UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
DEPARTAMENTO DE INFORMÁTICA
MESTRADO EM INFORMÁTICA

ELEU LIMA NATALLI

UM FRAMEWORK PARA CRIAÇÃO DE AMBIENTES COLABORATIVOS

VITÓRIA, AGOSTO 2011

ELEU LIMA NATALLI

UM FRAMEWORK PARA CRIAÇÃO DE AMBIENTES COLABORATIVOS

Dissertação submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Mestre em Informática.

VITÓRIA, AGOSTO 2011

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Central da Universidade Federal do Espírito Santo, ES, Brasil)

Natalli, Eleu Lima, 1983-
N272f Um framework para criação de ambientes colaborativos / Eleu Lima Natalli. – 2011.
144 f. : il.

Orientador: Crediné Silva de Menezes.
Dissertação (Mestrado em Informática) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Framework (Programa de computador). 2. Ambientes virtuais compartilhados. 3. Groupware (Software). 4. Projeto de sistemas. 5. Engenharia de software. I. Menezes, Crediné Silva de, 1952-. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 004

ELEU LIMA NATALLI

UM FRAMEWORK PARA CRIAÇÃO DE AMBIENTES COLABORATIVOS

Dissertação submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Mestre em Informática.

Aprovada em 31 de agosto de 2011.

COMISSÃO EXAMINADORA

_____

Prof. Dr. Crediné Silva de Menezes
Universidade Federal do Espírito Santo (UFES)
(Orientador)

_____

Prof. Dr. Davidson Cury
Universidade Federal do Espírito Santo (UFES)

_____

Prof. Dr. Alberto Nogueira de Castro Junior
Universidade Federal do Amazonas (UFAM)

_____

Prof. Dra. Rosane Aragon de Nevado
Universidade Federal do R. G. do Sul (UFRGS)

VITÓRIA, AGOSTO 2011

DEDICATÓRIA

Dedico este trabalho

aos meus pais, à minha irmã e à minha namorada,

que foram os principais responsáveis pelos incentivos,

motivações e apoio nos momentos de dificuldade.

AGRADECIMENTOS

Antes de tudo agradeço a Deus. Agradeço a Deus pela oportunidade de fazer um mestrado, já que em nosso país a carência de estudos é grande, acoplado com as poucas oportunidades. Agradeço a Deus por colocar em meu caminho pessoas especiais (familiares, amigos e professores), que em momentos difíceis nessa caminhada me ajudaram de alguma maneira a superar os diversos obstáculos.

Dedico um agradecimento especial ao professor Crediné Silva de Menezes, meu orientador. Através de seus questionamentos e dicas, me ajudou muito no direcionamento da caminhada exaustiva e difícil de confecção deste trabalho. Um grande abraço.

Outros agradecimentos aos professores Davidson Cury (vulgo “Dedê”) e Orivaldo Lira Tavares que, mesmo não me orientando diretamente, participaram das inúmeras discussões construtivas do projeto MOrFEu. Outro grande abraço.

Aos grandes companheiros de estudo do LIED Walber (vulgo “Walbera”), Ramon, Vinícius (vulgo “Gazela”), Leonardo (vulgo “Cachaça”), Eduardo (vulgo “Monteiro”), Bernardo (vulgo “Bernaúdo”), Victor, Charles, Everton, Otávio, Maikson e vários outros pelos diversos debates e conhecimentos trocados. Ficará o sentimento de falta destes momentos. Um abraço a todos.

Aos professores Alberto Nogueira de Castro Junior (UFAM) e Rosane Aragon de Nevado (UFRGS), membros externos da banca examinadora, por aceitarem o convite e avaliarem este trabalho, dispondo de seus preciosos tempos.

Aos meus amigos de infância, por compreenderem as inúmeras ausências em digníssimas confraternizações. A galera da UFES que direta ou indiretamente tiveram alguma influência na construção deste trabalho e também por compreenderem as ausências. Também não posso me esquecer da CAPES, que com apoio financeiro, oferecendo bolsa de estudos e apoio ao comparecimento em eventos.

À minha namorada, Lívia, pela imensa compreensão nestes anos de estudo. Pela paciência e incentivo nos diversos momentos em que tive que me ausentar, assim como pelos momentos de estresse que passei. Fica aqui um grande beijo.

E aos meus pais, Eleu e Vera, junto com minha irmã, Yane, são os grandes responsáveis por minha formação como pessoa. Meus agradecimentos pela compreensão dos momentos que ausentei e pelos momentos de impaciência devido aos esforços que este trabalho demandava. Fica aqui também um grande beijo a todos.

"A mente que se abre a uma nova idéia

jamais voltará ao seu tamanho original"

Albert Einstein

RESUMO

A Web oferece uma oportunidade inédita de promover a integração de pessoas para a realização de atividades de qualquer natureza, seja para o trabalho, para o lazer ou para o estudo. Com isto vê-se crescer a demanda por novas aplicações colaborativas. Neste contexto vimos surgir uma classe de aplicações denominada de Ambientes Colaborativos. Embora muito se tenha avançado com o surgimento de ambientes de propósito geral, onde os AVAs e CMSs se destacam, tem-se observado que estes nem sempre se ajustam às necessidade e preferências de grupos específicos. É portanto desejável a oferta de facilitadores para criação de propostas inovadoras. Nesse trabalho, é apresentado um *framework* para o desenvolvimento de ambientes pertencentes a esta classe de aplicações. O objetivo da proposta é facilitar o processo de construção deste gênero de ambientes, promovendo artefatos facilitadores para uso, segundo a abordagem do Modelo 3C de colaboração. A proposta foi construída a partir da metodologia *Hot-Spot Driven Design* acoplados com boas práticas da Engenharia de Software, além de utilizar os conceitos de Unidade de Produção Intelectual (UPI) e Veículos de comunicação (Vcom) para modelagem dos espaços de colaboração, propostos no projeto MOrFEu. Por fim, o trabalho também apresenta um protótipo do *framework* proposto, como forma de materializar os conceitos descritos e realçar a viabilidade da proposta, criando-se um conjunto de ambientes colaborativos.

Palavras-chave: *Framework*, Ambientes Colaborativos, Desenvolvimento de Ambientes Colaborativos.

ABSTRACT

The Web offers an unprecedented opportunity to promote the integration of persons to perform activities of any nature, whether for work, for leisure or for study. With this one sees growing demand for new collaborative applications. In this context we have seen an emerging class of applications called Collaborative Environments. Although much progress has been made with the emergence of general-purpose environments, where VLEs and CMSs stand out, it has been observed that these do not always fit the needs and preferences of specific groups. It is therefore desirable to offer facilitators for creating innovative proposals. In this work, we present a framework for the development of environments of this class of applications. The aim of the proposal is to facilitate the process of building this kind of environments, promoting artifacts facilitators to use, according to the approach of the 3C collaboration model. The proposal was built on the methodology Hot-Spot-Driven Design coupled with best practices of software engineering, in addition to using the concepts of Intellectual Production Unit (UPI) and communication vehicle (Vcom) for modeling collaborative spaces proposed in the MOrFEu project. Finally, the work also presents a prototype of the proposed framework as a way of materializing the concepts described and enhance the viability of the proposal, creating a set of collaborative environments.

Keywords: Framework, Collaborative Environments, Collaborative Development Environment

ÍNDICE DE FIGURA

ÍNDICE DE TABELA

SIGLAS

AJAX	*Asynchronous Javascript and* XML
API	*Application Programming Interface*
AVA	Ambiente Virtual de Aprendizagem
CER	Criação de Espaço Reservado
CMS	*Content Management System*
CMS	*Course Management System*
CSCL	*Computer-supported Cooperative Learning*
CSCW	*Computer-supported Cooperative Work*
CSS	*Cascading Style Sheets*
FAQ	*Frequently Asked Questions*
FrameColab	*Framework* de Ambientes Colaborativos
IDE	*Integrated Development Environment*
JAR	Java *ARchive*
LCMS	*Learning Content Management System*
LMS	*Learning Management System*
DAO	*Data Access Object*
DTO	*Data Transfer Object*
HVM	*Horizontal-Vertical-Metadata*
MOrFEu	Multi-Organizador Flexível de Espaços virtUais
MVC	*Model View Controller*
OMG	*Object Management Group*
PAC	*Presentation Abstraction Controller*
SCORM	*Sharable Content Object Reference Model*
SGBD	Sistema de Gerenciamento de Banco de Dados
SGML	*Standard Generalized Markup Language*
SQL	*Structured Query Language*
UFES	Universidade Federal do Espírito Santo
UML	*Unified Modeling Language*
UPI	Unidade de Produção Intelectual
Vcom	Veículo de comunicação
VLE	*Virtual Learning Environment*
W3C	*World Wide Web Consortium*
WAR	*Web application ARchive*
WYSIWYG	*What You See Is What You Get*
XML	*eXtensible Markup Language*

SUMÁRIO

CAPÍTULO 1 INTRODUÇÃO

1.1 INTRODUÇÃO

O surgimento da Internet e, mais tarde, da Web provocou uma verdadeira revolução na organização e desenvolvimento das atividades humanas, notadamente aquelas realizadas de forma colaborativa (Lee, 1999). Inicialmente partiu-se das conquistas tecnológicas para oferecer ferramentas que pudessem ser usadas no apoio a atividades intelectuais, com ênfase nas formas de trabalho de grupos com maior visibilidade. Hoje, busca-se identificar e apoiar as formas de trabalho diferenciadas de cada comunidade.

Segundo (Pessoa & Menezes, 2003):

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

Em um primeiro instante as atividades foram transportadas para o espaço virtual sob o condicionamento das ferramentas de comunicação, onde o compartilhamento das produções também ocorria através de tais ferramentas. Naquela fase, a produção coletiva foi socializada através dos programas de transferência de arquivos. Para cada atividade os usuários precisavam utilizar diferentes ferramentas.

O momento seguinte foi marcado pelo surgimento de ambientes voltados para a organização das produções coletivas, facilitando o compartilhamento e propiciando a integração de pessoas, documentos e interações. Assim surgiram os *wikis* e os *blogs*, onde os artefatos são diretamente integrados às ferramentas de gerência de revisão e de controle de versões.

O trabalho, o lazer e a aprendizagem construídos individual e coletivamente tomaram novo e determinante impulso a partir da disseminação das ferramentas de software que atualmente compõem os ambientes virtuais baseados na Web. Após um período de reconhecimento e apropriação desses recursos, o trabalho, o ensino, a aprendizagem e as relações sociais como um todo passam a apresentar demandas que os ambientes virtuais ainda não conseguem atender, em parte devido à conformação desses ambientes a certos aspectos funcionais mantidos principalmente por tradição. Uma estratégia possível para tratar esse problema é a concepção e desenvolvimento de ambientes flexíveis para apoiar a realização de atividades cooperativas, propiciando aos usuários uma melhor sintonia entre ferramentas de apoio e os objetivos e os perfis dos participantes de uma determinada atividade. Este trabalho irá apresentar uma possível estratégia, usá-la para concepção de Ambientes Colaborativos e ainda propor um *framework* que a materialize.

1.2 MOTIVAÇÃO

O desenvolvimento de Ambientes Colaborativos utilizando uma abordagem convencional, conforme será descrito neste trabalho, acaba se tornando custosa, o que resulta em propostas de ambientes pouco relevantes, dificultando surgimento de inovações. Assim, a abordagem diferenciada que esta dissertação utiliza para elaboração de espaços virtuais representa um grande motivador na construção da proposta, uma vez que trata-se de um modo diferente do convencional. A pouca quantidade de materiais produzidos nessa linha de pensamento também têm um peso na motivação para construção do trabalho, uma vez que ele será de grande utilidade e valor.

1.3 OBJETIVO

O objetivo geral e principal do trabalho é a concepção de um *framework* para auxílio na criação de ambientes colaborativos segundo um paradigma diferenciado para a

organização da produção intelectual de seus usuários. São objetivos específicos deste trabalho:

i. Identificar as limitações no desenvolvimento convencional de Ambientes Colaborativos, enfatizando as facilidades provindas pela abordagem presente no projeto MOrFEu;

ii. Propor um arcabouço teórico (tanto metodológico quanto conceitual) de modo que fundamente a construção de *frameworks* para este domínio;

iii. Construção de uma versão funcional do *framework* (protótipo), de modo que Ambientes Colaborativos possam ser construídos; e

iv. Relatar exemplos de Ambientes Colaborativos criados a partir do protótipo funcional de modo a evidenciar a aplicabilidade da proposta.

1.4 METODOLOGIA E HISTÓRICO DO DESENVOLVIMENTO DO TRABALHO

Inicialmente foi realizado um estudo acerca do desenvolvimento de softwares colaborativos. Após tal feito, foi realizado um esclarecimento da nova concepção de desenvolvimento. Posteriormente foi feito um levantamento na literatura acerca de *frameworks* (conceitos, metodologias para desenvolvimento/evolução, etc) e conceitos correlatos (padrões, componentes, etc), para servir de apoio para a confecção da proposta. Neste levantamento buscou-se entender como as abordagens de confecção de *frameworks* utilizam as informações contidas nos modelos e como as decisões de projeto são incorporadas no desenvolvimento de sistemas computacionais, bem como os conceitos descritos na teoria se relacionam.

1.5 ORGANIZAÇÃO

Este trabalho é estruturado em sete capítulos. O conteúdo de cada um desses é descrito resumidamente abaixo:

- *Capítulo 2 – Referencial Teórico*: apresenta o referencial teórico a respeito de *frameworks*, metodologias de desenvolvimento de *frameworks*, padrões e vários outros conceitos adjacentes.

- *Capítulo 3 – Domínio do Problema: Ambientes Colaborativos*: apresenta de forma clara e detalhada o domínio e os conceitos adjacentes do domínio a qual o trabalho está inserido. Ao final do capítulo é apresentado os pontos críticos pertencentes ao domínio.

- *Capitulo 4 – Uma abordagem inovadora para concepção de Ambientes Colaborativos*: apresenta uma descrição de conceitos que pertencem a uma nova abordagem para criação de Ambientes Colaborativos, bem como expor a real necessidade de tal demanda. Etapa muito importante pois descreve os conceitos/termos que serão absorvidos pelo *framework* proposto, além de retratar a importância desta.

- *Capítulo 5 – O framework proposto*: descreve em detalhes o *framework* proposto no trabalho. Está organizado de acordo com as etapas básicas do processo de desenvolvimento de Engenharia de Software, formalizando cada parte da proposta.

- *Capítulo 6 – Um protótipo para o FrameColab*: apresenta o protótipo desenvolvido baseado na proposta desta dissertação. Também ilustra a criação de alguns ambientes pré-estabelecidos através protótipo desenvolvido, permitindo evidenciar a viabilidade da proposta deste trabalho, além de reforçar a importância da abordagem descrita no Capítulo 4.

- *Capítulo 7 - Conclusões e Trabalhos Futuros*: apresenta as conclusões do trabalho, as contribuições e propostas de trabalhos futuros que podem ser desenvolvidos com base neste trabalho.

CAPÍTULO 2 REFERENCIAL TEÓRICO

2.1 INTRODUÇÃO

Este capítulo apresenta o referencial teórico necessário para o entendimento deste trabalho como um todo. O capítulo está organizado da seguinte forma:a seção 2.2 introduz uma breve contextualização do cenário a qual os conceitos pertencem, a seção 2.3 apresenta uma discussão acerca de padrões e os seus principais tipos, a seção 2.4 apresenta conceitos de *frameworks*, as principais classificações e arquiteturas, e ainda outros conceitos adicionais e, finalmente, a seção 2.5 descreve alguns processos de desenvolvimento de *frameworks*.

2.2 CONTEXTUALIZAÇÃO

A reutilização de software é um propósito enunciado quase simultaneamente com o surgimento da própria Engenharia de Software (Bosch, Molin, Mattsson, Bengtsson, & Fayad, 1999). Na década de 70 surgiram os primeiros trabalhos que se referiam ao termo “interesse” como um conceito de modularização de software (Dijkstra, 1976). Ainda essa época, com a definição da programação estruturada, os módulos podiam ser vistos como componentes reusáveis em sistemas novos. Até então o reuso era conseguido copiando-se o código e editando-o para adaptá-lo ao novo sistema.

Nos anos 80, com a popularização da programação orientada a objetos, o reuso por meio de herança mostrou-se mais poderoso na adaptação do código a novos sistemas. O reuso obtido era ainda apenas em pequena escala, consistindo de componentes de código a serem utilizados de sistemas maiores (Bosch, Molin, Mattsson, Bengtsson, & Fayad, 1999). O paradigma da orientação a objetos não resolvia o problema, bem mais complexo, de reutilizar componentes grandes que cuidassem de parte significativa de um sistema.

A análise de domínios ganhou forças, também na década de 80, com o objetivo de maximizar a reutilização no desenvolvimento de software, por meio da identificação e

organização do conhecimento a respeito de uma classe de problemas - um domínio de aplicações (Prieto-Diaz & Arango, 1991). Ela é um passo fundamental na criação de artefatos de software reutilizáveis, já que o modelo nela produzido capta a funcionalidade essencial requerida por um domínio e pode ser reutilizado durante a análise de sistemas de tais domínios.

Muito do que se propôs nessa época e em pesquisas subsequentes foi sendo gradativamente incorporado às linguagens de programação e aos modelos de processo de desenvolvimento de software, culminando na década de 90 com o desenvolvimento orientado a objetos e a padronização propiciada pela UML (*Unified Modelling Language*) (Jacobson, Booch, & Rumbaugh, 1999).

Nesse mesmo contexto surgiram os *frameworks* de software orientados a objetos (abordado em detalhes na seção 2.4), que permitem o reuso de grandes estruturas em um domínio particular, as quais são personalizadas para atender aos requisitos de aplicações específicas desse domínio (Johnson & Foote, 1988) (Johnson & Russo, 1991). Assim, famílias de aplicações similares, mas não idênticas, podem ser derivadas a partir de um único *framework*.

Ainda com o objetivo de reuso em diferentes níveis de abstração, surgiram também na década de 90, os padrões de software (Buschmann, Meunier, Rohnert, Sommerlad, & Stal, 1996) (Coad, 1992) (Gamma, Helm, Johnson, & Vlissides, 1994), que tentam captar a experiência adquirida no desenvolvimento de software e sintetizá-la em forma de problema e solução (discutido em detalhes na seção 2.3).

O domínio deste trabalho será apresentado em detalhes no Capítulo 3. Por fim, vale a pena enfatizar que *frameworks* são, na maioria das vezes, muito complexos para construir, entender e utilizar. Assim, seu desenvolvimento deve ser amparado por alguma metodologia, como forma de facilitar o processo. Tal assunto será exposto em detalhes na seção 2.5.

2.3 PADRÕES

Programadores freqüentemente empregam soluções que eles já utilizam antes em situações parecidas. Estas soluções envolvem o projeto de classes, redefinições de métodos, combinação de objetos, delegação de mensagens e assim por diante. Um padrão descreve uma solução para um problema que ocorre com freqüência durante o desenvolvimento de software, podendo ser considerado como um par "problema/solução" (Buschmann, 1996). Projetistas familiarizados com certos padrões podem aplicá-los imediatamente a problemas de projeto, sem ter que redescobri-los (Gamma, Helm, Johnson, & Vlissides, 1994). Um padrão é um conjunto de informações instrutivas que possui um nome e que capta a estrutura essencial e o raciocínio de uma família de soluções comprovadamente bem sucedidas para um problema repetido que ocorre sob um determinado contexto e um conjunto de repercussões (Appleton, 1997).

