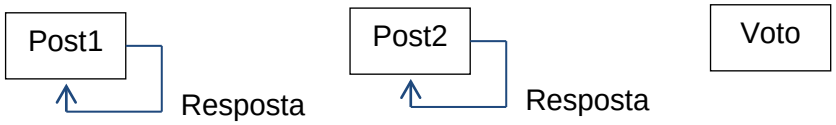


**Figura 84. Diagrama do modelo de documento de um quadro de discussão para o júri simulado.**

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.

## 10.10 Projeto de Aprendizagem

Na arquitetura pedagógica Projetos de Aprendizagem (FAGUNDES, *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.

Ou seja, o ambiente AMADIS proposto por Fagundes et al. (2005) estrutura um ambiente virtual para suporte a essa arquitetura pedagógica. Esse ambiente é composto por vários sub-VComs:

- um wiki, onde é feito o desenvolvimento do projeto;
- um diário de bordo, onde são registradas as ações cotidianas do projeto, num formato de blog;
- um fórum de orientação, usado para interação entre os participantes do grupo e o tutor da atividade; e
- um mural de visitas, onde outras pessoas podem postar seus comentários e perguntas sobre projeto que está sendo desenvolvido.

Na Figura 85, é exibido um diagrama que representa o documento desse tipo de VCom. As "nuvens" representam sub-VComs, na verdade um conjunto de UPIs e relações. Por já terem sido explicados anteriormente, não serão detalhados aqui. Da

forma que o AMADIS foi concebido, as UPIs não possuem relações entre si, e todo o VCom gira em torno da Questão de Investigação.


**Figura 85. Diagrama do modelo de documento de um projeto de aprendizagem.**

Dessa forma, as permissões desse VCom é o conjunto de permissões individuais de cada sub-VCom. Ele deve possuir uma Página inicial com a questão de investigação e links para os sub-VComs.

## 10.11 Facebook

O Facebook[45] 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 uma descrição da postagem original. Os comentários dessa "nova" postagem ficam no escopo do usuário que compartilhou, e não da postagem original.

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

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 86, é apresentado um diagrama do modelo de Documento de uma ferramenta como o Facebook, com as funcionalidades descritas acima.

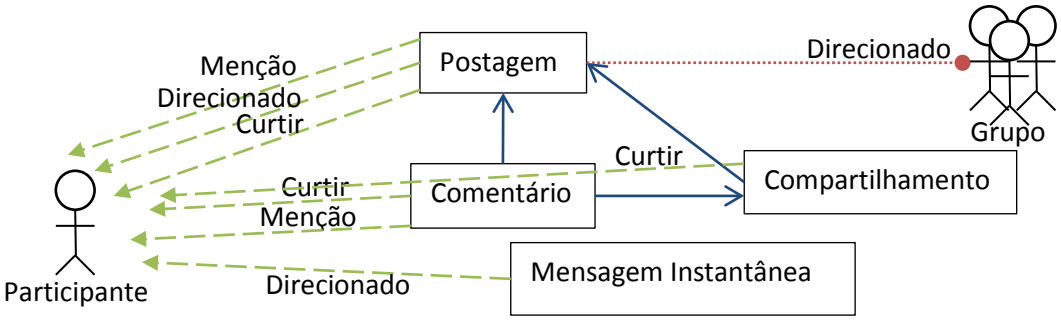


**Figura 86. Diagrama do modelo de documento do Facebook simplificado.**

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.

## 10.12 Apresentação de Artigos

Em sua tese, Paredes (2007) realizou um estudo de caso com o objetivo de exemplificar o seu modelo com um ambiente virtual simples, com fluxo de interação perceptível aos utilizadores. Trata-se de uma situação comum na comunidade acadêmica e científica, a apresentação de artigos científicos em conferências.

De acordo com o princípio apresentado por Paredes em sua tese, ele apresenta uma atividade do cotidiano "portada" para ambientes virtuais. Para criação de tal ambiente virtual foram observadas situações reais deste tipo e noções gerais sobre a situação. Como o objetivo primário era observar uma situação do cotidiano realizada em um ambiente virtual, a apresentação de artigos em conferência suportada por ambiente virtual está bem situada em seu trabalho.