Padrões de software podem se referir a diferentes níveis de abstração no desenvolvimento de sistemas orientados a objetos. Assim, existem padrões arquiteturais, em que o nível de abstração é bastante alto, padrões de análise, padrões de projeto, padrões de código, entre outros. As diversas categorias de padrões são discutidas a frente. O uso de padrões proporciona um vocabulário comum para a comunicação entre projetistas, criando abstrações num nível superior ao de classes e garantindo uniformidade na estrutura do software (Gall, Klösch, & Mittermeir, 1996). Além disto, eles atuam como blocos construtivos a partir dos quais projetos mais complexos podem ser construídos (Gamma, Helm, Johnson, & Vlissides, 1994).

2.3.1 Classificação de padrões

Conforme já mencionado, padrões de software abrangem diferentes níveis de abstração, podendo portanto ser classificados em diversas categorias de modo a facilitar sua recuperação e uso. Porém, essa classificação não é rigorosa, podendo haver padrões que se encaixem em mais do que uma categoria. A seguir resumem-se algumas categorias importantes de padrões. Outras categorias de padrões são discutidas em (Coplien & Schmidt, 1995), (Vlissides, Coplien, & Kerth, 1996), (Martin, Riehle, & Buschmann, 1998).

- Padrões de processo: definem soluções para os problemas encontrados nos processos envolvidos na engenharia de software: desenvolvimento, controle de configuração, testes, etc;

- Padrões arquiteturais: expressam o esquema ou organização estrutural fundamental de sistemas de software ou hardware;

- Padrões de padrão (em inglês, *patterns on patterns*): são padrões descrevendo como um padrão deve ser escrito, ou seja, que padronizam a forma com que os padrões são apresentados aos seus usuários;

- Padrões de análise: descrevem soluções para problemas de análise de sistemas, embutindo conhecimento sobre o domínio de aplicação específico;

- Padrões de projeto: definem soluções para problemas de projeto de software;

- Padrões de interface: definem soluções para problemas comuns no projeto da interface de sistemas. É um caso particular dos padrões de projeto;

- Padrões de programação: descrevem soluções de programação particulares de uma determinada linguagem ou regras gerais de estilo de programação;

- Padrões de Persistência: descrevem soluções para problemas de armazenamento de informações em arquivos ou bancos de dados;

- Padrões para Hipertexto: descrevem soluções para problemas encontrados no projeto de hipertextos;

- Padrões para Hipermídia: descrevem soluções para problemas encontrados no desenvolvimento de aplicações hipermídia.

2.3.2 Coletânea de padrões

Conforme já mencionado, diversos autores têm proposto centenas de padrões nas mais

diversas áreas de aplicação. Esses padrões são descobertos após a abstração de fatores comuns em diversos pares problema-solução, podendo-se situar em várias faixas de escala e abstração.

Olhando mais de perto para muitos padrões, percebe-se que um padrão resolve um problema, mas sua aplicação pode gerar outros problemas, que podem ser resolvidos por outros padrões. Em geral não existem padrões isolados: um padrão pode depender de outro no qual esteja contido, ou das partes que ele contém, ou ser uma variação de outro, ou ser uma combinação de outros. De maneira geral, um padrão e seus variantes descrevem soluções para problemas muito similares, que variam em algumas das influências envolvidas. Portanto, deve-se pensar em formas de agrupar os padrões existentes segundo algum critério, de forma a facilitar sua recuperação e reuso.

2.3.3 Alguns dos principais padrões utilizados no trabalho

- DAO: consiste em abstrair o mecanismo de persistência utilizado na aplicação. A camada de negócios acessa os dados persistidos sem ter conhecimento se os dados estão em um banco de dados relacional ou um arquivo XML, por exemplo. O padrão DAO esconde os detalhes da execução da origem dos dados (Sun, 2007). Trata-se de um padrão de persistência;

- DTO: é um padrão de pretende aumentar a granularidade de acesso e transferência de dados entre camadas e/ou nodos. Trata-se também de um padrão de persistência;

- *Façade*: fornecer uma interface unificada (ilustrado na Figura 2.1) para um conjunto de interfaces em um subsistema. Define uma interface de nível mais alto que torna o subsistema mais fácil de ser usado (Gamma, Helm, Johnson, & Vlissides, 1994). Trata-se de um padrão de interface;

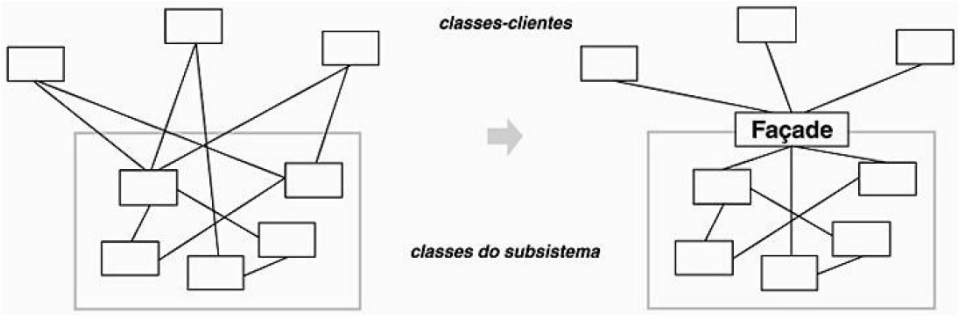


Figura 2.1 - Ilustração de uso de *façade*. Adaptado de (Gamma, Helm, Johnson, & Vlissides, 1994)

- *Factory*: oferece uma interface para a criação de famílias de objetos relacionados ou dependentes, sem especificar suas classes concretas. Ele é utilizado para definir e manter relacionamentos entre objetos. A idéia deste padrão consiste no cliente chamar um método abstrato especificado em alguma classe abstrata (ou interface) e a subclasse concreta vai decidir que tipo exato de objeto criar e retornar. Trata-se de um padrão de interface;

- HVM: consiste na definição de interfaces que independem de implementação (horizontal), interfaces que dependem de implementação (vertical) e no uso de metadados para acoplar as partes dinâmicas (ver Figura 2.2). Ajuda na utilização/criação de componentes e/ou trechos reutilizáveis. Trata-se de um padrão arquitetural;

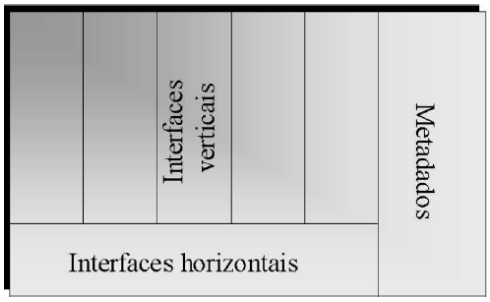


Figura 2.2 - Ilustração do HVM. Adaptado de (Mowbray & Malveau, 1997)

- MVC: segundo (Gamma, Helm, Johnson, & Vlissides, 1994), antes do MVC, os projetos de interface para o usuário tendiam a agrupar os objetos de modelo, visão e controle. MVC separa esses objetos para aumentar a flexibilidade e a reutilização. A separação entre os objetos é estabelecida por um protocolo de inserção/notificação entre eles, na qual uma visão deve garantir que a sua

aparência reflita o estado do modelo, de tal modo que quando os dados do modelo mudam, esse notifica as visões que dependem dele. Em resposta, cada visão tem a oportunidade de atualizar-se. Essa abordagem permite que uma aplicação possua várias visões para um modelo sem precisar reescrevê-lo. Também é outro exemplo de padrão arquitetural;

- *Repository*: tem, em uma abordagem muito semelhante ao DAO, a responsabilidade de representar a camada de dados na aplicação. A semelhança entre DAO e *Repository* é tanta que em muitas abordagens são programados da mesma forma, ou simplesmente delegam do Repositório para o DAO. Porém os padrões atuam em camadas distintas da aplicação, e podem até mesmo coexistir. Os Repositórios pertencem ao modelo da aplicação, são parte de camada de negócios complementando o *Model* e fornecendo estes objetos as outras camadas (como Controle e Apresentação, se aplicável) e recuperando estes do banco de dados. Trata-se de um padrão de persistência;

- *Singleton*: é responsável por assegurar que uma classe tenha uma única instância e por prover um ponto de acesso global a esta instância. Os clientes obtém um objeto da classe *Singleton* solicitando-o à classe. A classe *Singleton* faz o controle do número de instâncias. Quando a solicitação de uma instância é feita, a classe *Singleton* cria uma instância ou retorna a já existente. A instância deverá ser única para todo o sistema (Gamma, Helm, Johnson, & Vlissides, 1994). Trata-se de um padrão de projeto;

2.4 FRAMEWORKS: CONCEITOS GERAIS

De acordo com (Fayad, Johnson, & Schmidt, 1999), *frameworks* orientados a objetos são soluções de software que têm o objetivo de apoiar o desenvolvimento de aplicações em domínios específicos. O objetivo principal é a reutilização de projeto e código em um esforço para aumentar a velocidade do desenvolvimento e a qualidade das aplicações desenvolvidas.

Em (Wirfs-Brock, Vlissides, Cunningham, Johnson, & Bollette, 1990) é proposto a seguinte definição para *framework*: “um esqueleto de implementação de uma aplicação

ou de um subsistema de aplicação, em um domínio de problema particular. É composto de classes abstratas e concretas e provê um modelo de interação ou colaboração entre as instâncias de classes definidas pelo *framework*. Um *framework* é utilizado através de configuração ou conexão de classes concretas e derivação de novas classes concretas a partir das classes abstratas do *framework*". Já (Wirfs-Brock & Johnson, 1990) estabelece que um *framework* é "uma coleção de classes concretas e abstratas e as interfaces entre elas, e é o projeto de um subsistema". (Johnson, 1993) ressalta: "não apenas classes, mas a forma como as instâncias das classes colaboram".

Dois aspectos caracterizam um *framework* segundo (Taligent, 1995):

- Os *frameworks* fornecem infra-estrutura e projeto: *frameworks* portam infra-estrutura de projeto disponibilizada ao desenvolvedor da aplicação, que reduz a quantidade de código a ser desenvolvida, testada e depurada. As interconexões preestabelecidas definem a arquitetura da aplicação, liberando o desenvolvedor desta responsabilidade. O código escrito pelo desenvolvedor visa estender ou particularizar o comportamento do *framework*, de forma a moldá-lo a uma necessidade específica;

- Os *frameworks* "chamam", não são "chamados": um papel do *framework* é fornecer o fluxo de controle da aplicação (em seções posteriores tal conceito ficará mais claro). Assim, em tempo de execução, as instâncias das classes desenvolvidas esperam ser chamadas pelas instâncias das classes do *framework*.

Um *framework* se destina a gerar diferentes aplicações para um domínio. Precisa, portanto, conter uma descrição dos conceitos deste domínio. As classes abstratas de um *framework* são os repositórios dos conceitos gerais do domínio de aplicação. No contexto de um *framework*, um método de uma classe abstrata pode ser deixado propositalmente incompleto para que sua definição seja acabada na geração de uma aplicação. Apenas atributos a serem utilizados por todas as aplicações de um domínio são incluídos em classes abstratas.

Por fim, (Fayad, Johnson, & Schmidt, 1999) também é comentado que um ponto importante durante a construção de um *framework* é identificar as partes fixas (*frozen-*

spots) e variáveis (*hot-spots*) de um domínio de aplicação. Diferentes aplicações em um mesmo domínio são distinguidas por uma ou mais partes variáveis. Elas representam as partes do *framework* que são específicas de sistemas individuais. As partes variáveis são projetadas para serem genéricas e podem ser adaptadas às necessidades da aplicação. Partes fixas definem a arquitetura geral de um sistema de software – seus componentes básicos e os relacionamentos entre eles. As partes fixas são usadas sem nenhuma modificação em todas as instanciações de um *framework* de aplicação. Uma outra denominação semelhante aos pontos variáveis é a de gancho (em inglês, *hook*), que consiste de um ponto do *framework* passível de ser adaptado de alguma forma, por exemplo, por meio do preenchimento de parâmetros ou da criação de subclasses (Froehlich G. , Hoover, Liu, & Sorenson, 1997). A descrição de cada gancho documenta um problema ou requisito do *framework* e oferece diretrizes de como utilizar o gancho para satisfazer tal requisito.

2.4.1 Vantagens no uso de frameworks

As principais vantagens e objetivos da utilização de um *framework*, segundo (Fayad & Schmidt, 1997), (Fayad, Johnson, & Schmidt, 1999), (Jacobsen, Kristensen, & Nowack, 1997) e (Froehlich G. , Hoover, Liu, & Sorenson, 1997), derivam da modularidade, reusabilidade, extensibilidade e inversão de controle que ele provê aos desenvolvedores, como descrito a seguir:

- **Modularidade**: *frameworks* aumentam a modularidade encapsulando detalhes voláteis da implementação atrás de interfaces estáveis. A modularidade de um *framework* ajuda a promover a qualidade do software localizando o impacto das mudanças de projeto e implementação. Esta localização reduz o esforço necessário para entender e manter um sistema;

- **Reusabilidade**: as interfaces estáveis providas pelo *framework* aumentam a reusabilidade definindo componentes genéricos que podem ser reaplicados para gerar novos componentes. A reutilização dos componentes do *framework* podem ainda aumentar substancialmente a produtividade dos desenvolvedores, bem como a qualidade, performance, confiança e interoperabilidade do software;

- **Extensibilidade**: um *framework* aumenta a extensibilidade provendo métodos de “gancho” explícitos (Pree, 1994) que permitem às aplicações estender suas interfaces;

- **Inversão de Controle**: a arquitetura de tempo de execução de um *framework* é caracterizada por uma “inversão de controle”. Essa arquitetura possibilita um processamento padronizado do gerenciamento de eventos que são invocados pelo mecanismo de envio do *framework*. Quando eventos ocorrem, o emissário do *framework* reage invocando métodos de gancho previamente registrados no objeto, que fazem um processamento específico da aplicação para os eventos. A inversão de controle possibilita ao *framework* (assim como toda a aplicação) determinar quais os conjuntos de métodos específicos da aplicação serão invocados em resposta aos eventos externos.

2.4.2 Problemas no uso de frameworks

Vários autores (Sparks, Benner, & Faris, 1996) (Bosch, Molin, Mattsson, Bengtsson, & Fayad, 1999) (Mattsson, Bosch, & Fayad, 2000) (Pree & Koskimies, 2000) (Casais, 1996) discutem alguns problemas associados à *frameworks*:

- O projeto de um *framework* é difícil. Em conseqüência da complexidade e tamanho dos *frameworks* de aplicação, e da dificuldade de compreensão do processo de projeto, *frameworks* são geralmente projetados iterativamente, amparados por metodologias de desenvolvimento, exigindo reestruturação substancial de inúmeras classes e ciclos de desenvolvimento;

- O processo de reuso é difícil. Convencionalmente, um *framework* consiste em um conjunto de classes básicas de um domínio de aplicação. Assim, o responsável pelo reuso deve compreender a arquitetura básica de um tipo de aplicação particular para ser capaz de especializar o *framework*;

- A combinação de *frameworks* é difícil. Freqüentemente um *framework* assume o controle da aplicação. Combinar dois ou mais *frameworks* que fazem isso, sem quebrar a integridade deles é muito difícil;

- A presença de classes desnecessárias nas aplicações instanciadas. Em geral, as aplicações instanciadas de um *framework* levam consigo todas as classes do *framework*, independentemente se todas elas são usadas ou não.

2.4.3 Classificações

Frameworks orientados a objetos podem ser caracterizados por diferentes dimensões. As mais importantes são: quanto ao domínio do problema, quanto à estrutura do *framework* e quanto ao seu uso (Fayad, Johnson, & Schmidt, 1999), (Froehlich G. , Hoover, Liu, & Sorenson, 1997), (Froehlich G. , Hoover, Liu, & Sorenson, 1997).

Quanto ao domínio do problema

- *Frameworks* de aplicação cobrem funcionalidades que podem ser aplicadas a diferentes domínios. Exemplo: *frameworks* de interfaces gráficas (GUI) (Weinand, Gamma, & Marty, 1989);

- *Frameworks* de domínio capturam o conhecimento e as peculiaridades de um problema específico. *Frameworks* de controle de produção e multimídia são exemplos de *frameworks* de domínio (Schmidt & Stephenson, 1995);

- *Frameworks* de suporte oferecem serviços de baixo nível em sistemas como, por exemplo, *drivers* para dispositivos e acesso a arquivos.

Quanto à estrutura

Se a descrição da estrutura interna do *framework* for visível, torna-se fácil o entendimento do seu comportamento. A estrutura interna de um *framework* está relacionada à concepção da arquitetura de software. O princípio da estrutura interna de um *framework* orientado a objetos é descrito pela arquitetura do *framework*. As principais arquiteturas para *frameworks* são (Buschmann, 1995):

- *Framework* com arquitetura em camadas: ajuda a estruturar aplicações que podem ser decompostas em grupos de sub-tarefas com diferentes níveis de abstrações;

- *Framework* com arquitetura *pipes and filters*: são utilizados para estruturar aplicações que podem ser divididas em várias tarefas complexas e independentes, que devem ser realizadas em alguma seqüência fortemente determinada ou em paralelo;

- *Framework* com arquitetura MVC *(Model View Controller*): define uma estrutura para aplicações interativas que separam a sua interface com o usuário do seu núcleo de funcionalidade;

- *Framework* com arquitetura PAC (*Presentation Abstraction Controller*): é uma arquitetura utilizada para software que possui muita interação com o usuário, possibilitando controles múltiplos da apresentação dos seus modelos de abstração que podem ser decompostos em sub-funções independentes;

- *Framework* com arquitetura reflexiva: é utilizado em aplicações que necessitam considerar adaptações futuras, como mudanças do ambiente, tecnologia, e requisitos, mas sem nenhuma alteração em sua estrutura e implementação;

- *Framework* com arquitetura *Microkernel*: é utilizado em sistemas de software que promovem diferentes visões de acordo com suas funcionalidades e que tenham que adaptar novos requisitos ao sistema. Ex.: sistemas operacionais;

- *Framework* com arquitetura *Blackboard*: ajuda a estruturar aplicações complexas que contêm diversos subsistemas especializados para diferentes domínios. Esses subsistemas cooperam entre si para construir a solução para o problema;

- *Framework* com arquitetura *Broker*: é utilizado em sistemas distribuídos, nos quais a interação de componentes é feita por chamadas remotas.

Quanto ao uso

Um *framework* orientado a objetos pode ser utilizado de três maneiras (tipo de reuso). O usuário deriva novas classes a partir de sua estrutura, combina classes já existentes ou faz as duas coisas. O primeiro método é conhecido como *architecture-driven* ou *white-box* (caixa branca) *framework*, o segundo como *data-driven* ou *black-box* (caixa preta) *framework* e o último como *gray-box* (caixa cinza) *framework* (Yassin & Fayad, 1999).

- *Framework* caixa branca: o reuso é provido por herança, ou seja, o usuário deve criar subclasses das classes abstratas contidas no *framework* para criar aplicações específicas. Para tal, ele deve entender detalhes de como o *framework* funciona para poder usá-lo. *Frameworks* desse tipo são mais difíceis de serem utilizados, uma vez que o seu usuário necessita conhecer detalhes de implementação e design;

- *Framework* caixa preta: o reuso é por composição, ou seja, o usuário combina diversas classes concretas existentes no *framework* para obter a aplicação desejada. Assim, ele deve entender apenas a interface para poder usá-lo. Configurar um *framework* selecionando componentes tende a ser mais simples do que herdar de classes já existentes. Desta forma, *frameworks black-box* tendem a ser mais fáceis de serem utilizados;

- *Framework* caixa-cinza: há uma mistura entre os *frameworks* caixa branca e caixa preta. Assim, o reuso é obtido por meio de herança, por ligação dinâmica e também pelas interfaces de definição.

2.4.4 Arquitetura básica e componentes de um *framework*

Como a Figura 2.3 mostra, um *framework* possui um *kernel* que contém um conjunto de classes que não são passíveis de adaptação e que permanecem presentes em todas as aplicações geradas. Todas estas classes possuem implementações reais de seus vários métodos e interfaces.

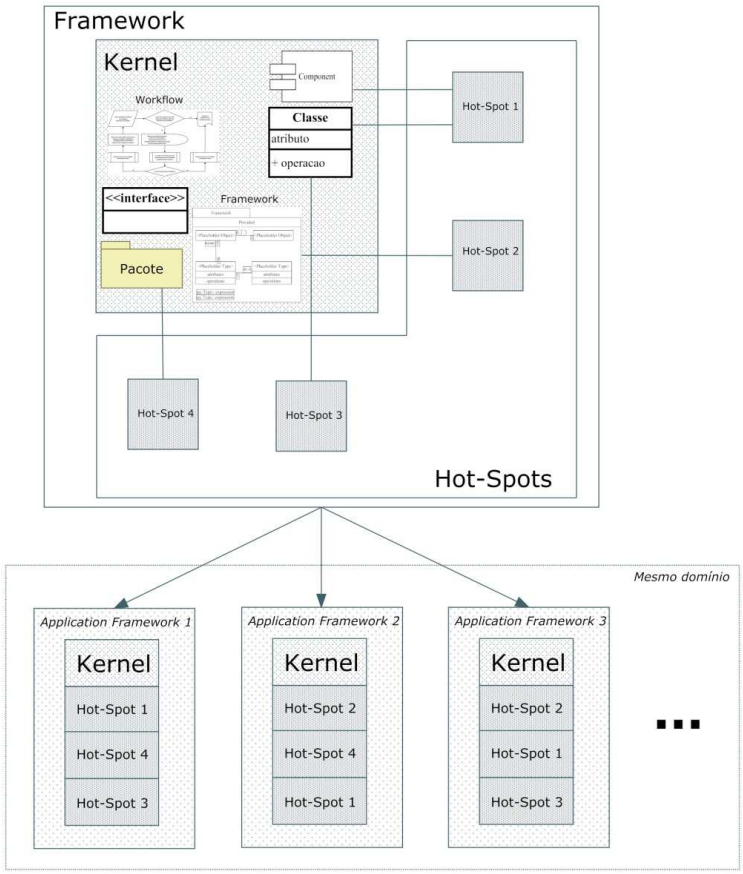


Figura 2.3 - Visão geral das partes básicas de um *framework*

O que diferencia, basicamente, as aplicações geradas são as diversas formas de se adaptar a estrutura básica do *framework* por meio de seus diversos pontos de flexibilização (*hot-spots*). E, conforme dito em 2.4, um *hot-spot* é uma classe abstrata que não possui implementação e deve ser especializada (customizada) para necessidades específicas da aplicação a ser gerada. A especialização pode ser realizada tanto por herança como por delegação, dependendo da forma como os *hot-spots* foram planejados.

2.4.5 Frameworks x Outras abordagens

Existem outras formas de reuso além de *frameworks* e padrões, como por exemplo componentes de software, geradores de aplicações e linhas de produto. Porém, elas não serão aqui abordadas em detalhes por estarem fora do foco específico deste trabalho, que é *framework*. Ainda assim, a seguir as definições desses termos e seu relacionamento (caso seja apropriado) com *frameworks*.

Componentes

Segundo definição proposta por (Werner & Braga, 2000), feita com base em definições de (Sametinger, 2001) e (Kruchten, 1998), “componentes reutilizáveis são artefatos auto-contidos, claramente identificáveis, que descrevem e realizam uma função específica e tem interfaces claras, em conformidade com um dado modelo de arquitetura de software, documentação apropriada e um grau de reutilização definido”. A distinção entre *frameworks* e componentes é feita principalmente considerando-se o fluxo de controle do programa, conforme visto na seção 2.4.1. Quando utiliza um componente, o desenvolvedor é responsável por sua chamada, ou seja, pelo fluxo de controle do programa, enquanto em um *framework* há a inversão de papéis: o programa principal é reutilizado e o desenvolvedor decide quais componentes são chamados e devem, portanto, ser derivados (Johnson & Foote, 1988). Em relação a padrões, é possível implementar um ou mais componentes com base em um padrão, já que o padrão possui um problema a ser resolvido sob determinadas influencias, o que seria traduzido, em termos de componente, para a funcionalidade específica do componente e sua interface.

Design *pattern* (padrões de projeto) orientado a objeto

Um *framework* difere, principalmente, de um design *pattern* em três aspectos:

- Um design *pattern* é mais abstrato do que um *framework*. Os *frameworks* têm a sua apresentação na forma de código (implementado), enquanto somente os exemplos de design *patterns* são apresentados de forma implementada;

- Os design *patterns* são arquiteturas menores do que os *frameworks*. Um *framework* pode conter vários designs *patterns*, mas o contrário não é possível;

- Os *frameworks* são mais especializados do que os designs *patterns*, sendo mais específico para um domínio de aplicação. Já os design *patterns* são mais gerais e podem ser aplicados em diversos domínios de aplicação.

Biblioteca de classes

Bibliotecas de classes são um conjunto de classes relacionadas que têm funcionalidades de propósito geral. As classes em uma biblioteca de classes não são relacionadas a um domínio de aplicação específico, como no caso de classes em um *framework*. A diferença entre uma biblioteca de classes e um *framework* é o grau de reutilização e o seu impacto na arquitetura da aplicação. Uma classe da biblioteca de classes são usados artefatos de software isolados, cabendo ao desenvolvedor estabelecer sua interligação, e no caso do *framework*, é procedida a reutilização de um conjunto de classes inter-relacionadas - inter-relacionamento estabelecido no projeto do *framework*. Uma diferença bastante evidente também na diferença entre biblioteca e *framework*, é exatamente o que a Figura 2.4 ilustra: inversão de controle. Ou seja, conforme relatado em 2.4.1, uma vez o *framework* é invocado, ele assume o controle da aplicação; algo que não ocorre com bibliotecas.

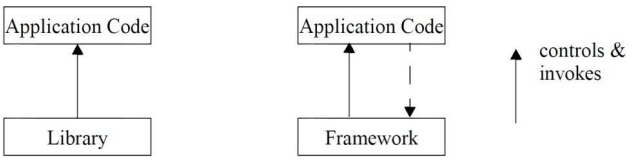


Figura 2.4 - Ilustração da inversão de controle em *frameworks* (Mattsson, 2000)

CMS

Um Sistema Gerenciador de Conteúdo (do inglês *Content Management Systems* - CMS) trata-se de um sistema gestor de portais Web integrando ferramentas necessárias para criar, gerir (editar e inserir) conteúdos em tempo real, sem a necessidade de programação de código (alguns CMS fazem até uso de *workflows* para definirem as regras em um alto nível). *Frameworks* diferem de CMS principalmente na flexibilidade para elaboração de ambientes: nos CMS os portais são construídos através de parametrização e, eventualmente, permite alguns componentes serem implementados; em *frameworks*, pode-se prover um conjunto de artefatos de modo a se facilitar o desenvolvimento, definições arquiteturais já feitas.

2.5 PROCESSO DE DESENVOLVIMENTO DE FRAMEWORKS

No processo de desenvolvimento de uma aplicação, cria-se um programa que atende a todos os requisitos, passando pelas fases de análise e projeto (*design*). Segundo (Mattsson, 1996), (Bosch, Molin, Mattsson, Bengtsson, & Fayad, 1999) e (Froehlich G. , Hoover, Liu, & Sorenson, 1997), o processo de desenvolvimento de um *framework* é mais difícil e trabalhoso, já que se estuda, agora, soluções para um conjunto de problemas de um determinado domínio. Na análise do domínio, observam-se as características comuns das diversas aplicações. Desenvolve-se o *framework*, e as diversas aplicações serão produzidas a partir da sua instanciação.

A Figura 2.5 relata em termos gerais como se dá o processo de desenvolvimento tradicional de aplicações. Basicamente é feita a análise, a construção em si (*design*) e é gerada a aplicação.

Figura 2.5 - Desenvolvimento tradicional de aplicações

Já em Figura 2.6 é ilustrado genericamente o processo de desenvolvimento de aplicações baseadas em *frameworks*. Como a partir de um *framework* se construirá diversas aplicações, é natural que o desenvolvimento deste seja mais complexo. Assim, é necessário e recomendado a adoção de alguma metodologia para amparar o desenvolvimento do *framework* proposto.

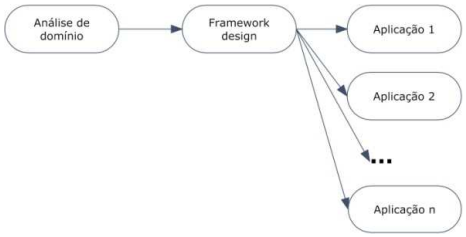


Figura 2.6 - Desenvolvimento de aplicações baseado em *frameworks*

Mas antes de descrever em si algumas metodologias de desenvolvimento de *frameworks*, será exposto alguns conceitos/abordagens que podem ser utilizados em

conjunto com as metodologias, servindo de apoio principalmente nas etapas de análise de domínio (ilustrada na Figura 2.6) assim como na transição desta etapa para a posterior.

2.5.1 *Language-driven*

Linguagem dirigida (*language-driven*) é uma linguagem de abordagem de alto nível para o *design* de aplicações. A abordagem é adequada para aplicações que seu conteúdo são constituídas com estruturas sofisticadas, e cujo comportamento interativo é impulsionado por estas estruturas. Os *designs* resultantes são passíveis de apoio de prototipagem rápida, a exploração e descoberta precoce dos recursos do aplicativo, a aplicação sistemática usando tecnologias padrão da Web, e racionais processos de colaboração entre especialistas e desenvolvedores durante a produção e manutenção.

Language-driven concebem o desenvolvimento de software como concepção, implementação e manutenção de um domínio específico. Um exemplo de *language-driven* amplamente difundido é SCORM[1], que é voltada para o domínio *e-learning*[2], contendo uma coleção de padrões/especificações do dado domínio.

2.5.2 *Model-driven*

Atualmente existe uma grande distância entre a etapa de elicitação de requisitos e o desenvolvimento (codificação) de sistemas computacional (Almeida, 2006), tornando, assim, o desenvolvimento de um aplicativo computacional demorado e custoso. O Desenvolvimento Orientado a Modelos é uma das possíveis soluções para aumentar a velocidade e diminuir o custo do desenvolvimento de um aplicativo computacional.

O Desenvolvimento Orientado a Modelos é uma abordagem de desenvolvimento de sistemas computacionais, que aumenta o poder da utilização de modelos para esta finalidade (OMG, 2003). Tal abordagem proporciona um meio de utilizar modelos no

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

[2] É o modelo de ensino não presencial suportado por tecnologia. Têm aumentado as possibilidades de difusão do conhecimento e da informação para os alunos e tornou-se uma forma de democratizar o saber para as camadas da população com acesso às novas tecnologias, permitindo que o conhecimento esteja disponível a qualquer hora e em qualquer lugar.

curso do entendimento, projeto, construção, implantação, operação, manutenção e modificação de sistemas computacionais (OMG, 2003). Os modelos são expressos em linguagens de modelagem que possuem sua sintaxe abstrata descritas em modelos conhecidos como metamodelos (Almeida, Dijkman, Pires, Quartel, & Sinderen, 2006).

2.5.3 Ontologias

De acordo com o dicionário Webster (Woolf, 1981), a palavra “ontologia” pode ser definida como uma teoria particular que diz respeito à natureza dos seres e das coisas em si. Já há bastante tempo, filósofos têm usado ontologias para descrever domínios naturais, ou seja, as coisas naturais do mundo como os tipos de existência e as relações temporais. Apesar de sua difusão e do longo tempo em que vem sendo usado, ainda não há um consenso (principalmente na comunidade de Ciência da Computação) sobre a semântica do termo “ontologia”. Em alguns casos, ele é usado apenas como um nome mais rebuscado, denotando o resultado de atividades familiares como modelagem de domínio e análise conceitual. No entanto, em muitos outros casos, as ditas ontologias apresentam algumas peculiaridades como a forte ênfase na necessidade de uma abordagem altamente formal e interdisciplinar, na qual a filosofia e a lingüística desempenham um papel fundamental (Guizzardi, 2000).

A conceituação tem uma importância fundamental em qualquer atividade de modelagem do conhecimento, pois é impossível representar o mundo real, ou mesmo uma parte dele, em sua completa riqueza de detalhes. Todo modelo de conhecimento é, portanto, comprometido com alguma conceituação, implícita ou explicitamente (Gruber, 1995). Para representar um certo fenômeno ou parte do mundo, a que chamamos domínio, é necessário concentrar a atenção em um número limitado de conceitos, suficientes e relevantes, para criar uma abstração do fenômeno em questão (Guizzardi, 2000).

(Gruber, 1995) define uma ontologia como sendo uma especificação de uma conceituação. (Guarino, 1998) estende essa definição dizendo que uma ontologia é na verdade uma especificação parcial e explícita que tenta, da melhor forma possível, aproximar a estrutura de mundo definida por uma conceituação. Uma ontologia, portanto, passa a ter compromisso apenas com a consistência em um determinado domínio e não com a completude. Ao conjunto de elementos de um domínio que podem

ser representados em uma ontologia é dado o nome de universo de discurso (Guizzardi, 2000).

O estudo de ontologias é crescente atualmente, favorecendo o aparecimento de muitas propostas fazendo o seu uso. Há classificações para ontologias, metodologias para seu desenvolvimento, meios para validá-las e inúmeros outros conceitos importantes relacionados. No entanto, ontologia não é o foco deste trabalho, assim, este tópico de estudo só é citado a caráter informativo, no entanto, pode ser utilizando na etapa de análise do domínio (Figura 2.6) para o desenvolvimento de *frameworks*.

2.5.4 Metodologia de desenvolvimento de *frameworks*

Conforme dito anteriormente, o desenvolvimento de *frameworks* é algo custoso o que leva a adoção de metodologias/práticas para guiarem o desenvolvimento do *framework*. Vale salientar que um processo de desenvolvimento de um *framework* depende do grau de experiência da organização no domínio do problema. Uma organização com maior experiência poderá adotar um processo mais avançado no desenvolvimento (Mattsson, Bosch, & Fayad, 2000) (Mattsson, 2000). E, atualmente existem algumas propostas de metodologias de desenvolvimento de *frameworks* que consistem procedimentos que abrangem a captura de informações de um domínio (podendo ser utilizados conceitos exemplificados em 2.5.1, 2.5.2 e em 2.5.3), a construção e o teste da estrutura de *frameworks*.

A seguir são descritas três metodologias voltadas ao desenvolvimento de *frameworks*: Processo convencional para o desenvolvimento de *frameworks*, Projeto Dirigido por Exemplo (*Example-Driven Design*) (Johnson, 1993), Projeto Dirigido por *Hot-Spot* (*Hot-Spot Driven Design*) (Pree, 1994). Estas metodologias se caracterizam por estabelecer o processo de desenvolvimento de *frameworks* em linhas gerais, sem se ater à definição de técnicas de modelagem ou detalhar o processo.

2.5.4.1 Processo convencional para o desenvolvimento de *frameworks*

O elementos comuns aos processos de desenvolvimento (ilustrado na Figura 2.7) são:

- Análise do domínio do problema;

- A primeira versão do *framework* é desenvolvida utilizando-se as abstrações encontradas;

- Uma ou mais aplicações são desenvolvidas baseadas no *framework*. O teste é importante para verificar se o *framework* é realmente reutilizável;

- Problemas encontrados no desenvolvimento da aplicação baseado em *framework* são capturados e resolvidos na sua próxima versão;

- Após repetir o ciclo várias vezes, o *framework* vai atingindo um nível de maturidade aceitável, possibilitando, desta forma, a sua reutilização por vários usuários.

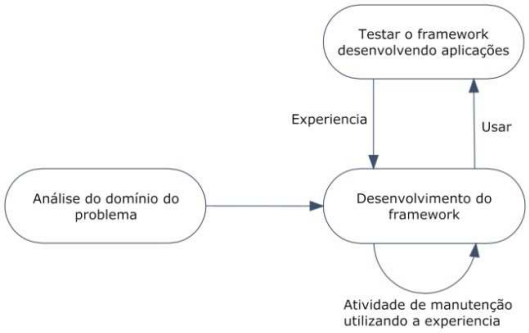


Figura 2.7 - Processo convencional de desenvolvimento de *frameworks*

Portanto, trata-se de um processo muito genérico e que usa como artifício as experiências das versões geradas. Fica evidente que falta etapas que incentivem o uso de padrões, por exemplo, algo imprescindível em *frameworks*.

2.5.4.2 Projeto Dirigido por Exemplo

Em (Johnson, 1993) é estabelecido que o desenvolvimento de um *framework* para um domínio de aplicação é decorrente de um processo de aprendizado a respeito deste domínio, que se processa concretamente a partir do desenvolvimento de aplicações ou do estudo de aplicações desenvolvidas. Porque as pessoas pensam de forma concreta, e

não de forma abstrata, a abstração do domínio, que é o próprio *framework*, é obtida a partir da generalização de casos concretos - as aplicações. Abstrações são obtidas de uma forma *bottom-up*, a partir do exame de exemplos concretos: aspectos semelhantes de diferentes aplicações podem dar origem a classes abstratas que agrupam as semelhanças, cabendo às classes concretas do nível hierárquico inferior a especialização para satisfazer cada caso.

O processo de generalização ocorre a partir da busca de elementos que recebem nomes diferentes, mas são a mesma coisa; recorrendo à parametrização para eliminar diferenças; particionando elementos para tentar obter elementos similares; agrupando elementos similares em classes.

Na Figura 2.8 é ilustrado a metodologia, com os seguintes passos:

- Análise do domínio: assimilar as abstrações já conhecidas; coletar exemplos de programas que poderiam ser desenvolvidos a partir do *framework* (no mínimo, quatro); avaliar a adequação de cada exemplo;

- Projetar uma hierarquia de classes que possa ser especializada para abranger os exemplos (um *framework*) - nesta etapa o autor recomenda a utilização de *design patterns*;

- Testar o *framework* usando-o para desenvolver os exemplos (implementar, testar e avaliar cada exemplo usado na primeira etapa, utilizando para isto, o *framework* desenvolvido).

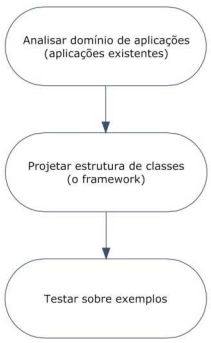


Figura 2.8 - Projeto Dirigido por Exemplo

Esta metodologia apresenta-se mais específica do que a descrita em 2.5.4.1, mas é focada em exemplos. Portanto, em domínios onde é restrito a existência de aplicações, esta metodologia acaba não sendo apropriada.