No entanto, o ambiente virtual construído por Paredes não é um ambiente web, contudo ele possui características que podem facilmente ser adequadas para a Web.

Numa apresentação de um artigo científico numa conferência, existem três tipos de participantes: o autor do artigo que apresenta seu trabalho; o moderador da sessão em que o artigo é apresentado; e a audiência a quem se destina a apresentação, normalmente constituída por membros da comunidade científica que se inscreveram na conferência. Dessa forma são caracterizados os papéis que atuam nesse ambiente virtual.

Acontecem duas fases distintas numa apresentação de artigo: a apresentação do artigo e a sessão de questões. Isso caracteriza dois estados desse VCom.

Na fase de apresentação, o autor do artigo apresenta seu trabalho, recorrendo a serviços de interação presentes no ambiente virtual. Um serviço de conversação que permite ao autor expor seu trabalho à audiência, enviando mensagens de um-para-muitos. E um espaço compartilhado, na forma de "quadro branco" (*whiteboard*), que permite ao autor expor gráficos, desenhos ou uma apresentação eletrônica para a audiência.

Ainda na fase de apresentação, o moderador tem um papel importante: ele deve se assegurar que o autor não ultrapasse o tempo previsto para a apresentação, podendo interromper a apresentação ou até mesmo finalizá-la.

Nessa fase, Paredes identifica quatro atividades: falarAudiencia, utilizarWhiteboard, interromperApresentacao e finalizarApresentacao.

Para a atividade de falar à audiência é necessária uma ferramenta que permita ao autor se comunicar com a audiência numa via única. Se tratando de uma forma textual, como foi sugerido por Paredes, uma ferramenta semelhante a um Chat se mostra adequada, onde apenas o autor pode escrever e a audiência pode ler.

O *whiteboard* é uma ferramenta mais gráfica. Ou seja, suas UPIs não são do tipo textual, são do tipo figura ou similar. Ainda assim, é semelhante a um Chat, com UPIs de tipo diferente, permitindo a comunicação em apenas uma via de um-para-muitos.

A segunda fase (ou estado) do VCom é a sessão de questões da audiência. A audiência realiza suas perguntas e o autor deve dar a devida resposta para tais perguntas. Há a possibilidade também de membros da audiência somente comentarem o trabalho, ou seja, não há necessidade do autor dar uma resposta a esse tipo de manifestação.

O moderador é o responsável por organizar essa sessão de questões. Os membros da audiência devem fazer requisições de questões e comentários e o moderador é responsável por permitir que a audiência se manifeste, um de cada vez.

No sentido de organizar esta fase da interação, Paredes define doze ações de interação: interporQuestão, responderQuestão, comentar, moderar, terminarQuestões, requererComentário, requererQuestão, permitirQuestão, permitirComentário, permitirResposta, fimQuestao e fimComentario.

Dessa forma, é possível modelar o Documento desse VCom. Na Figura 87, são apresentadas as UPIs e suas relações: as mensagens da apresentação e do whiteboard; as solicitações e as permissões de questões e comentários são UPIs do tipo nulo; associada às permissões estão as questões e os comentários da audiência; e associado uma questão, está a resposta do autor.

Na Figura 88, é apresentado um diagrama de estados do VCom apresentação de artigos com três estados:

- Apresentação: quando o autor faz sua apresentação;

- Interrupção: o moderador pode suspender a apresentação;
- Questões: é a fase onde a audiência faz os questionamentos e comentários para o autor do trabalho.