2.5.4.3 Projeto Dirigido por *Hot-Spots* (*Hot-Spot Driven Design*)

Uma aplicação orientada a objetos é completamente definida. Um *framework*, ao contrário possui partes propositalmente indefinidas - o que lhe dá a capacidade de ser flexível e se moldar a diferentes aplicações. Os *hot-spots* são as partes do *framework* mantidas flexíveis. A essência da metodologia é identificar os *hot-spots* na estrutura de classes de um domínio, e, a partir disto, construir o *framework*. (Pree, 1994) propõe a seqüência de procedimentos ilustrada na Figura 2.9.

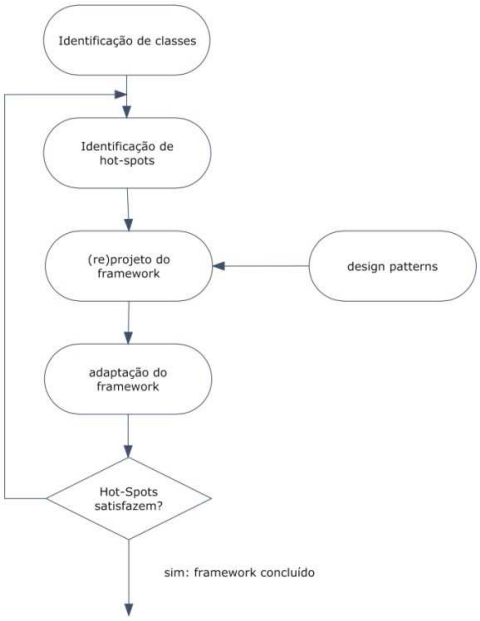


Figura 2.9 - Projeto Dirigido por *Hot-Spot*.Adaptado de (Pree, 1994)

Na primeira etapa, semelhante ao que é feito no desenvolvimento de uma aplicação orientada a objetos, é procedida a identificação de classes. O projetista do *framework*, a partir de informações de especialistas do domínio, define uma estrutura de classes.

Na segunda etapa, também com o auxílio de especialistas do domínio, são identificados os *hot-spots*. É preciso identificar que aspectos diferem de aplicação para aplicação, e definir o grau de flexibilidade que deve ser mantido em cada caso. Quando os aspectos

que diferem entre aplicações são os dados, a conseqüência é que classes abstratas agruparão os dados (atributos) comuns e classes concretas, específicas para aplicações, agruparão os dados específicos. Quando os aspectos que diferem entre aplicações são as funções, a conseqüência é a presença nas classes além dos métodos base (métodos completamente definidos), de métodos abstratos (em classes abstratas) que definem apenas o protocolo do método, métodos *template*, que chamam outros métodos (*hook*) para completar seu processamento e métodos *hook*, que flexibilizam o comportamento dos métodos *template*.

A terceira etapa, o projeto do *framework* (ou a re-elaboração do projeto), ocorre após a identificação dos *hot-spots*. Consiste em modificar a estrutura de classes inicialmente definida, de modo a comportar a flexibilidade requerida. Nesta etapa se definirá o que o usuário do *framework* deve fazer para gerar uma aplicação - que subclasses deve definir, a que classes deve fornecer parâmetros para a criação de objetos, quais as combinações de objetos possíveis.

A quarta etapa consiste num refinamento da estrutura do *framework* a partir de novas intervenções de especialistas do domínio. Se após isto o *framework* for avaliado como satisfatório, está concluída uma versão do *framework* (a questão é: os *hot-spots* definidos dão ao *framework* o grau de flexibilidade necessário para gerar as aplicações requeridas?), caso contrário, retorna-se à etapa de identificação de *hot-spots*.

Segundo (Pree, 1994), na etapa de projeto o desenvolvimento do *framework* é centrado em *design patterns,* cuja aplicação é fundamental para garantir flexibilidade ao *framework.*

2.5.4.4 Comparação entre metodologias

As metodologias descritas anteriormente apresentam características marcantes: o processo convencional toma como principal etapa a análise de domínio, o projeto dirigido por exemplo toma como foco exemplos de aplicações já desenvolvidas e o projeto dirigido por *hot-spots* toma como foco o provimento de flexibilidade (*hot-spot*). E cabe a característica do projeto em si para decidir a escolha mais adequada de metodologia utilizada.

As metodologias, entretanto, descrevem como produzir *frameworks* sem estabelecer um processo detalhado de construção de modelos que dirija o desenvolvimento, nem estabelecem um conjunto de mecanismos de descrição que contenham o projeto do *framework*. Mas não são inúteis devido a isso, elas são utilizadas, em conjunto com as boas práticas da Engenharia de Software, na condução do desenvolvimento do trabalho. Isso será detalhado no capítulo que será apresentado a formalização da proposta deste trabalho (Capítulo 5).

2.6 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo apresentou um referencial teórico que embasará a proposta deste trabalho. Primeiramente foi descrito uma contextualização da evolução do desenvolvimento de software, para realçar a importância de artefatos facilitadores na construção de software (no caso, *frameworks*). Posteriormente foi discutido detalhes de vários assuntos que permeiam um *framework*, desde sua definição, comparação com outras abordagens e até metodologias de desenvolvimento. Este levantamento e esclarecimento foi vital para a elaboração do trabalho.

Os conceitos levantados e descritos neste capítulo remetem-se a todos artefatos técnicos produzidos neste trabalho.

CAPÍTULO 3 DOMÍNIO DO PROBLEMA: AMBIENTES COLABORATIVOS

3.1 INTRODUÇÃO

A colaboração é necessária para que um grupo de pessoas consiga realizar tarefas interdependentes, com um objetivo como de alinhar e refinar idéias para a obtenção de êxito nas tomadas de decisões. E o computador tem sido cada vez mais utilizado com esse intuito, seja para ser uma ferramenta de trabalho para automação de processos, ou até, em alguns momentos, assumir papel determinante na comunicação entre indivíduos e na organização do trabalho em ambientes virtuais. Neste capítulo será feita uma discussão sobre essas questões, tendo sempre como foco de interesse os ambientes para apoio ao trabalho ou à aprendizagem realizados de forma coletiva. Este capítulo representa um ponto chave para o entendimento dos conceitos que serão descritos no capítulo posterior, que foram os alicerces deste trabalho.

Assim, o capítulo foi dividido da seguinte forma: a seção 3.2 apresenta uma contextualização sobre trabalho apoiado por computador; a seção 3.3 mostra algumas das principais categorias de ambientes virtuais, pertinentes a serem descritas devido as similaridades com o trabalho desenvolvido; a seção 3.4 ilustra alguns tipos de ambientes colaborativos; a seção 3.5 apresenta uma abordagem da colaboração segundo a ótica do Modelo 3C, que é o modelo de referência utilizado no desenvolvimento desde trabalho; e, por fim, a seção 3.6 apresenta alguns pontos críticos em Ambientes Colaborativos, evidenciando deficiências nos sistemas/projetos convencionais.

3.2 TRABALHO APOIADO POR COMPUTADOR

(Hammer, 1990) discute questões relacionadas à reengenharia do trabalho e das organizações num contexto onde o computador e várias tecnologias relacionadas já se faziam presentes, sustentando que pesados investimentos em Tecnologia da Informação frequentemente produzem resultados desanimadores devido ao fato de que as organizações tendem a usar a tecnologia para mecanizar antigas formas de trabalho,

deixando os processos existentes intactos, usando computadores simplesmente para torná-los mais velozes. Diz ainda que é comum que o trabalho seja organizado como uma seqüência de tarefas separadas e que sejam empregados complexos mecanismos para monitorar seu progresso.

Hammer mostrava que banco de dados compartilhados e rede de computadores poderiam apresentar duas vertentes: possibilitar diferentes tipos de informação a uma pessoa além de sistemas especialistas poderem ajudá-las nas decisões e a possibilidade de uma excessiva (ou inadequada) estruturação dessa comunicação vir a prejudicar a interação entre membros de um grupo.

Em (Lyytinen & Ngwenyama, 1992) é dito:

> *A integração de grupos de trabalho geograficamente dispersos foi apenas uma das possibilidades trazidas pelas ferramentas de comunicação, ajudando a tornar mais explícita a natureza multidimensional e a riqueza social na articulação dos processos de trabalho, características antes negligenciadas no desenvolvimento de aplicações da Tecnologia da Informação – situação que mudaria significativamente a partir de meados da década de 80.*

3.3 AMBIENTES VIRTUAIS

O virtual no contexto da Internet é mediado pelas tecnologias Web, sendo um produto da exteriorização de construções mentais em espaços de interação (Peirce, 1977). O espaço cibernético transforma um computador pessoal em uma espécie de computador coletivo, oferecendo um espaço de possibilidades infinitas para a navegação virtual (Lévy, 2000).

A Internet possibilitou/permitiu que cada membro fosse considerado como produtor e/ou consumidor de informações. A existência de ambientes que usuários interagem coletivamente bem como relacionam-se, podem proporcionar a formulação de novas

teorias e conceitos, além do amadurecimento de conhecimentos já adquiridos (Natalli & Menezes, 2010).

Como citado, a forma de operar sistemas está passando, hoje em dia, por uma transformação significativa. No modelo antigo, o individuo interagia com o computador para uma grande quantidade de tarefas apoiadas por sistemas computacionais e, quando precisava interagir com outras pessoas, usava recursos convencionais, como comunicação pessoal, reuniões, telefone etc. Na forma atual, tudo pode ser feito via recursos provindos dos ambientes virtuais. As pessoas interagem com os sistemas e outras pessoas usando os recursos computacionais de forma completa: solução das necessidades de automação, comunicação pessoal e interação com o grupo. Esta nova forma, permite que os membros de uma equipe compartilhem idéias, informações, realizando suas tarefas de uma maneira muito mais eficiente, além de potencializar os ganhos.

Nesta seção serão descritos algumas categorias de ambientes virtuais voltados à colaboração, que de alguma forma estão ligados com o desenvolvimento deste trabalho. Tal explanação é importante visto que o produto final deste trabalho é a produção de um *framework* para criação de Ambientes Colaborativos.

3.4 AMBIENTES COLABORATIVOS

Segundo (Brna, 1998), para falar de Ambientes Colaborativos, é preciso analisar a palavra colaboração. Uma das maneiras de analisar a colaboração é enxergá-la a partir da comunicação, coordenação e cooperação, conforme será discutido mais detalhadamente na seção 3.5. Aqui será descrito de maneira sucinta, para um breve entendimento a princípio.

Colaboração é uma maneira de trabalhar em grupo, onde os membros do grupo atuam em conjunto visando o sucesso do projeto, sendo que a falha de um dos participantes normalmente implica na falha do grupo como um todo (Grosz, 1996). Na colaboração, os participantes se ajudam objetivando o sucesso das tarefas e entre eles há uma hierarquia menos rígida, formando uma estrutura onde pares atuam em conjunto com objetivos comuns e compartilhados. De acordo com (Barros, 1994):

> *colaborar (co-labore) significa trabalhar junto, que implica no conceito de objetivos compartilhados e uma intenção explicita de somar algo – criar alguma coisa nova ou diferente através da colaboração, se contrapondo a uma simples troca de informação ou de instruções*.

Para ajudar a clarificação sobre o que são Ambientes Colaborativos, a seguir alguns termos empregados por pesquisadores da área:

- *Interação*: é uma forma de relacionamento onde há trocas e influência mútua. Dirigir em uma grande cidade, por exemplo, é uma atividade interativa, porém normalmente não é colaborativa. Nesta atividade, não há um objetivo compartilhado pelo grupo, não há um comprometimento com o sucesso do outro e não há uma negociação sobre um plano compartilhado. Por outro lado, ao dirigir em comboio, os motoristas combinam o caminho, os *checkpoints*, se comunicam por sinais, rádio ou telefone, e se um precisar de ajuda, os outros param. O sucesso do grupo é todos chegarem ao destino. Ao dirigir em comboio, há um comprometimento com o sucesso dos companheiros e um objetivo comum e compartilhado, caracterizando a colaboração (Grosz, 1996);

- *Trabalho em grupo*: é um conjunto de atividades com objetivo de atingir um determinado fim, produzindo um resultado. No trabalho em grupo, não necessariamente o interesse do participante é atingir o objetivo do trabalho, pode ser, por exemplo, receber um pagamento, não ser castigado, etc. Para caracterizar a colaboração é necessário saber as intenções e objetivos dos participantes (Brna, 1998);

- *Competição*: se assemelha em muitos aspectos à colaboração. Ela é de natureza interativa e há um objetivo comum, porém conflitante. Ao invés dos indivíduos se ajudarem, eles disputam entre si os recursos e o sucesso de um normalmente implica no fracasso dos outros. Apesar disto, os concorrentes se comunicam (pouco), coordenam-se, seguindo regras normalmente preestabelecidas, e atuam em conjunto em um espaço compartilhado. Mesmo dentro da competição, em alguns casos os participantes colaboram. Por exemplo, nas cadeias de

suprimento, os fabricantes de automóveis concorrentes se unem para definir padrões e fazer compras em conjunto, para cortar custos em aspectos comuns, sem prejudicar os diferenciais competitivos (Tapscoot, Ticoll, & Lowy, 2000);

- *Cooperação*: alguns pesquisadores diferenciam colaboração e cooperação de acordo com o grau de divisão do trabalho (Dillenbourg, 1999) (Roschelle & Teasley, 1995) (Brna, 1998). Na cooperação, os membros do grupo executam tarefas individualmente e depois combinam os resultados parciais para obter o resultado final. Na colaboração, os membros dos grupos trabalham juntos em um esforço coordenado (Dillenbourg & Self, 1992). No contexto deste trabalho, a cooperação é uma das atividades da colaboração (conforme já mencionado anteriormente, será abordado na seção 3.5).

Os ambientes que apresentam tais características serão citados neste trabalho como Ambientes Colaborativos. Obviamente trata-se de uma denominação muito ampla, assim, em seções posteriores é descrito algumas categorias de Ambientes Colaborativos.

3.4.1 CSCW e Groupware

A sigla CSCW vêm de *Computer Supported Cooperative Work* (Trabalho Cooperativo Apoiado por Computador) e é uma ciência multidisciplinar que estuda métodos e técnicas de trabalho em grupo auxiliadas por tecnologia e comunicação, abordando áreas como: psicologia, antropologia (estudos etnográficos), economia, sociologia, ciência da computação, teoria das organizações, ergonomia, aspectos culturais, etc. Emergiu desde então as equipes virtuais compostas por indivíduos que interagem entre si através das tecnologias de comunicação e ferramentas *groupware*.

O termo *groupware* designa a tecnologia (hardware e/ou software) gerada pelas pesquisas CSCW. Um dos requisitos fundamentais de *groupware* é que os sistemas sejam altamente configuráveis, para se adaptarem às necessidades dos usuários (Brooke, 1993).

A Figura 3.1 ilustra a classificação proposta por (Ellis, Gibbs, & Rein, 1991), segundo tempo e espaço.

	MESMA LOCALIZAÇAO	LOCALIZAÇAO DIFERENTE
MESMO TEMPO	Interação Síncrona Presencial - Sistemas de Apoio àTomada de Decisão - Sistemas de Apoio a Reuniões	Interação Síncrona Distribuída - Sistemas de Apoio àTomada de Decisão* - Sistemas de Apoio a Reuniões* - Editores Cooperativos - Sistemas de Comunicação Síncrona . Sistemas de instant message . Sistemas chat . Sistemas de videoconferência *Com recursos para acesso remoto.
TEMPOS DIFERENTES	Interação Assíncrona Presencial - Sistemas para Gerenciamento Eletrônico de Documentos (GED) - Workflow	Interação Assíncrona Distribuída - Sistemas para Gerenciamento Eletrônico de Documentos (GED)** - Workflow** - Editores Cooperativos - Sistemas de Comunicação Assíncrona . Correio eletrônico . Listas de discussão . Fóruns . Blogs - Área de Trabalho Compartilhada . BSCW . Quickplace **Com recursos para acesso remoto.

Figura 3.1 - Classificação Espaço x Tempo de Ellis para *groupware*. Adaptado de (Ellis, Gibbs, & Rein, 1991)

3.4.2 CSCL, LMS, AVA e outros

CSCL vêm de *Computer Supported Collaborative Learning* (Aprendizagem Colaborativa com Suporte Computacional) é a área de conhecimento que trata do suporte computacional às atividades de aprendizagem colaborativa. O CSCL cresceu em torno de um amplo leque de investigações sobre CSCW. O foco de CSCL está em apoiar a aprendizagem pela colaboração mútua, enquanto que as aplicações de CSCW facilitam a comunicação e produtividade em grupo; o conteúdo a ser comunicado direciona as pesquisas sobre CSCL, enquanto que CSCW preocupa-se com o processo de comunicação em si.

Ambientes Virtuais de Aprendizagem (AVA) são *groupwares* estudados pela área de pesquisa CSCL. No entanto, para esses há ainda outras nomenclaturas e classificações, que foram exploradas por (Watson & Watson, 2007), que define LMS, CMS e LCMS. Assim, um *Learning Management System* (LMS) é um *framework* que cuida de todos os aspectos do processo de aprendizagem (Watson & Watson, 2007), não se limitando apenas ao ambiente, mas também cuida das atividades e processos realizados nesse ambiente. No entanto, LMS é um termo muito utilizado na literatura para se referir a AVAs em geral. E um *Course Management System* (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 online e o subseqüente ensino e gerência do curso incluindo as interações dos estudantes do curso (EDUCAUSE Evolving Technologies Committee, 2003, p.1 apud Watson&Watson, 2007). Por fim, um *Learning Content Management System* (LCMS) é um sistema usado para criar, armazenar, montar e entregar conteúdo personalizado de *e-Learning* na forma de objetos de aprendizagem (Oakes, 2002, p. 73 apud Watson&Watson, 2007). Aqui, entende-se por *e-Learning* (aprendizagem eletrônica), como a aprendizagem realizada através de tecnologias digitais.

3.4.3 Arquiteturas pedagógicas

Segundo (Carvalho, Nevado, & Menezes, 2005):

> *As arquiteturas pedagógicas são, antes de tudo, estruturas de aprendizagem realizadas a partir da confluência de diferentes componentes: abordagem pedagógica, software educacional, internet, inteligência artificial, Educação a Distância, concepção de tempo e espaço. O caráter dessas arquiteturas pedagógicas é pensar a aprendizagem como um trabalho artesanal, construído na vivência de experiências e na demanda de ação, interação e meta-reflexão do sujeito sobre os fatos, os objetos e o meio ambiente sócio-ecológico.*

Ainda diz que:

> *As arquiteturas pedagógicas funcionam metaforicamente como mapas ao mostrar diferentes direções para se realizar algo, entretanto, cabe ao sujeito escolher e determinar o lugar para ir e quais caminhos percorrer. Pode-se percorrê-los individual ou coletivamente, ambas as formas são necessárias.*

No âmbito dessas arquiteturas, ambientes virtuais que as comportem implicam em 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. A visão da totalidade das produções individuais favorece o auto-acompanhamento (e também o acompanhamento do professor) e a meta-reflexão responsável pelas transformações nas formas de pensar. A autoria coletiva é facilitada pela interação e agilidade no acesso às produções de todos os participantes de uma comunidade de aprendizagem.

Em (Menezes, Nevado, Castro Junior, & Santos, 2008) é falado:

> *Ambientes com essas características permitem ações pedagógicas, se não libertas, pelo menos afastadas dos modelos orientados pela “lógica” do ensino transmissivo que interpõe fragmentações e “barreiras ou paredes virtuais” ainda mais resistentes que as paredes materiais e que muitas vezes dificultam a criação de propostas pedagógicas abertas e interativas, pela imposição de uma lógica reprodutivista, traduzida em uma organização rígida dos espaços.*

3.5 A COLABORAÇÃO VISTA PELO MODELO 3C

O Modelo 3C nasce do artigo seminal de (Ellis, Gibbs, & Rein, 1991). Tal modelo é utilizado para classificação do suporte computacional à colaboração. Nesta dissertação, o Modelo 3C é utilizado como base para a modelagem e desenvolvimento de *groupwares* e cada C é analisado. Há também uma diferença de terminologia; a operação conjunta no espaço compartilhado é chamada por Ellis de colaboração, enquanto no modelo 3C é chamada de cooperação. Em projetos de software esse modelo é amplamente utilizado na elaboração de *groupwares* e exemplos dessa utilização são exaustivamente discutidos em (Fuks, Raposo, & Gerosa, 2002), (Fuks, Raposo, & Gerosa, 2003), (Fuks, Raposo, Gerosa, & Lucena, 2005) e (Gerosa, Raposo, Fuks, & Lucena, 2006).

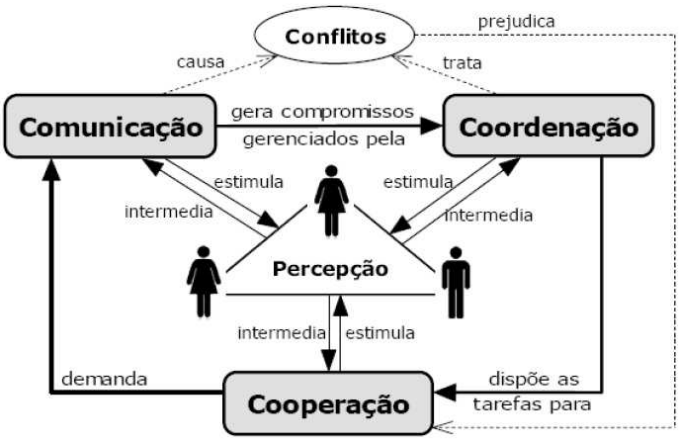


Figura 3.2 - Visão geral do modelo 3C (Fuks, Raposo, Gerosa, & Lucena, 2005)

Na Figura 3.2 é apresentada uma visão geral do Modelo 3C de colaboração e como seus elementos se relacionam, representando, dessa forma, a maneira pela qual a colaboração ocorre entre indivíduos. A colaboração é baseado na premissa de que, para colaborarem indivíduos necessitam realizar ações de comunicação, coordenação e cooperação. No modelo, é ilustrado como a comunicação é utilizada pela cooperação e coordenação, gerando compromissos e possibilitando a colaboração. Já a coordenação fornece as regras e protocolos necessários, para que indivíduos comuniquem-se e cooperem de forma harmoniosa. Nas próximas 3 subseções será discutido os 3C do modelo acima.

3.5.1 Comunicação

Durante a comunicação, as pessoas almejam construir um entendimento comum e compartilhar idéias, discutir, negociar e tomar decisões. Os participantes de uma equipe de trabalho devem se comunicar para conseguir realizar tarefas interdependentes, não completamente descritas ou que necessitem de negociação (Fussell, Kraut, Learch, Scherlis, McNally, & Cadiz, 1998).

Delvin e Rosenberg ressaltam a importância do conhecimento individual e das práticas cooperativas, como a linguagem das mãos na comunicação face-a-face, que as pessoas desenvolvem de forma a coordenar a variedade de conhecimentos individuais e atingir o entendimento mútuo. O contexto cultural, o domínio em questão e os conhecimentos individuais influenciam como as expressões de linguagem são produzidas pelo comunicador e interpretadas pelo receptor (Delvin & Rosemberg, 1996).

Como o ambiente define o espaço compartilhado de informação entre os indivíduos, ele pode fornecer elementos adicionais não-verbais à estrutura de linguagem utilizada na conversação. Isto simplifica a comunicação verbal, que é complementada pelos elementos presentes no ambiente (Gutwin & Greenberg, 1999).

As informações são transmitidas através de um canal de percepção (termo também encontrado na literatura como *awareness*) criado no espaço compartilhado onde ocorre a conversação. Este canal de percepção fica implícito no canal de comunicação. Para haver entendimento e a comunicação cumprir seu objetivo, é necessário o conhecimento de todos sobre a utilização das mídias de transmissão e de recebimento dos dados, bem como a participação ativa do receptor, que deve estar atento às informações transmitidas e aos elementos utilizados, para que seja viabilizado o canal de percepção. Dada a importância da percepção para viabilizar o canal de transmissão de informação na comunicação, deve-se projetar e avaliar cuidadosamente nos ambientes virtuais colaborativos os elementos disponíveis para o emissor codificar sua mensagem e os elementos de percepção para que o receptor receba a mesma.

Ao contrário de algumas situações onde o importante é saber apenas se o receptor recebeu uma mensagem, na colaboração é importante assegurar-se do entendimento da mesma. Sem um entendimento compartilhado, os participantes terão dificuldade em se coordenar de modo a somar seus esforços para a conclusão das tarefas. Porém, não há como inspecionar se o conteúdo recebido é equivalente ao enviado e se ele foi assimilado pelo receptor. A única forma de se obter indícios do entendimento é através das ações (e reações) do receptor, pois as mesmas são guiadas por seus conhecimentos. Uma falha na comunicação seria então uma discordância entre as expectativas do emissor e as ações do receptor. Algumas vezes estas falhas são identificadas no discurso do receptor, e outras, em suas atitudes.

3.5.2 Coordenação

Conversação para ação gera compromissos (Winograd & Flores, 1987) (Winograd, 1988). Para garantir o cumprimento destes compromissos e a realização do trabalho colaborativo através da soma dos trabalhos individuais, é necessária a coordenação das atividades. Esta coordenação organiza o grupo para evitar que esforços de comunicação

e cooperação sejam perdidos e que as tarefas sejam realizadas na ordem correta, no tempo correto e cumprindo as restrições e objetivos (Raposo, Magalhães, Ricarte, & Fuks, 2001).

A coordenação envolve tanto a pré-articulação das atividades, que corresponde às ações necessárias para preparar a colaboração, normalmente concluídas antes do trabalho colaborativo se iniciar, e o gerenciamento do aspecto dinâmico da colaboração, renegociada de maneira quase contínua ao longo de todo o tempo. Olhando apenas para esse aspecto dinâmico e contínuo da coordenação, ela pode ser definida como “o ato de gerenciar interdependências entre as atividades realizadas para se atingir um objetivo” (Malone & Crowston, 1990). Apesar da interdependência normalmente positiva entre as tarefas na colaboração (um participante desejando que o trabalho do outro seja bem sucedido), ela nem sempre é harmoniosa. Sem coordenação, há o risco de os participantes se envolverem em tarefas conflitantes ou repetitivas.

Algumas atividades envolvendo múltiplos indivíduos não exigem um planejamento formal. Atividades ligadas às relações sociais são bem controladas pelo chamado protocolo social, caracterizado pela ausência de qualquer mecanismo de coordenação explícito entre as atividades e pela confiança nas habilidades dos participantes de mediar as interações. Exemplos de ferramentas para este tipo de atividade são a maioria dos *chats* e as áudio e videoconferências. Por outro lado, atividades mais diretamente voltadas para o trabalho colaborativo (e não para as relações sociais) exigem sofisticados mecanismos de coordenação para garantir o sucesso da colaboração. Exemplos de ferramentas que dão suporte a este tipo de atividade são gerenciamento de fluxo de trabalho (*workflow*), jogos multiusuários e ferramentas de autoria e de desenvolvimento de software colaborativo. Na prática, entretanto, nem sempre é claro o que deve ficar a cargo do protocolo social e o que deve ter um mecanismo de coordenação associado. É tarefa do desenvolvedor de *groupware* a decisão sobre como será feita a coordenação de cada uma das atividades a serem realizadas. O ideal é que sistemas colaborativos não imponham padrões rígidos de trabalho ou de comunicação (Schmidt K. , 1991).

Para a coordenação do grupo são essenciais informações de percepção. É importante que cada um conheça o progresso do trabalho dos companheiros: o que foi feito, como

foi feito, o que falta para o término, quais são os resultados preliminares, etc. As informações de percepção são necessárias principalmente durante a fase dinâmica da coordenação, para transmitir mudanças de planos e ajudar a gerar o novo entendimento compartilhado. Elas ajudam a medir a qualidade do trabalho com respeito aos objetivos e progressos do grupo e a evitar duplicação desnecessária de esforços (Dourish & Belloti, 1992).

3.5.3 Cooperação

No trabalho colaborativo, a cooperação é o esforço conjunto em um ambiente compartilhado para alcançar um determinado objetivo (Oliveira, Antunes, & Guizzardi, 2007). O ambiente pode fornecer ferramentas de gerenciamento destes artefatos, como por exemplo, registro e recuperação de versões, controle e permissões de acesso, etc.

O registro da informação visa aumentar o entendimento entre as pessoas, reduzindo a incerteza (relacionada com a ausência de informação) e a equivocalidade (relacionada com a ambigüidade e com a existência de informações conflitantes) (Daft & Lengel, 1986). Os indivíduos trabalham as informações e se comunicam na tentativa de solucionar os desentendimentos. A forma de garantir a "memória" do grupo nos projetos colaborativos é preservando, catalogando, categorizando e estruturando a documentação produzida pelos participantes.

Os indivíduos buscam nos elementos de percepção as informações necessárias para criar um contexto compartilhado e antecipar ações e necessidades com relação às metas da colaboração. Isto possibilita identificar as intenções dos companheiros do grupo, de forma a tornar possível prestar assistência ao trabalho deles quando for possível e necessário. Estas interações geram novos acontecimentos e informações no espaço compartilhado, que por sua vez irão se refletir nos elementos de percepção. Neles os indivíduos buscarão conhecimentos para se comunicar e coordenar interações posteriores. Deve-se prever onde informações de percepção são relevantes, como elas podem ser obtidas ou geradas, onde elementos de percepção são necessários, como apresentá-los e como dar aos indivíduos o controle sobre eles. O excesso de informações pode causar sobrecarga e dificultar a colaboração. Para evitar a sobrecarga,

é necessário balancear a necessidade de fornecer informações com a de preservar a atenção sobre o trabalho (Fuks, Raposo, & Gerosa, 2002).

3.6 AMBIENTES COLABORATIVOS - ASPECTOS CRÍTICOS

Diversos autores têm publicado materiais avaliando as diversas ferramentas e ambientes para construção de Ambientes Colaborativos. Têm ficado cada vez mais claro que algumas dificuldades que decorrem da estrutura destes ambientes estarem apoiadas pelo conceito de “ferramentas”. Isto significa que ambientes convencionais são concebidos para propósitos específicos e restritos, o que implica na necessidade de contornar tais restrições estruturais para desenvolver propostas de realmente diferenciadas.

Assim, de modo geral, os Ambientes Colaborativos podem ser criticados sob vários aspectos, como podemos encontrar na literatura especializada (Fioravanti, Nakagawa, & Barbosa, 2010), (Dodge, 2004), (Dougiamas, 2009), (Vieira Júnior, Santos, Rafalski, Bada, Silva, & Menezes, 2011), (González, 2005), (Menezes, Nevado, Castro Junior, & Santos, 2008), (Beltrame, Monteiro, Rangel, & Cury, 2008), (Natalli & Menezes, 2010), (Nevado, Dalpiaz, & Menezes, 2009), (Nevado, Carvalho, & Menezes, 2007), dentre outros. Com base nestas literaturas identificamos alguns pontos a ponderar na adoção ou concepção desses ambientes:

Cronograma de atividades (*workflow*)

A possibilidade de configurar prazos para ações de usuários pode facilitar a concepção de ambientes. Exemplo: prazo para postagem em um determinado tópico em um fórum. Isso irá ficar mais claro no próximo capítulo (Capítulo 4), uma vez que será apresentado uma forma de concepção diferenciada de ambientes. Outra questão é o oferecimento de mecanismos para encadearem atividades, com prioridades, sequenciamento, etc. Trata-se de um requisito muito difícil de se encontrar em Ambientes Colaborativos.

Papéis e permissões de acesso

Em geral, os papéis[3] disponíveis são fixos. Softwares voltados para a educação, por exemplo, oferecem os papéis de professor e estudante por padrão, porém, papéis mais específicos, como monitores, revisores, visitantes nem sempre são oferecidos. Ou até mesmo outros papéis que podem ser necessários de acordo com a demanda.

Alto acoplamento entre dados e ambientes

Os dados/informações produzidos pelos usuários que frequentam o ambiente em questão são fortemente acoplados aos ambientes que pertencem. Exemplo: um *post* em um *blog* está fortemente acoplado ao *blog* que recebeu um *post*. Se o *blog* for apagado, o *post* também será apagado.

Arquitetura e organização do código-fonte

Uma das formas de se adaptar um sistema é por meio da customização do seu código-fonte. Entretanto, essa alternativa torna-se inviável quando o código-fonte do sistema não está disponível, quando a alteração do código é dificultada em função da forma como o sistema foi desenvolvido ou quando não há documentação suficiente sobre o código e arquitetura do sistema.

Em geral, muitos dos sistemas educacionais *open-source*, como por exemplo o *TelEduc*, apresentam um conjunto de funcionalidades, implementadas sobre arquiteturas integradas que dificultam sua alteração. Em geral isso acontece porque esses sistemas foram projetados para atender um conjunto de objetivos bem definidos inicialmente, e não para serem moldados ao gosto do usuário. Pequenas alterações em pontos específicos desses sistemas podem afetar o comportamento do sistema se o desenvolvedor não conhecer detalhes sobre todo o código. Isso pode ser evidenciado, no *AulaNet* em (Pimentel, et al., 2005), que retrata um trabalho para refazer a estrutura interna de modo a permitir o acoplamento de novas funcionalidades e até mesmo facilitar a manutenção, devido a complexidade do código.

---

[3] Papéis (*Roles*) são rótulos que identificam classes de usuários de um sistema, por exemplo, administrador, professor, estudante, etc. Os papéis são caracterizados por um conjunto de permissões sobre as funcionalidades do sistema. Essas permissões definem o que o usuário, portador do papel, pode fazer no sistema.

Plataforma de execução e sistemas gerenciadores de banco de dados

A dependência de plataforma de um sistema educacional pode ter influência sobre a escolha dos usuários por um sistema ou outro. Há sistemas que podem ser utilizados em servidores Linux e há outros que em servidores Windows. Mas usuários podem desejar que seja independente de plataforma, o que seria um grande problema para a grande maioria dos ambientes existentes. Também existe a questão da dependência de algum SGBD (Sistema Gerenciador de Banco de Dados) específico, podendo também influenciar nas escolhas de usuários e fazer que o projeto do sistema não atinja o sucesso esperado. Há sistemas que utilizam o SGBD Oracle[4], SQL Server[5], MySQL[6], etc, não sendo independente de SGBD, o que seria a situação ideal.

3.7 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo apresentou um arcabouço teórico acerca do termo colaboração e suas adjacências, servindo de base assim, para elaboração do produto final deste trabalho (proposta de um *framework* para criação de Ambientes Colaborativos). Como o *framework* proposto é voltado para Ambientes Colaborativos (formalizado no Capítulo 5), é vital fazer tal explanação de conceitos como a realizada neste capítulo.

No início do capítulo foi abordado os primórdios de trabalhos utilizando computador e sua evolução até o nascimento de *groupwares*. Após isso, na seção 3.3, foi discutido algumas das principais nomenclaturas de Ambientes Colaborativos bem como algumas possíveis classificações e características.

O capítulo também fez uma discussão da colaboração sob a ótica do Modelo 3C, que é um modelo amplamente difundido na literatura e usado em larga escala em projetos do gênero. Por fim, o capítulo mostrou alguns trabalhos correlatos como forma de materializar os conceitos antes mencionados, além de evidenciar algumas deficiências.

---

[4] http://www.oracle.com/

[5] http://www.microsoft.com/sqlserver/

[6] http://www.mysql.com/

CAPÍTULO 4 UMA ABORDAGEM INOVADORA PARA CONCEPÇÃO DE AMBIENTES COLABORATIVOS

4.1 INTRODUÇÃO

Neste capítulo será descrito uma abordagem inovadora para construção de Ambientes Colaborativos, de modo que se potencialize o projeto de ambientes inovadores, introduzindo facilidades conceituais para tal feito.

Foi definido a seguinte estruturação para o capítulo: a seção 4.2 mostra, com base em temas descritos em capítulos anteriores, que há uma demanda crescente por ambientes flexíveis, de modo que potencialize as idealizações dos usuários ao invés de se colocar barreiras; na seção 4.3 é exposto a abordagem, em detalhes sob algumas perspectivas, de construção de Ambientes Colaborativos que fora proposta por (Menezes, Nevado, Castro Junior, & Santos, 2008); a seção 4.4 aborda Vcoms (Veículo de comunicação, será descrito neste capítulo) de um modo a se potencializar a flexibilidade; a seção 4.5 modela alguns ambientes virtuais segundo os conceitos descritos anteriormente, servindo como forma de prova (evidenciação); e, por fim, a seção 4.6 apresenta as conclusões dessa nova abordagem.

4.2 DEMANDA DE AMBIENTES FLEXÍVEIS

Atividades relacionadas com o trabalho, os negócios, o lazer, a filantropia, a aprendizagem e outras, podem ocorrer em ambientes digitais. Embora visem diferentes objetivos, todas elas requerem facilidades para autoria cooperativa, aquisição e socialização de conhecimento.

A Internet está enormemente povoada por ferramentas de comunicação e interação, que são implementadas por sistemas virtuais que se encaixam na categoria de CMS ou de LMS (resultando em um AVA). Das ferramentas usuais pode ser citado: fóruns, emails, *chats*, murais, *wikis*, *blogs*, FAQs, *webfólios*, etc. As dificuldades impostas pelos ambientes virtuais integrados determinaram a concepção de sistemas de computador

específicos e rígidos, conforme descrito em (Fagundes, Nevado, Basso, Bitencourt, Menezes, & Monteiro, 2006) e (Monteiro, Menezes, Nevado, & Fagundes, 2005). Tais limitações também são evidenciadas em (González, 2005). Em geral, cada ambiente permite a configuração de uso para um elenco restrito de ferramentas, de estrutura predefinida, com pequenas facilidades de configuração. Uma hipótese que pode ser levantada é que as ferramentas hoje disponíveis ainda seguem os modelos surgidos historicamente e baseados nas possibilidades disponíveis na Web quando foram criadas.

É sabido que quando uma nova idéia surge, para ferramentas dessa natureza, ela em geral é implementada em ferramentas específicas, independente dos ambientes virtuais, e para que se possa usá-las nesses ambientes é necessário aguardarmos a implementação pela equipe de programação. Por outro lado, à medida que um sistema desses ganha popularidade seus idealizadores buscam perpetuá-lo como sistema independente, incorporando facilidades dos sistemas convencionais. A titulo de ilustração, tomemos como exemplo o caso do *blog*, uma ferramenta bastante conhecida hoje, muito usada por profissionais ou amadores de diferentes ramos. À medida que o *blog* foi se popularizando, foram incorporando outras ferramentas tais como fóruns, marcadores, enquetes e outros. É comum encontrar um *blog*, com recursos adicionais mas só que projetado para um ambiente específico, sem uma maior flexibilidade de implantar-se em um outro local, ou customizar determinadas funcionalidades.