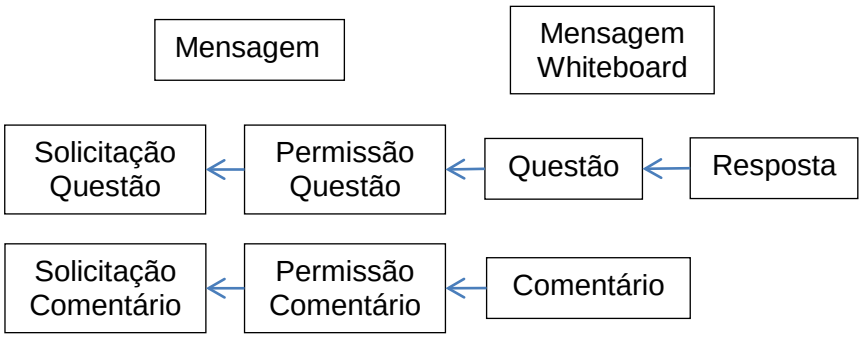


**Figura 87. Diagrama do modelo de documento da apresentação de artigo.**

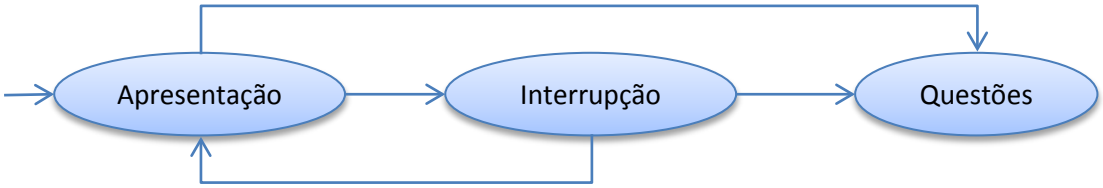


**Figura 88. Diagrama de estados de uma apresentação de artigos.**

As atividades desse VCom já foram antecipadas por Paredes, e podem ser descritas da seguinte forma:

- Falar à Audiência: publica uma UPI "Mensagem";
- Utilizar Whiteboard: publica uma UPI "Mensagem Whiteboard";
- Interromper Apresentação: muda o estado para "Interrupção";
- Retomar Apresentação: muda o estado para "Apresentação";
- Finalizar Apresentação: muda o estado para "Questões";
- Requerer Questão: publica uma UPI "Solicitação Questão" do tipo nulo;
- Requerer Comentário: publica uma UPI "Solicitação Comentário" do tipo nulo;
- Permitir Questão: publica uma UPI "Permissão Questão" do tipo nulo e a associa a uma UPI "Solicitação Questão";
- Permitir Questão: publica uma UPI "Permissão Comentário" do tipo nulo e a associa a uma UPI "Solicitação Comentário";
- Interpor Questão: publica uma UPI "Questão" e a associa a uma UPI "Permissão Questão";

- Comentar: publica uma UPI “Comentário” e a associa a uma UPI “Permissão Comentário”;
- Responder Questão: publica uma UPI “Reposta” e a associa a uma UPI “Questão”.

As permissões são associadas a cada estado. Em todos os estados é permitida a qualquer pessoa ler todo o documento.

No estado Apresentação:

- O autor pode Falar à Audiência;
- O autor pode Utilizar Whiteboard;
- O moderador pode Interromper Apresentação;
- O moderador pode Finalizar Apresentação;

No estado Interrupção:

- O moderador pode Retomar Apresentação;
- O moderador pode Finalizar Apresentação;

No estado Questões:

- Membros da audiência podem Requerer Questão;
- Membros da audiência podem Requerer Comentário;
- O moderador pode Permitir Questão, especificando qual solicitação foi atendida;
- O moderador pode Permitir Comentário, especificando qual solicitação foi atendida;
- Um membro da audiência pode Interpor Questão, se há uma solicitação feita por ele e foi permitida de ser feita;
- Um membro da audiência pode Comentar, se há uma solicitação feita por ele e foi permitida de ser feita;
- O autor pode Responder Questão, se a questão a ser respondida ainda não possui uma resposta.

A interface desse VCom constitui-se de apenas uma Página. Nessa página, são exibidas as mensagens da apresentação e as mensagens do whiteboard. São exibidas também as solicitações, as questões, as respostas e os comentários realizados.

Nota-se que o papel do moderador é indireto, ele não possui um canal de comunicação direta com nenhum dos outros participantes. Assim, pode-se imaginar uma adaptação desse VCom que inclua canais de comunicação com o autor do trabalho e com a plateia. Isso pode ser feito apenas acrescentando-se novas atividades e permissões de se realizar essa atividade. Novas UPIs podem também ser necessárias.

## 10.13 Estudo Bíblico

Esse VCom tem por finalidade o estudo do texto bíblico. Tem inspiração nas experiências pessoais do autor deste trabalho na utilização de bíblias online. Tem por objetivo a interação e colaboração para estudo do texto bíblico, de forma semelhante a que ocorre de forma presencial, onde um grupo de pessoas discute um trecho lido da bíblia. Nota-se que o texto bíblico, nesse caso, poderia ser substituído por qualquer outro texto (um romance ou um livro didático, por exemplo), e os mesmos princípios de interação poderiam ser aplicados.

Nas bíblias online encontradas na web, as únicas funcionalidades existentes são: a seleção de um texto especificado através do "Livro" e do "Capítulo" da bíblia; e a busca por palavras-chave dentro do texto bíblico, que retorna uma lista de "Versículos" (os "Capítulos" são compostos por uma lista de "Versículos", que são pequenos fragmentos de texto) associados com a busca realizada. Na versão para smartphones, alguns apps permitem que você faça ainda marcação de versículos favoritos. No entanto, essas aplicações (na web ou nos apps de smartphones) não permite a interação entre leitores interessados num mesmo texto.

Propõe-se, então, uma ferramenta de interação web baseada em um texto em comum, que permite que os leitores desse texto façam questionamentos e comentários acerca do que está sendo lido.

Um usuário pode fazer uma pergunta associada com um trecho do texto. Essa pergunta pode então ser respondida por outros leitores, criando uma discussão acerca

da questão levantada. Isso possibilita que futuros leitores que possivelmente tenham questões similares leiam uma discussão prévia feita sobre o tema.

É comum nos textos bíblicos haver citações, explicações e associações dentro dos próprios textos. Dessa forma, deve haver a funcionalidade de citações dentro do próprio texto. Essas associações podem ser entre trechos do texto ou entre citações em perguntas e/ou respostas da discussão.

Nota-se a importância do "trecho", ou seja, deve haver alguma forma de agrupar os versículos em trechos, pois todas as referências devem ser feitas baseadas em trechos e não apenas em versículos. Isso não exclui a possibilidade de um trecho ser apenas um versículo.

Normalmente, esses trechos são referenciados de forma resumida. Por exemplo, o trecho referenciado na forma "Ef 2:8-9" quer dizer que o trecho está presente no livro de "Carta aos Efésios", no Capítulo 2, nos versículos de número 8 a 9. Esse trecho é o seguinte: "Pois vocês são salvos pela graça, por meio da fé, e isto não vem de vocês, é dom de Deus; não por obras, para que ninguém se glorie" (Texto da versão NVI em português). Assim, deve haver um mecanismo que faça as associações com as referências realizadas pelos usuários aos trechos referenciados.

Na Figura 89, é apresentado o esquema do documento de um VCom Estudo Bíblico. Nele são organizados os livros, capítulos e versículos da bíblia. Os trechos são associados a um conjunto de versículos. Associados a um trecho um usuário pode fazer uma pergunta ou um comentário na forma de Post, que inicia um ramo de discussão, de forma muito parecida com um Fórum.

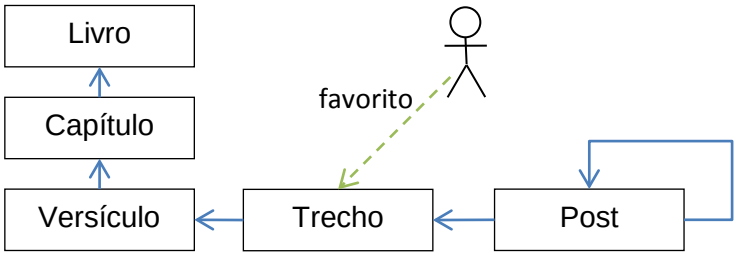


**Figura 89. Diagrama do modelo de documento de um estudo bíblico.**

Dessa forma, as atividades realizadas nesse VCom são:

- Criar Trecho: publica uma UPI "Trecho" e associa com um ou mais "Versículo";
- Marcar Favorito: um usuário faz uma associação dele com um trecho, indicando que aquele trecho é um "favorito";
- Começar Post: publica uma UPI "Post" e a associa com um "Trecho";
- Responder Post: publica uma UPI "Post" e a associa com outro "Post".

As permissões de atividades de escrita, listadas acima, podem ser feitas por qualquer usuário. Assim como, qualquer usuário pode ler todo o Documento.

A Interface desse VCom possui uma Página inicial, a página de Leitura de Capítulo e a página de Discussão.