Segundo (Menezes, Nevado, Castro Junior, & Santos, 2008):

> *[...] em vista a plasticidade dos ambientes virtuais e as novas possibilidades de automatização de tarefas mecânicas, há necessidade de buscar-se uma nova concepção para criação de ambientes virtuais, pautados pelos seguintes aspectos: plasticidade, ergonomia, redução da repetição de trabalho e redução da sobrecarga cognitiva.*

Em (Beltrame, Monteiro, Rangel, & Cury, 2008), é dito ainda:

> *[...] viabilidade/demanda de se desenvolver sistemas Web capazes de gerenciar as idealizações ("sonhos") de espaços virtuais coletivos, de forma a não limitar a criatividade à percepção inicial dos programadores e possibilitar aos agentes do ambiente virtual o desenvolvimento contínuo do espaço de acordo com as suas cognições e suas individualidades.*

Buscando criar condições para que a apropriação do mundo virtual seja realizada de forma mais rápida e efetiva, precisa-se repensar a concepção destes mundos e ao mesmo tempo oferecer objetos mais adequados a esta apropriação assim como dar suporte à percepção das transformações e dos resultados das interações que fazemos com estes objetos.

4.3 MORFEU

A seção anterior deixa claro que para facilitar que a apropriação do mundo virtual seja realizada de forma mais rápida e efetiva, precisamos repensar na concepção destes. E é baseado nessa hipóteses que surge o MOrFEu, proposto por (Menezes, Nevado, Castro Junior, & Santos, 2008). A partir daqui, vamos abstrair o MOrFEu como um ambiente virtual, para facilitar o entendimento.

MOrFEu é um acrônimo para "**M**ulti-**O**rganizador **F**lexível para **E**spaços virt**U**ais", que é um trabalho que busca dar essa flexibilidade requerida ao usuário. Seguindo a concepção para modelagem de espaços virtuais flexíveis proposto em (Menezes, Nevado, Castro Junior, & Santos, 2008), apresentamos a seguir uma concepção de Ambientes Colaborativos, ilustrado através da Figura 4.1. O registro das produções individuais é realizado através de um ambiente de autoria (Parte 1 da Figura 4.1). Cada item de registro deve possuir, pelo menos, um autor, um título e um conteúdo (corpo). A qualquer instante o usuário pode criar ou editar um elemento básico de autoria, sem a preocupação do destino que dará a ela. A cada edição resulta em uma nova versão. Uma versão pode ser posteriormente usada e reusada em situações de interação. A idéia é armazenar as autorias (e naturalmente seus desdobramentos – versões) de forma independente e desacoplada de ferramentas de interação.

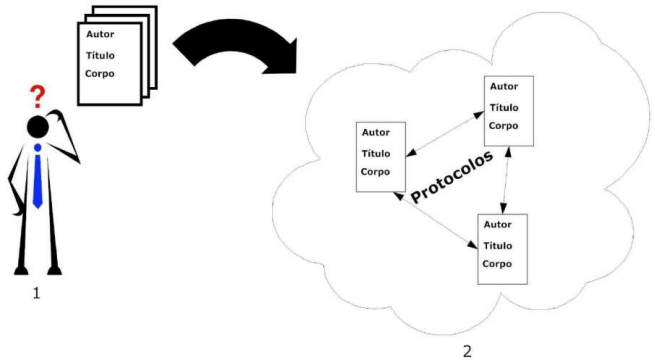


Figura 4.1 - O autor cria elementos de autoria (1) e, possivelmente, os publica em um ambiente virtual (em 2) regidos por um protocolo de interação específico (Natalli & Menezes, 2010)

Outra questão importante é que os elementos de autoria podem, possivelmente, ser agrupados por um ambiente de colaboração (ver Parte 2 da Figura 4.1). Cada ambiente de colaboração possui suas diretrizes de composição (protocolos a serem seguidos – protocolos estes criados no momento da confecção do ambiente). Ou seja, basicamente tudo se resume a artefatos estruturados e agrupados/conectados segundo algum protocolo.

É esperado também que toda produção do usuário seja registrada e versionada, independente das disponibilizações em ambientes virtuais, ao contrário dos ambientes convencionais onde o conjunto das produções individuais fica atrelado às ferramentas – uma mensagem postada em um fórum faz parte do acervo daquele fórum, se o fórum for excluído, o autor perde a mensagem. Da mesma forma acontece com as mensagens escritas em um *chat*, por exemplo. Em tal concepção, todos as produções dos usuários são materializadas através do elementos básico de autoria e ficam, antes de mais nada, registrada em um agregador, associada a cada indivíduo (autor).

4.3.1 Principais conceitos

No trabalho exposto por (Menezes, Nevado, Castro Junior, & Santos, 2008), foram apresentados alguns dos principais conceitos presentes no MOrFEu. Abaixo segue:

UPI

Um elemento básico de autoria, mencionado anteriormente, no MOrFEu é chamado de UPI (vem de **U**nidade de **P**rodução **I**ntelectual). A UPI, como já dito, é composta no

mínimo de um corpo, um conteúdo e um autor. Obviamente, para representar produções mais elaboradas, como um vídeo, pode ser pensado em atributos adicionais em uma UPI. Um usuário pode criar/editar uma UPI em qualquer momento sem a preocupação no destino que dará a ela, sendo esta sempre armazenada suas versões. Pode também usá-la e reusá-la em situações de iterações, tais como: uma UPI utilizada para representar uma mensagem para um usuário em um *chat* pode ser usada para representar uma mensagem em um fórum.

Todo o conteúdo gerado pelos usuários será armazenados na forma de UPI, além de não terem vinculação forte com o ambiente que a utiliza, somente sendo uma referência (diferencial em relação aos ambientes tradicionais).

Editor de UPI

Ferramenta através do qual o usuário poderá criar e editar UPIs sem necessariamente definir o destino final para ela. O ato de um usuário criar/editar uma UPI não significa que ela necessariamente deve ser colocada (publicada) em um ambiente (Vcom). A criação de uma UPI pode ser um simples pensamento, algo momentâneo que o usuário queira manter sem publicação alguma, e, em um momento posterior, possa vir a desejar vinculá-la em um Ambiente Colaborativo (Vcom).

Minhas UPIs

Todas produções dos usuários (UPIs) ficam unidas através do agregador chamado "Minhas UPIs", sendo estas vinculadas ou não comum (ou mais) Ambientes Colaborativos (Vcom). O objetivo do agregador é recuperar UPIs do usuário permitindo que ele as vincule mais facilmente em um ambiente. Na recuperação ser utilizadas técnicas de recuperação inteligente de informação, provendo agentes inteligentes[7] que auxiliem a busca, dentre outros mecanismo que auxiliem a gerência do conhecimento.

[7] Um agente é um sistema de computador que está situado em algum ambiente e que é capaz de executar ações autônomas de forma flexível neste ambiente, a fim de satisfazer seus objetivos de projeto (Wooldridge, 1999).

Veículos de comunicação (Vcom)

Os ambientes de colaboração são chamados no MOrFEu de Vcom (**V**eículos de **com**unicação). Um Vcom pode conter referências/vinculações de UPIs, além de terem definições (diretrizes) de composição (protocolo de iteração). O protocolo de iteração estabelece as definições (regras) gerais do Vcom. Por exemplo, assumindo que um *blog* seja um Vcom, os *posts* seriam UPIs organizadas de modo cronológico "inverso" (as mais recentes aparecem primeiro). *Posts* poderiam ter respostas, que estas poderiam se relacionar e estarem organizadas sob a forma de árvore além de serem hierarquicamente inferior a um *post* (as hierarquia entre UPIs também seria um exemplo de uma definição em um Vcom).

Outro detalhe importante a ser comentado é que para um Vcom se tornar disponível (ou acessível) ele deve ser instanciado. Ou seja, no MOrFEu pode haver um Vcom denominado *blog*, e várias instâncias de tal Vcom disponíveis para acesso.

Template

Cada instância de Vcom no MOrFEu deve ter um *template* associado, sendo este responsável por definir a forma visual do Vcom. A forma visual de um Vcom embloga estilos visuais, *design*, cores, disposição e outras características que componentes visuais podem assumir. É importante ressaltar que o *template* não possui ligação com os UPIs presentes no Vcom. Uma analogia que poderia ser feita com *templates* de Vcom são folhas de estilos CSS; entretanto, os *templates* vão além dos elementos tratados em uma folha de estilo, eles definem por qual conteúdo é formado e não somente de que forma é apresentado este conteúdo, como nas folhas de estilo.

Editor de *Template*

Para facilitar a customização/personalização do visual geral de um Vcom, o MOrFEu pode prover um mecanismo para facilitar a edição de *templates*. Tal mecanismo é o denominado "Editor de *Template*", sendo um recurso que o usuário pode utilizar para alterar o *template* sem necessidade de programação, apenas realizando simples ajustes visuais.

Biblioteca de Vcom

Trata-se de um conjunto de Vcoms que o MOrFEu permite instanciar. Um Vcom instanciado é um Ambiente Colaborativo, já a Biblioteca de Vcom é um conjunto de possibilidades de Vcoms que podem ser instanciados pelos usuários. Outro ponto que merece ser enfatizado é que o usuário não fica limitado a usar somente os Vcoms disponíveis na Biblioteca de Vcom, o MOrFEu permite mecanismos facilitadores para criação de novos Vcoms.

A Tabela 1 apresenta de forma resumida os conceitos descritos anteriormente bem como seus significados no projeto.

Tabela 1 - Resumo dos principais conceitos presentes no MOrFEu

Conceito	Significado
UPI	Abstração para representar o elemento básico de autoria.
Editor de UPI	Ferramenta que permite criação/edição de UPIs.
Minhas UPIs	Agregador de UPIs de um dado usuário, é através deste que usuários vinculam UPIs nos Vcoms instanciados.
Vcom	Abstração de Ambiente Colaborativos.
Template	Modelo de apresentação visual de um Vcom.
Editor de *Template*	Ferramenta facilitadora na edição de *templates* de Vcom.
Biblioteca de Vcom	Conjunto de “modelos” de Vcoms.

4.3.2 Principais interações

Aqui será descrito algumas das principais maneiras de interação por parte dos usuários, no MOrFEu:

Publicar UPI

Trata-se da interação de vincular (referenciar) uma UPI a um Vcom. Vale ressaltar, conforme dito anteriormente, o Vcom apenas armazena uma referência para uma UPI. Caso o Vcom seja excluído, não implica em exclusão da UPI. A publicação pode ser feita automaticamente e transparente ao usuário no momento da interação com o Vcom ou pode ser utilizando o agregador “Minhas UPIs”.

Versionar UPI

Ato de editar uma UPI já criada. Tal processo pode feito de maneira transparente ao usuário, no momento que ele realiza a interação com um Vcom. Ou UPIs podem ser versionadas via ferramenta “Editor de UPIs”, e posteriormente publicadas com o agregador “Minhas UPIs”.

Instanciar Vcom

Ação de disponibilizar um dado Vcom (presente na “Biblioteca de Vcom”). Vale ressaltar que usuários têm a liberdade de criarem novos moldes de Vcoms e adicionarem na “Biblioteca de Vcom”. Um mesmo tipo de Vcom pode ser instanciado várias vezes. Exemplo: na Biblioteca de Vcom pode haver o Vcom fórum, e este ser instanciado várias vezes, cada um com seu devido *template* e protocolos de interação que o regem.

A Tabela 2 apresenta o resumo de algumas das principais interações descritas.

Tabela 2 - Resumo das principais ações

Conceito	Significado
Publicar UPI	Referenciar uma UPI em um Vcom
Versionar UPIs	Gerar nova versão da UPI
Instanciar Vcom	Disponibilizar um Vcom.

4.3.3 Um modelo conceitual

No trabalho de (Rangel, Beltrame, Cury, & Menezes, 2009) foi exposto um modelo conceitual para os principais conceitos do MOrFEu. Com base na Figura 4.2 pode-se fazer algumas constatações:

- UPIs são referenciadas aos Vcoms instanciados (ato de publicar descrito na seção anterior);

- “Minhas UPIs” está representado no modelo como Biblioteca de Mídia;

- Usuários podem pertencer a grupos, e estes se relacionarem entre si e Vcoms também podem pertencer a grupos;

- UPIs e Vcoms podem ser categorizados;

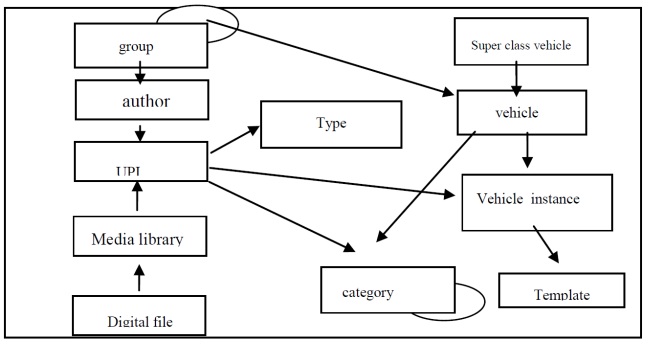


Figura 4.2 - Esboço de um possível modelo conceitual (Rangel, Beltrame, Cury, & Menezes, 2009)

4.3.4 Esquema de funcionamento

A Figura 4.3 apresenta uma visão geral de funcionamento do MOrFEu. Usuários *podem criar Vcom*, de acordo com suas necessidades para vir a atender suas demandas. Ao instanciar tais criações, este estará disponível para acesso a quem o tiver. Assim como usuários *podem usar Vcom* e, eventualmente, gerarem informações através de UPI.

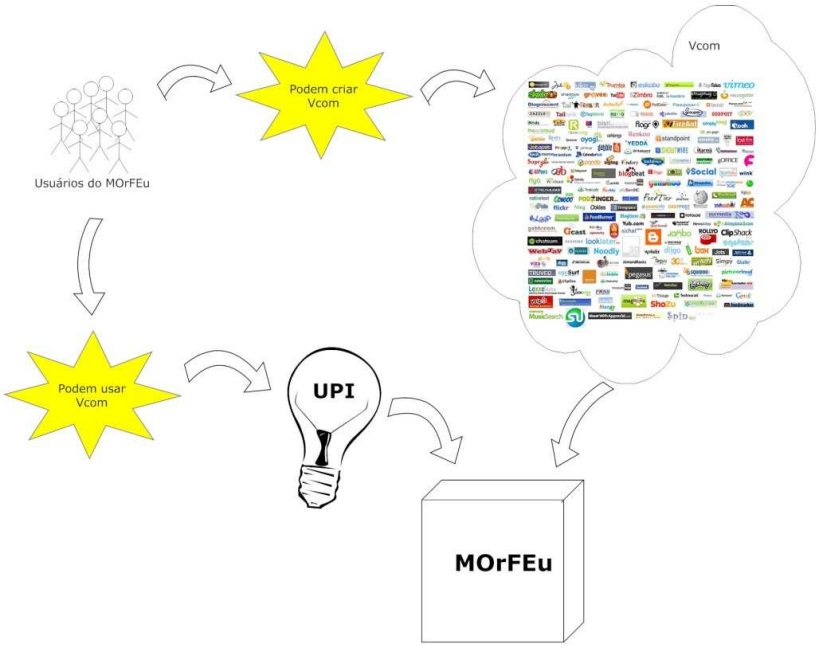


Figura 4.3 - Esquema de funcionamento, em visão geral

Já a Figura 4.4 ilustra outra perspectiva do MOrFEu, onde tem-se a visão de relacionamentos entre os conceitos presentes. UPIs, eventualmente podem ter relacionamento, e publicadas em Vcoms. E Vcoms instanciados a partir da “Biblioteca de Vcom” disponível no MOrFEu.

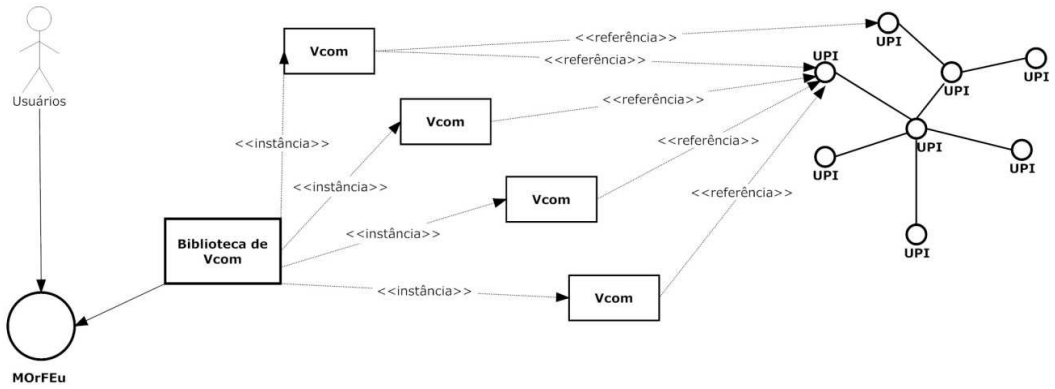


Figura 4.4 - Outra perspectiva esquema de funcionamento, em visão geral

4.4 TORNARNDO VCOMS FLEXÍVEIS

O uso de abordagens de cunho educacionais (como por exemplo Arquiteturas Pedagógicas, citado na seção 3.4.3), apoiadas/amparadas por ambientes virtuais, podem apresentar contribuições com relação à aprendizagem de seus usuários. Mas, para desenvolver Arquiteturas Pedagógicas utilizando tecnologias convencionais pode-se deparar com diversas dificuldades. Dificuldades que decorrem pelo fato dos ambientes convencionais serem concebidos para propósitos específicos e restritos (e de maneira rígida), o que ocasiona na necessidade de contornar tais restrições estruturais (esforço altíssimo) para desenvolver as propostas desejadas. Assim, como Vcom busca abstrair um Ambiente Colaborativo, cabe a ele boa parte da responsabilidade de prover flexibilidade.

Neste trabalho, como já mencionado em seções anteriores, o modelo de referência para abordagem da colaboração escolhido é o Modelo 3C (descrito na seção 3.5). Assim, aqui é descrito alguns dos principais conceitos presentes no Vcom, categorizando-os segundo Modelo 3C de Colaboração. Abaixo, alguns dos principais conceitos comportados por um Vcom:

- Vcoms possuem controle de acesso, podendo ser públicos ou privados;

- Usuários podem pedir acesso ao Vcom ou não (adesão);

- Vcoms tem *template*;

- O Vcom é composto por seções (e estas possuir subseções). As seções que definem protocolos de interação, restrições de acesso à seção (papéis específicos e previamente definidos para acessar), quantidade de usuários, modo de comunicação (síncrono ou assíncrono), podem permitir C.E.R.[8], entre outros;

- UPIs estão dispostas, necessariamente, dentro de seções, no Vcom. Isso é justificável pelo fato de ser as seções que definem o protocolo de interação;

- A seção também pode afetar a sequência de ordenação (ordem) de um conjunto de UPIs publicadas no Vcom;

Como forma de deixar claro a abordagem do Modelo 3C, as próximas tabelas agrupam informações do Vcom segundo cada C do modelo. A Tabela 3 representa os principais elementos de comunicação de um Vcom.

Tabela 3 - Principais elementos de comunicação de um Vcom

Conceito	Significado
Interação	Ação de usuários se comunicarem.
UPI	Elemento básico de autoria.
Comunicação	síncrono ou assíncrono.
Enviar	Evento de enviar uma UPI.
Receber	Evento de receber uma UPI.

Já a Tabela 4 mostra os principais elementos de coordenação de um Vcom.

[8] C.E.R. (Criação de Espaço Reservado) é o requisito funcional para estabelecer um espaço reservado dentro do ambiente virtual, capaz de satisfazer a criação de um espaço compartilhado para acesso dos participantes (Vieira Júnior, Santos, Rafalski, Bada, Silva, & Menezes, 2011).

Tabela 4 - Principais elementos de coordenação de um Vcom

Conceito	Significado
Vcom	Veículo de comunicação.
Seção	Seção do Vcom, responsável por determinar o protocolo de iteração, restrições de acesso, etc. Pode conter subseções.
Acesso	Restrições de acesso do Vcom e da seção.
Protocolo	Conjunto de diretrizes a serem seguidas. Prazos para realizar ações, organização estrutural de seções, C.E.R., são exemplos de modos de definir-se um protocolo.
Conjunto	Mecanismo que também faz parte da definição de um protocolo de interação, permitindo vincular um conjunto estipulado de UPIs em uma seção.

Por fim, a Tabela 5 ilustra os principais elementos de cooperação de um Vcom.

Tabela 5 - Principais elementos de cooperação de um Vcom

Conceito	Significado
Publicação	Publicar UPI em uma seção.
Papel	Cada seção pode ter um papel atrelado a ela, responsável por publicações.

Com base nos parâmetros utilizados para analisar diversos Ambientes Colaborativos em (Fioravanti, Nakagawa, & Barbosa, 2010) e nas teorias descritas em (Menezes, Nevado, Castro Junior, & Santos, 2008) (Carvalho, Nevado, & Menezes, 2005) (Beltrame, Monteiro, Rangel, & Cury, 2008) (Monteiro, Menezes, Nevado, & Fagundes, 2005) (Natalli & Menezes, 2010) (Nevado, Carvalho, & Menezes, Aprendizagem em rede na educação a distância: estudos e recursos para formação de professores, 2007), a Figura 4.5 representa uma representação estrutural (arbórea) de um Vcom e os conceitos comportados por ele.

Na representação exposta na Figura 4.5 foi utilizado uma notação visual diferente para cada termo representado: o paralelogramo representa que trata-se de um atributo[9] de um conceito[10], a elipse representa alguns possíveis valores que atributos podem assumir, o trapézio os dados em si e o retângulo com canto arredondado os conceitos.

[9] Qualidade associada a um elemento já mencionado.

[10] Idéia ou noção, representação geral e abstrata de uma realidade.

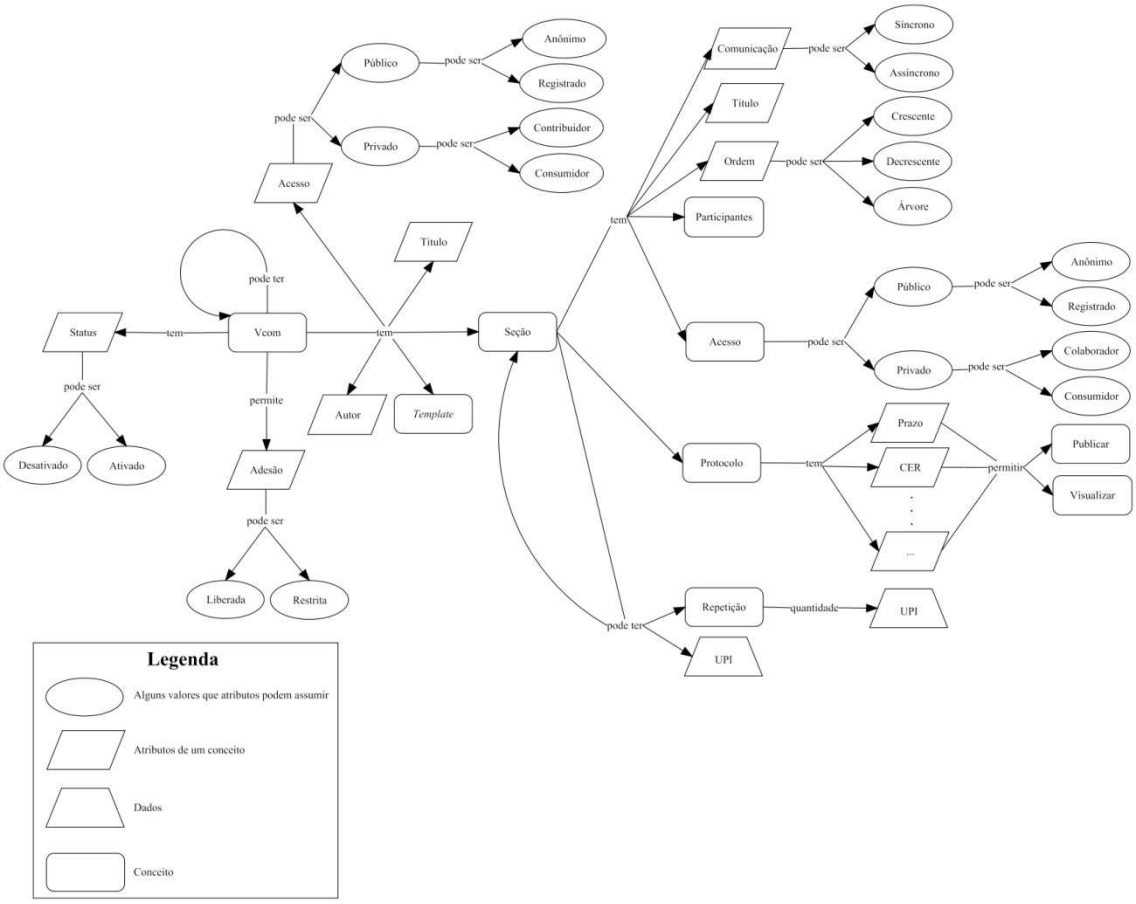


Figura 4.5 - Uma representação da estrutura de um Vcom

Vale ressaltar, na Figura 4.5, a representação do conceito “seção”. É através de seções que os protocolos de interação de um Vcom são representados e descritos. Tipo de comunicação (síncrono ou assíncrono), ordem (crescente, decrescente ou em árvore - que é a disposição das UPIs na seção), restrição de acessos, C.E.R., estruturação de seções (organizando hierarquicamente as seções), dentre outros; são exemplos de configurações que são estipuladas pela seção do Vcom que compõem as diretrizes de um Ambiente Colaborativo. Exemplo: o Vcom de um fórum pode ser abstraído através de 2 seções presentes no Vcom. Uma seção mais externa, buscando representar os tópicos, com ordenação decrescente na data e outra seção, hierarquicamente inferior a outra - uma subseção da seção de tópicos, para definir as respostas de cada tópico, só que esta com ordem em forma de árvore. Uma maneira bastante completa para definição de protocolos de interação de um Vcom é com a utilização de *workflow*, no entanto, está fora do escopo deste trabalho, sendo assim utilizado outros mecanismos para representação do protocolo.

4.5 AMBIENTES COLABORATIVOS MOLDADOS SEGUNDO CONCEITOS DO PROJETO MORFEU

Nesta seção será exposto dois exemplos básicos de ambientes convencionais que podem ser concebidos facilmente segundo os conceitos do MOrFEu. No capítulo de apresentação do protótipo será descrito outros ambientes mais complexos, como forma de realçar os ganhos.

Chat

Chats são aplicações que permitem conversação de modo síncrono. São largamente utilizados na Web. A Figura 4.6 representa uma possível modelagem de um sistema de *chat* segundo a concepção apresentada no intróito da seção anterior. Dentro de uma "seção" (definição do protocolo de interação) contém o "repetição", que representa a possibilidade um conjunto de UPIs (várias mensagens no *chat*).

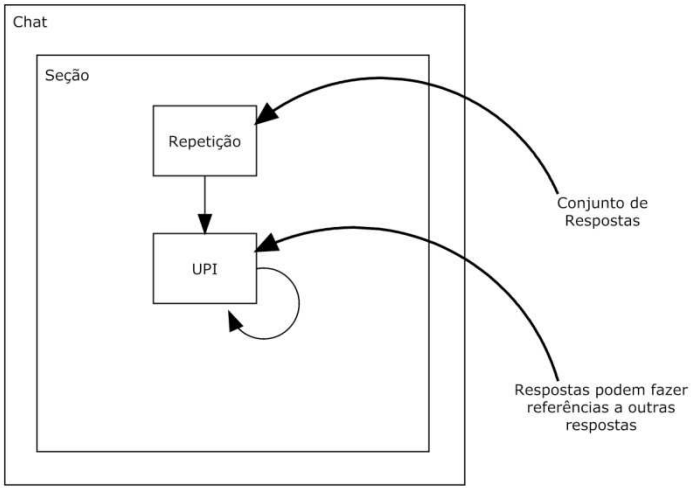


Figura 4.6 - Estruturação de um *chat*

Fórum

Um Fórum (ilustrado na Figura 4.7) também é outro exemplo de ambiente largamente utilizado e que pode ser facilmente modelado segundo a concepção aqui defendida. A natureza arbórea da organização da "seção" pode denotar os tópicos e os *posts* de um

fórum, bem como outras regras que poderiam ser estabelecidas como protocolo de interação. O “repetição” representa as várias possibilidades de *posts* em um tópico.

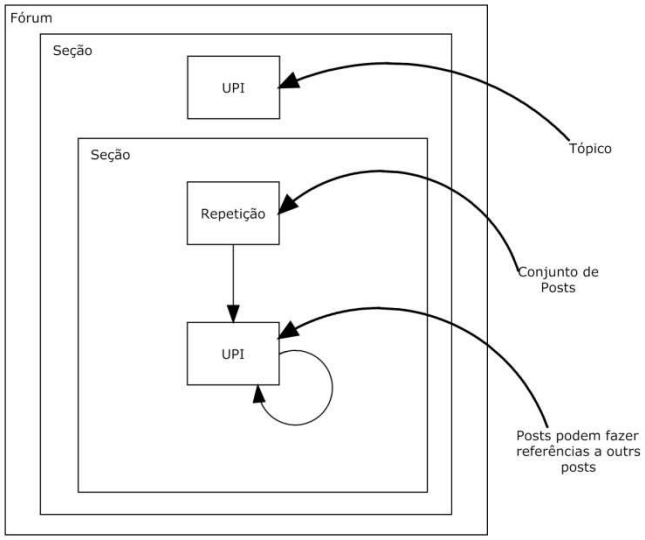


Figura 4.7 - Estruturação de um fórum

4.6 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo formalizou os pilares a qual esse trabalho usou para elaboração da proposta. Como a proposta final dessa dissertação é a elaboração de um *framework* para construção de Ambientes Colaborativos, seria imprescindível deixar de falar do projeto MOrFEu e dos conceitos que o permeiam.

O capítulo iniciou descrevendo a demanda de ambientes flexíveis, para servir como justificativa para o surgimento do projeto MOrFEu, e depois foi introduzido conceitos principais do projeto e os modelos conceituais.

Outra parte importante do capítulo é a descrição dos conceitos de Vcom e relacionados a ele. Foi utilizado o Modelo 3C de Colaboração como auxílio (parâmetro) para fazer uma análise mais esclarecedora.

Por fim, a última seção molda alguns Ambientes Colaborativos básicos utilizando conceitos do projeto MOrFEu, contribuindo com o início da evidenciação de viabilidade dessa concepção.

CAPÍTULO 5 O FRAMEWORK PROPOSTO

5.1 INTRODUÇÃO

Neste capítulo é apresentado a proposta deste trabalho. A partir deste ponto, será utilizado o acrônimo FrameColab (**Frame**work de Ambientes de **Colab**orativos) para designar a proposta em questão.

O capítulo está dividido basicamente em duas grandes seções: a seção 5.2 abordando alguns conceitos presentes no FrameColab que ajudam a compreendê-lo em linhas gerais e, a seção 5.3 que apresenta o detalhamento do FrameColab, segundo as melhores práticas na literatura da Engenharia de Software.

5.2 CONCEITOS GERAIS

Aqui é apresentada a proposta em linhas gerais. O objetivo de tal estratégia é facilitar a compreensão, uma vez que a formalização em si da proposta (seção 5.3) apresenta uma visão mais detalhada e complexa do FrameColab.

5.2.1 Visão geral

A Figura 5.1 apresenta uma visão geral de funcionamento do FrameColab proposto neste trabalho. O FrameColab possui dois propósitos básicos: fornecer mecanismos para criação (instanciação) de Vcoms e prover o ambiente necessário para usuários consumirem Vcoms instanciados. Cada propósito será descrito em seções posteriores.

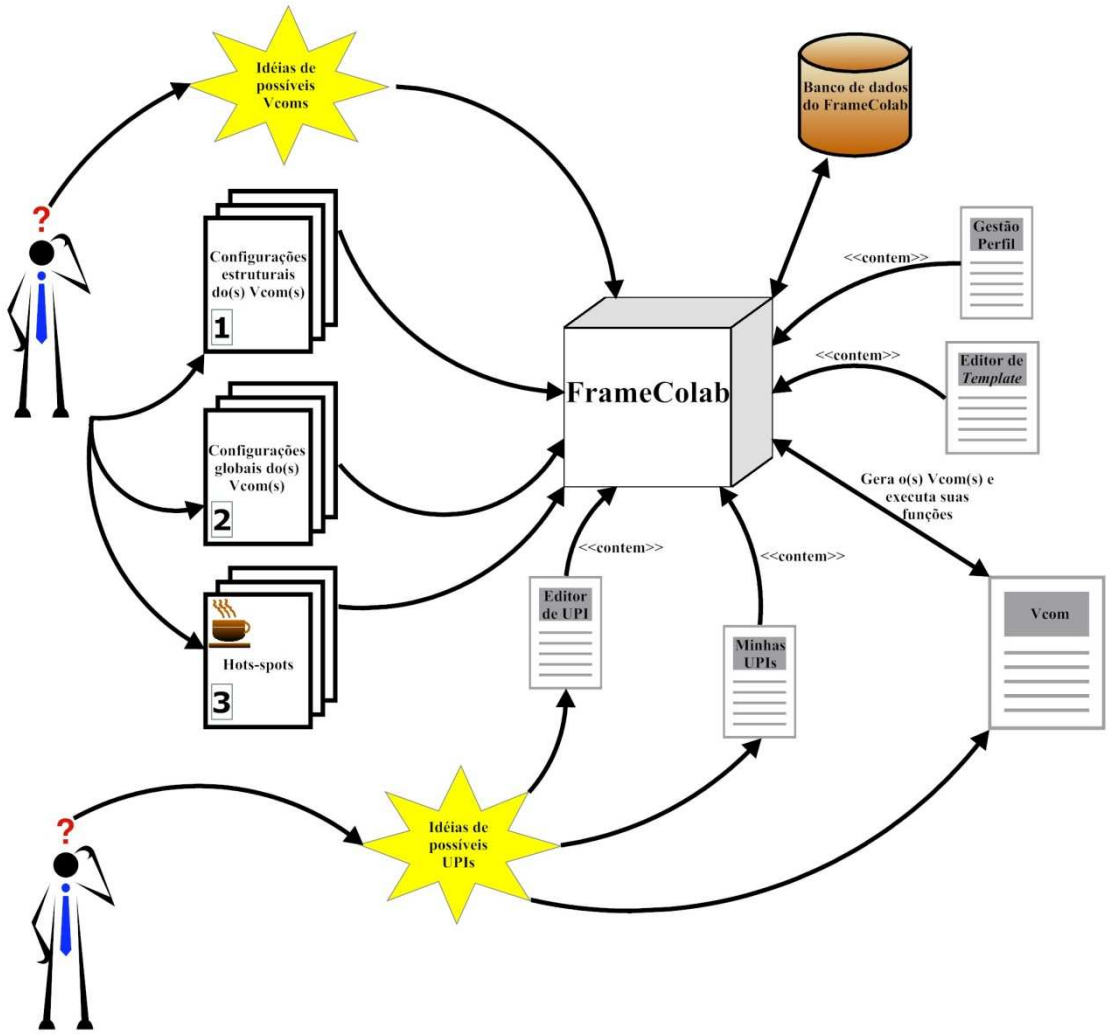


Figura 5.1- Visão geral de funcionamento do FrameColab

Ainda referindo-se à Figura 5.1, para criação de Ambientes Colaborativos com o FrameColab, usuários descrevem estruturalmente o Vcom a ser criado segundo uma linguagem de descrição (apresentada na próxima seção e ilustrada por 1 na Figura 5.1), descrevem as configurações globais (descrito na seção 5.2.3 e mostrado em 2 na Figura 5.1) e, por último, descrevem os *hot-spots* (implementações de pontos flexíveis do Vcom a ser projetado, ilustrado em 3 na Figura 5.1). Através do "Editor de *Template*" o *template* do Vcom é customizado, sem necessidade de programação. Já com a "Gestão Perfil" os perfis do Vcom são criados de acordo com a necessidade. Os Vcoms são gerados em tempo de execução, na medida que são demandados (para permitir ao usuário atualizar qualquer ponto em qualquer momento). Por fim, interagindo com Vcoms instanciados ou simplesmente criando UPIs sem publicá-las (via "Editor de

UPI"), usuários esboçam suas idéias/pensamentos (UPIs), que são agrupadas com o "Minhas UPIs".

Vale ressaltar que o foco deste trabalho não é o manejo de *template*, assim, o projeto de um "Editor de *Template*" está fora do escopo da proposta. No entanto, é levado em consideração tal necessidade e previsto no FrameColab mecanismo para acoplar um "Editor de *Template*" futuramente. O mesmo é aplicado no "Editor de UPI" e "Minhas UPIs", que são provindos pelo FrameColab com o caráter ilustrativo, não sendo foco do trabalho fazer a proposta de cada um.

5.2.2 Em busca de uma linguagem para descrição de Vcoms

A estrutura hierárquica (arbórea), a modularidade e a necessidade de extensão nas representações de Vcoms e suas configurações, conforme visto na seção 4.4, podem ser descritos utilizando representações em XML - Linguagem de Marcação Extensível (*eXtensible Markup Language*). Aqui será descrito uma linguagem, baseada em XML, para descrição de Vcoms, como forma de prover flexibilidade aos usuários na concepção de Vcoms. O FrameColab processa a XML contendo descrições de Vcoms e os gera.

XML é uma linguagem de marcação extensível baseada na linguagem de marcação padrão generalizada - SGML (*Standard Generalized Markup Language*) (International Organization for Standard ISO, 1986), projetada para descrever dados. A XML também oferece uma grande de quantidade de ferramentas de processamento disponíveis para esta linguagem. As duas principais características da XML são a ênfase em uma marcação descritiva ao invés da procedural e ao conceito de tipo de documento. Utilizando XML é possível definir um conjunto próprio de marcas. Além disso, pela utilização de esquemas de XML, é possível definir estruturas (tipos) de documentos.

A Tabela 6 mostra as marcações (*tag*) disponíveis na linguagem proposta:

Tabela 6 - Descrição das marcações da linguagem de descrição de Vcoms proposta

Marcação	Descrição
<repeticao/>	Comportamento semelhante a palavra reservada "*while*" de linguagens de programação em geral. Representa

	repetições das entidades que estão hierarquicamente inferiores a ela. Pode também ser utilizado para representar o conceito “repetição”, exposto na Figura 4.5.
<secao/>	Representa uma seção de um Vcom. Esta marcação materializa o conceito “seção”, presente no Vcom e descrito no capítulo anterior, buscando representar o protocolo de interação do Ambiente Colaborativo projetado.
<upi/>	Define a existência de uma UPI. Deve sempre ser utilizada dentro da marcação de seção, pois uma UPI precisa de definições sobre seu comportamento (protocolo de interação).