A página inicial é apenas uma página de boas vindas e que dá acesso aos capítulos da bíblia. Pode haver também nessa página uma listagem dos últimos trechos que foram realizados Post.

Na página de Leitura de Capítulo, devem ser exibidos os versículos daquele capítulo e as interações realizadas a partir de trechos que continham algum de seus versículos. Para cada ramo de discussão deve haver um link para a página dessa discussão.

Na página de Discussão, deve haver toda a interação realizada e também os trechos associados a essa discussão.

Dois mecanismos que devem ser tratados como agentes são: um serviço de busca e um serviço de gerência de citações e trechos.

O serviço de busca é aquela funcionalidade presente nas atuais bíblias online, que permite que o usuário busque por versículos que contenham uma palavra-chave. Esse serviço pode ser um pouco mais inteligente, se levar em consideração também os trechos que possuem relação com o versículo e as discussões realizadas sobre esse versículo.

E o serviço de gerência de citações e trechos tem duas funcionalidades. A primeira é associar dinamicamente uma citação do tipo "Ef 2:8-9" ao trecho correspondente. E a segunda finalidade é garantir que não haja repetição de trechos no documento, ou seja, mais de um trecho que possua um mesmo conjunto de versículos.

Funcionalidades mais avançada podem ser desenvolvidas posteriormente. Tais como: sugestão de versículos interessantes; sugestão de discussões interessantes; inclusão de comentários pessoais; plano de leitura e marcação de versículos lidos; e produção de pregações e comentários bíblicos.

## 10.14 Interpretador Online de Código-fonte

Neto *et al.* (2006) propuseram uma ferramenta de acompanhamento das soluções dos estudantes durante o processo de aprendizagem de programação. Trata-se de um interpretador online (para Web) de códigos-fonte escritos na linguagem de programação Haskell, que foi dado o nome de AAEP – Um Ambiente para Acompanhamento e Análise de Programas. O ambiente foi projetado para atender essencialmente às demandas do professor.

Essa ferramenta possui três categorias de serviços: a gerência de problemas e usuários, a operação de um interpretador da linguagem de programação utilizada (no caso Haskell), e um controle de versões dos códigos gerados.

O professor é responsável por cadastrar os problemas e os usuários no sistema, ele também pode analisar as soluções que os estudantes desenvolveram.

Os estudantes conseguem visualizar uma lista de problemas propostos e desenvolver soluções para cada um deles. O editor de código-fonte é online, o usuário deve escrever o código e isso lhe permite salvar e ver as versões já produzidas por ele. A qualquer momento, ele pode submeter seu programa a testes, um interpretador de código Haskell é então executado, interpretando o código gerado e executando comandos do usuário em seu console interativo.

Essa ferramenta foi construída utilizando a linguagem PHP e banco de dados PostgreSQL. O interpretador Haskell utilizado foi o Hugs, mesmo programa utilizado nas aulas de laboratório. E o controle de versões foi feito com a ferramenta Subversion. Na Figura 90, é exibida uma tela do sistema em funcionamento, no momento em se está editando um código para a solução de um problema.

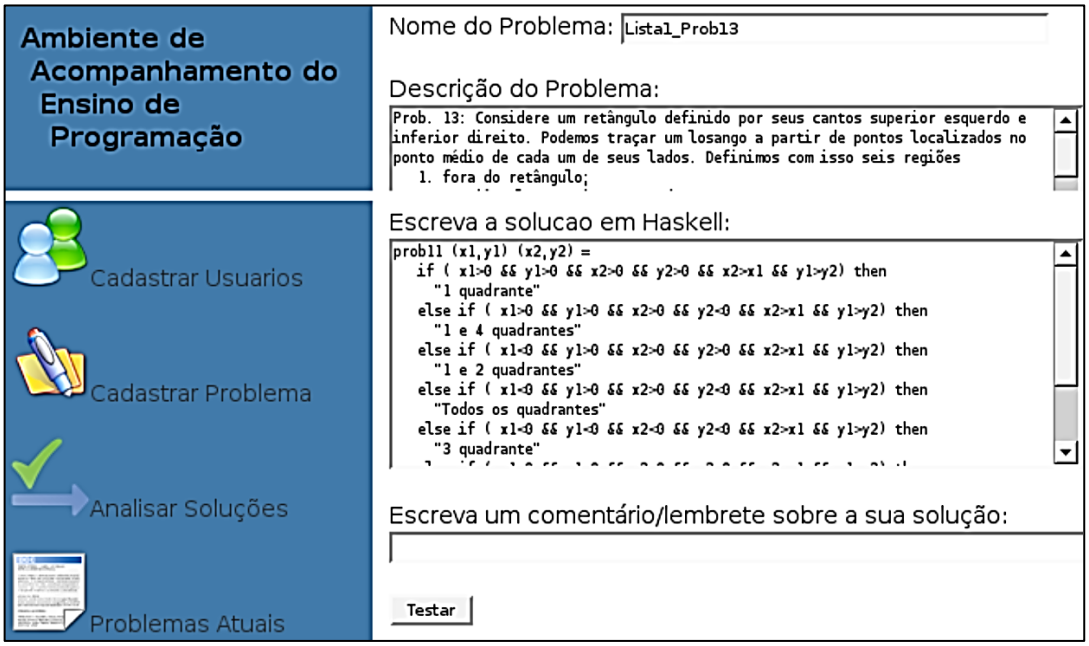


**Figura 90. Edição de Código Online no AAEP. Fonte: (NETO, CASTRO e CASTRO, 2006).**

É possível identificar dois tipos de UPI: os problemas e as soluções. Na Figura 91, é apresentado esse esquema, que é muito parecido com um Blog, com seus posts e comentários. As versões são salvas em várias soluções para um mesmo problema. As edições, são realizadas sempre na solução mais recente, é feita uma cópia dessa UPI e são aplicadas as modificações.


**Figura 91. Diagrama do modelo de documento do AAEP.**

O interpretador de Haskell é um agente de serviço separado. Ele deve ser invocado a partir de uma página. Pega o código e as entradas do console interativo do Hugs, processa e devolve o resultado, que deve ser apresentado para o usuário.

Algumas atividades desse VCom são:

- Cadastrar problema: publica uma UPI "Problema";
- Cadastrar Solução: publica uma UPI "Solução";

A interface desse VCom possui três páginas: um lista de problemas, uma página com a solução atual do problema, e um página para execução dos testes. Na Figura 92, é exibido um diagrama que contém essa representação.

É obvio que como foi apresentado por Neto *et al.* (2006), esse ambiente virtual proporciona pouca interação entre os usuários. Porém, como foi relatado, "o ambiente foi projetado para atender essencialmente às demandas do professor". Isso evidencia mais uma vez como os ambientes virtuais são construídos: nascem de uma demanda de um grupo, e no decorrer do tempo essas demandas vão evoluindo.

Algumas funcionalidades que podem ser incluídas:

- Registro de testes e resultados da execução;
- Possibilidade de comentários em códigos de outros usuários, ou num esquema parecido com o Diário de Resolução de Problemas de Matemática (SERRES e BASSO, 2009);
- Possibilidade de desenvolvimento de soluções em grupo, podendo inclusive suportar um esquema progressivo de resolução de problemas em grupo, tal como o sugerido por Castro *et al.* (2008);
- Possibilidade de apresentação de mais de uma solução, estimulando o estudante a testar formas diferentes de resolver o problema.

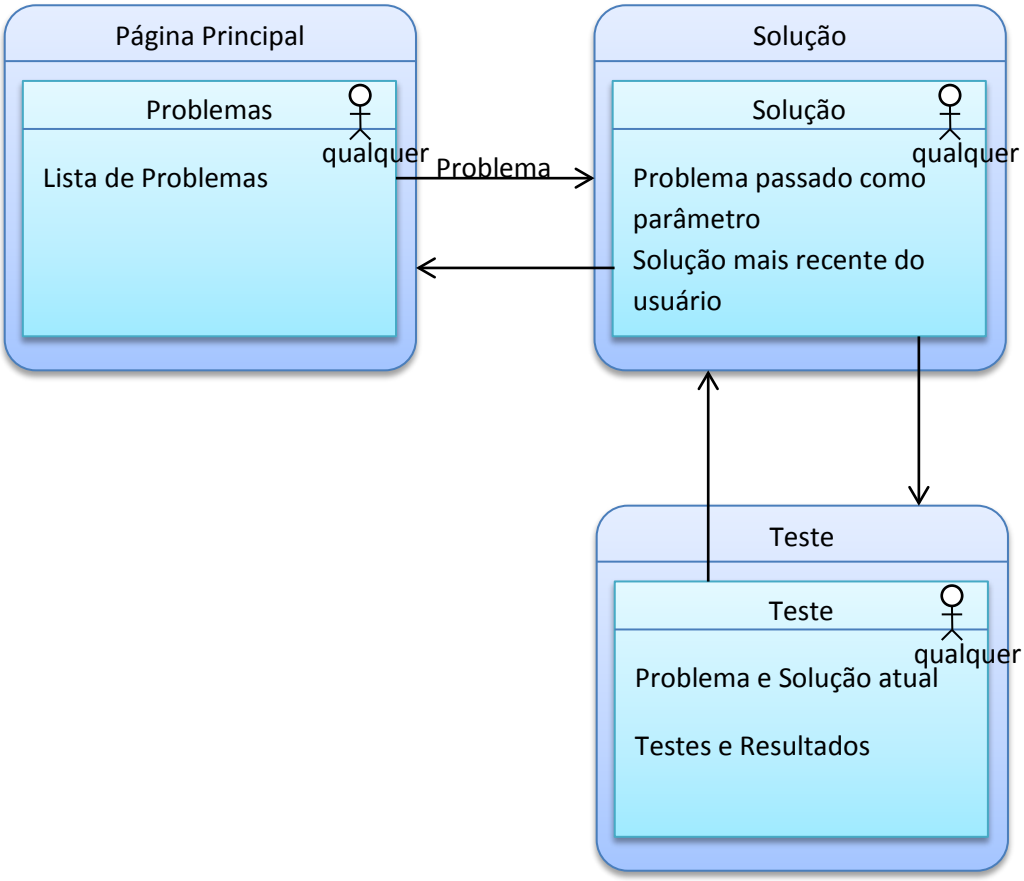


**Figura 92. Diagrama de navegação da interface do AAEP.**

## 10.15 Bolão de Apostas da Copa do Mundo

O Bolão de Apostas da Copa do Mundo foi uma ferramenta web criada por Guilherme de Alves Toda, em 2006, para proporcionar uma diversão entre um grupo de amigos que estavam acompanhando os jogos da Copa do Mundo de Futebol.

Bolão é uma espécie de jogo de aposta, onde os participantes palpitam sobre o resultado de algum jogo que ainda virá acontecer, onde ganha aquele participante que acertar o palpite.

Sua principal funcionalidade era cadastrar os palpites dos participantes do bolão com relação a todos os jogos que iriam ocorrer na Copa, o número de gols que será feito pelas duas equipes de uma partida. Depois de cadastrado o resultado real do jogo, o sistema calculava as pontuações de cada participante: 6 pontos por acertar o

vencedor do jogo e 3 pontos por acertar o número de gols de uma das equipes, podendo no máximo marcar 12 pontos (6+3+3=12) se acertar o placar exato da partida. O sistema deve controlar também a permissão de serem feitos os palpites: segundo a regra do bolão, o palpite de um jogo deve ser cadastrado antes do horário de início do jogo.

Na Figura 93, é exibida uma tela do sistema funcionando, para os jogos da Copa das Confederações, ocorrida em junho de 2013. No centro da tela estão os palpites realizados e na lateral direita estão os resultados reais dos jogos.


**Figura 93. Tela do Bolão da Copa.**

Dessa forma, as UPIs desse VCom são os jogos, os palpites e os resultados, como é apresentado na Figura 94.

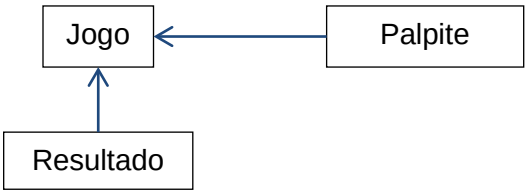


**Figura 94. Diagrama do modelo de documento do Bolão da Copa.**

Esse VCom possui dois pápeis: o gerente e os participantes. O gerente é responsável por cadastrar os jogos e o resultado. E os participantes podem cadastrar seus palpites e visualizar o ranking de pontuações no bolão.