Obviamente, cada marcação descrita acima possui um conjunto de atributos, que são descritos conforme a Tabela 7, abaixo:

Tabela 7 - Descrição dos atributos de cada marcação da linguagem de descrição de Vcoms

Marcação	Atributo	Descrição do atributo	Valor
<repeticao/>	quantidade	Atributo para descrever o número de vezes que irá se repetir a ação.	Numérico
<secao/>	titulo	Título da seção.	Textual
<secao/>	ordem	Descreve a forma de recuperação das UPIs hierarquicamente inferiores a ela. Disponíveis: data crescente/decrescente, título crescente/ decrescente e árvore.	Textual
<secao/>	prazo_inicio	Data de início de permissão de postagem na seção.	Data
<secao/>	prazo_fim	Data fim de permissão de postagem na seção.	Data
<secao/>	codigo	Código identificador da funcionalidade, que será previamente cadastrado pelo administrador no momento de criação dos perfis para concessão de acessos.	Numérico
<secao/>	cer	Indicativo se na seção será C.E.R. (cada usuário posta e só visualiza suas UPIs).	sim/nao
<upi/>	titulo	Indicativo se a UPI contém título. Há UPIs que não contém título, por exemplo um mensagem em um *chat*.	sim/nao

<upi/>	editor	Indicativo se o preenchimento da UPI deverá ser feito com um editor WYSIWYG[11].	sim/nao
<upi/>	textarea	Indicativo se o corpo da UPI será um campo extenso.	sim/nao
<upi/>	autorelacionamento	Indicativo se a UPI permite se auto-relacionar com outra UPI.	sim/nao

E a Figura 5.2 mostra um exemplo de um Vcom convencional descrito estruturalmente na linguagem de descrição proposta.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <blog>
3      <repeticao quantidade="2">
4          <secao titulo="Posts" codigo="1" cer="nao" prazo_inicio="21/03/2011" ordem="data_crescente">
5              <upi titulo="sim" editor="sim" textarea="sim" autorelacionamento="nao" />
6              <secao titulo="Comentarios" codigo="2" cer="nao" prazo_inicio="28/03/2011" ordem="arvore">
7                  <repeticao quantidade="3">
8                      <upi titulo="sim" editor="sim" textarea="sim" autorelacionamento="sim" />
9                  </repeticao>
10             </secao>
11         </secao>
12     </repeticao>
13 </blog>
```

Figura 5.2 - Exemplo de descrição de um Vcom

A Figura 5.2 representa uma configuração estrutural de um *blog*. Neste caso, trata-se de um *blog* que permite somente 2 *posts* (conforme o atributo “quantidade” da linha 3), com 3 comentários (conforme o atributo “quantidade” da linha 7) para cada *post*. Vale enfatizar que, neste caso, cada *post* terá um prazo de postagem (definido nos atributos da marcação “secao” na linha 4) e que cada comentário terá um prazo de postagem (também definido nos atributos da marcação “secao”, só que na linha 6). Por fim, o atributo “cer” da linha 6 define que tal seção permitirá que todas UPIs desta seção sejam visíveis aos usuários que tiverem permissão à seção, que é o que ocorre nos *blogs* em geral (todos podem ver respostas a um *post*).

É importante ressaltar a flexibilidade que é dada ao usuário permitindo-o que estruture seus Vcoms segundo uma linguagem, de alto nível. Com mudanças simples, o usuário altera estruturalmente o Vcom que está sendo projetado. Alterando-se simplesmente o atributo “ordem” de uma marcação “secao”, pode-se, por exemplo, alterar o modo como

[11] WYSIWYG é o acrônimo da expressão em inglês “**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et” (O que você vê é o que você obtém) (OQVVEOQVO). Significa a capacidade de um programa de computador de permitir que um documento, enquanto manipulado na tela, tenha a mesma aparência de sua utilização, usualmente sendo considerada final a forma impressa.

as UPIs de dentro daquela seção serão recuperadas (sem necessidade de esforços em implementações).

Em “Apêndice I: Representação do XMLSchema da linguagem de descrição segundo o padrão da W3C” e em “Apêndice III: XMLSchema da linguagem de descrição utilizada e das configurações do Vcom” são mostrados o diagrama segundo o padrão da W3C e a formalização da linguagem de descrição de acordo com o padrão de representação XMLSchema[12], respectivamente.

5.2.3 Configurações globais de um Vcom

Um Ambiente Colaborativo pode ter também algumas características globais, conforme exposto na seção 4.4 (atributos do conceito Vcom na Figura 4.5 - Uma representação da estrutura de um Vcom), do capítulo anterior. A proposta deste trabalho sugere as seguintes configurações: título, CSS, formatação de mensagens, formatação de datas, indicativo de privado, indicativo de permissão de acesso anônimo, indicativo de permissão de adesão, indicação de ativação e os *hot-spots*. Estes últimos serão detalhados mais a frente.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <vcomconfig>
3      <titulo>Meu vcom blog</titulo>
4      <css>blog.css</css>
5      <mensagemAutoria>[%s] %s: </mensagemAutoria>
6      <formatoData>EEE, dd/MM/yyyy, HH:mm:ss</formatoData>
7      <privado>nao</privado>
8      <permiteanonimo>nao</permiteanonimo>
9      <permiteadesao>sim</permiteadesao>
10     <ativado>sim</ativado>
11     <hotspotpublicacao>exemploshotspots.HotSpotPublicacaoImpl</hotspotpublicacao>
12 </vcomconfig>
```

Figura 5.3 - Exemplo de configurações globais de um Vcom

A Figura 5.3 ilustra uma possível configuração para um *blog*. Na linha 3 é definido o título, na linha 4 o CSS do Vcom, na linha 5 a formatação da mensagem de autoria, na linha 6 a formatação das datas exibidas, na linha 7 o indicativo de privado, na linha 8 o indicativo de permissão de anônimos acessarem, na linha 9 o indicativo de permissão de

[12] XML Schema é uma linguagem baseada no formato XML para definição de regras de validação (“esquemas”) em documentos no formato XML. Um arquivo contendo as definições na linguagem XML Schema é chamado de XSD (*XML Schema Definition*), este descreve a estrutura de um documento XML.

adesão, na linha 10 o indicativo de ativado e na linha 11 o apontamento de qual classe irá implementar um determinado *hot-spot*.

Em “Apêndice II: Representação do XMLSchema do XML de configuração do Vcom segundo o padrão da W3C” e em “Apêndice III: XMLSchema da linguagem de descrição utilizada e das configurações do Vcom” são apresentados o diagrama segundo o padrão da W3C e a formalização das configurações de acordo com o padrão de representação XMLSchema, respectivamente.

5.3 DETALHAMENTO DA PROPOSTA

Nesta seção é apresentado em detalhes a proposto deste trabalho. Será descrito a metodologia de desenvolvimento utilizada bem como o modo de sua execução e suas limitações. Cada etapa será detalhada através de modelos, diagramas e figuras, como forma de esclarecimento do trabalho descrito.

5.3.1 Metodologia de desenvolvimento

O desenvolvimento da proposta (FrameColab) passou pelas fases tradicionais da Engenharia de Software: Levantamento de Requisitos, Análise, Projeto, Implementação, Testes e Implantação. As etapas de Levantamento de Requisitos, Análise e Projeto serão abordadas neste capítulo, uma vez que retratam o detalhamento da proposta. Já as etapas Implementação, Testes e Implantação serão descritas no próximo capítulo, onde será apresentado o protótipo desenvolvido baseado na proposta deste trabalho.

Na Figura 5.4 é ilustrado as fases no desenvolvimento, sendo um exemplo de modelo de desenvolvimento iterativo e incremental. Iterativo pois corresponde à ideia de “melhorar (ou refinar) pouco-a-pouco” o sistema (iterações). E incremental pois corresponde à ideia de “aumentar (alargar) pouco-a-pouco” o âmbito do sistema, disponibilizando novas versões.

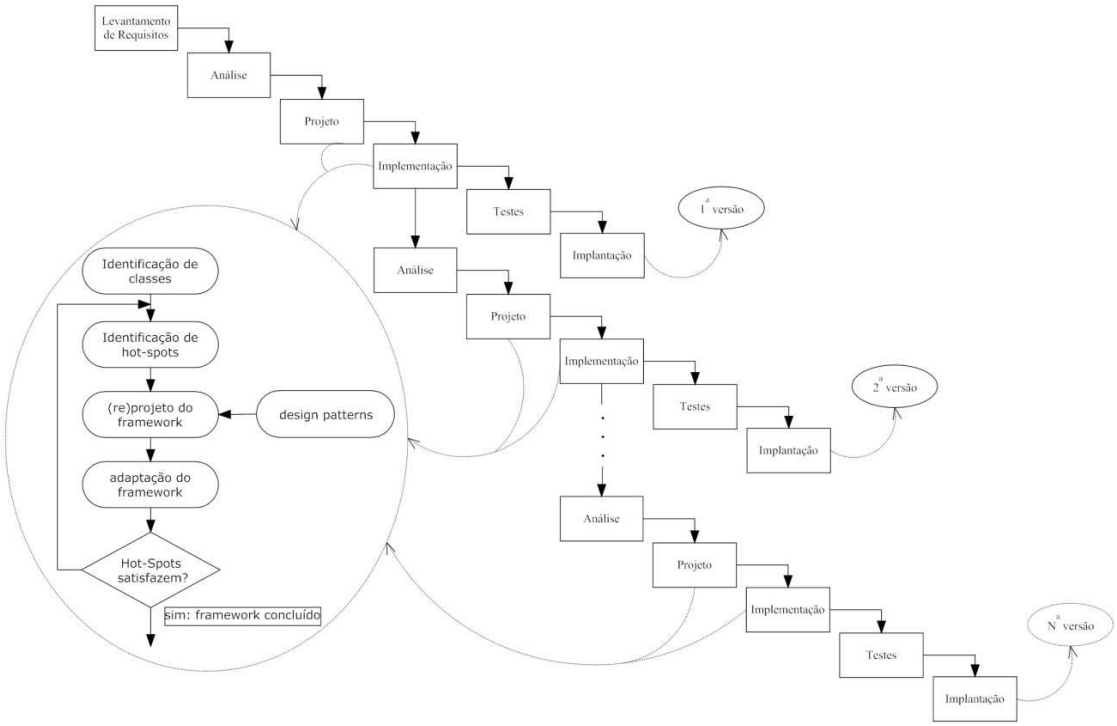


Figura 5.4 - Visão geral da metodologia de desenvolvimento

A Figura 5.4 também retrata que nas etapas de Projeto/Implementação a metodologia Projeto Dirigido por *Hot-Spots* (descrita na seção 2.5.4.3) é utilizada, como um meio de ajudar no refinamento do *framework* desenvolvido. E também são nessas etapas que há sinalizações dos requisitos a serem cobertos na próxima versão.

5.3.1.1 Levantamento de requisitos

O objetivo do Levantamento de Requisitos é transmitir uma idéia global e genérica do projeto a ser desenvolvido segundo a visão dos usuários. Ela é elaborada justamente para compreender a real necessidade dos usuários e suas expectativas, fazendo com que estes desejos sejam atendidos da melhor maneira possível.

A proposta deste trabalho é de um *framework*, denominado FrameColab, que tenha os conceitos do projeto MOrFEu (descritos no Capítulo 4), de modo que facilite a superação dos desafios na concepção de novos Ambientes Colaborativos, conforme citado no Capítulo 3.

Obviamente o contexto do projeto MOrFEu é muito amplo, contendo várias linhas de pesquisa. Assim, foi decidido os seguintes requisitos funcionais a serem atendidos pela proposta deste trabalho:

- Facilitar o projeto de Ambientes Colaborativos fazendo uso das premissas do projeto MOrFEu;

- Prover um ambiente para execução dos Vcoms instanciados;

- Definir uma arquitetura básica (com definição das camadas e padrões utilizados) para os Vcoms originados do *framework*;

- Dar flexibilidade ao usuário para definição de papéis a serem utilizados nos Vcoms projetados;

- Vcoms gerados a partir do FrameColab contenham representações dos conceitos de *template* e seção (descrito na seção 4.4);

- Tornar características do Vcom configuráveis (ver seção 5.2.3);

- Permitir a descrição estrutural de Vcoms através de uma linguagem de marcação (relatado na seção 5.2.2);

- Fornecer pontos, estrategicamente escolhidos, onde usuários podem customizar algumas rotinas nos Vcoms (conceito de *hot-spot* ilustrado na seção 2.4.4);

- Fornecer mecanismos de básico manipulação (busca, criação, edição e publicação) de UPI (figura do “Editor de UPI” e do “Minhas UPIs”, descritos na seção 4.3.1) e também prever o acoplamento de um “Editor de *Template*”;

A seção 3.6 teve forte influência na escolha dos requisitos, uma vez que enfatiza as principais deficiências dos ambientes convencionais. Na Figura 5.5, é apresentado uma visão conceitual das principais atividades em cada nível conceitual do FrameColab. São descritos quatro papéis e suas respectivas ações.

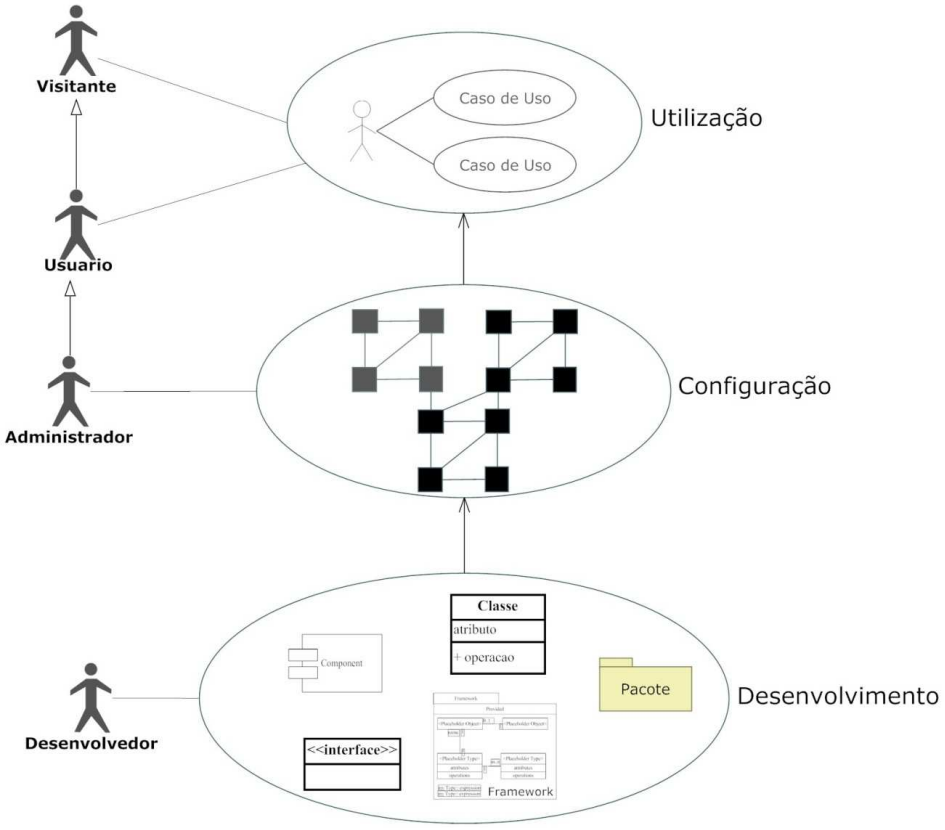


Figura 5.5 - Visão conceitual (atores x papéis). Adaptado de (Natalli & Menezes, 2010)

Um “Desenvolvedor” é o perito/especialista em programação, com o papel de desenvolver o ambiente virtual a partir do *framework* proposto. Utilizando o FrameColab, tal ator realiza a implementação dos *hot-spots* que o Vcom irá prover, além também de instanciar o Vcom, trabalhando sob orientações do “Administrador”. Um “Administrador” é o responsável pela parametrização do Vcom já instanciado pelo desenvolvedor. Tratando-se do domínio de softwares voltados para a educação, é a figura do professor, que configura do modo que desejar. Ele também delega a implementação de customizações em *hot-spots* aos desenvolvedores. O “Usuario” é o usuário final, cadastrado no ambiente, que interage e consome os recursos que exigem cadastro. O “Visitante” é o usuário que faz acesso sem registro, ou seja, mantêm-se no anonimato, mas mesmo assim pode acessar alguns recursos do Vcom instanciado. Tratando-se do domínio de softwares voltados para a educação, estes dois últimos atores são os aprendizes (ou alunos).

5.3.1.2 Análise

Na fase de análise o problema descrito na Especificação de Requisitos é estudado com mais profundidade, tentando elaborar um modelo que represente o domínio do sistema. Em especial, na Análise Orientada a Objetos são identificados os objetos, seus relacionamentos e comportamentos necessários para a construção dos mesmos.

Criação de Vcom

As etapas para criar um Vcom no FrameColab podem ser descritas, conforme a Figura 5.6. Nesta ilustração é mostrado um diagrama de atividades, buscando mostrar os passos em um nível “macro”. Inicialmente o ator acessa o ambiente do FrameColab, autentica-se com permissões administrativas e cria o Vcom, definindo suas configurações estruturais (5.2.2), suas configurações globais (5.2.3) e a implementação do(s) *hot-spot(s)* utilizados. Depois, os perfis de acesso devem ser criados, que fazem a junção entre as funcionalidades, também cadastradas, do Vcom e suas devidas permissões.

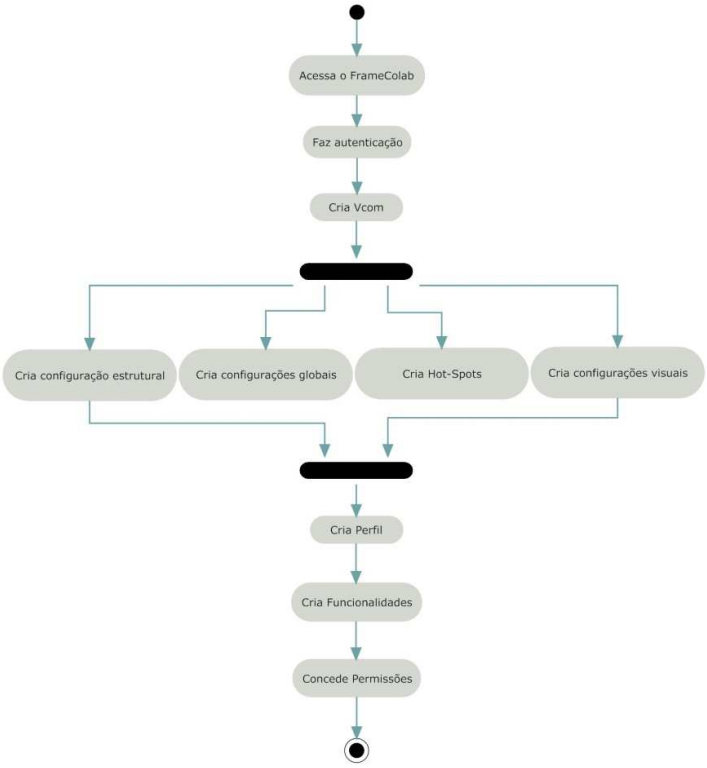


Figura 5.6 - Criação de Vcom

Modelo conceitual do domínio

O modelo conceitual tem como objetivo apresentar as entidades presentes no sistema e como estas se relacionam. A Figura 5.7 representa um modelo conceitual das entidades presentes no FrameColab. Algumas entidades merecem atenção:

- Usuario: entidade para caracterizar os atores presentes no sistema;
- Vcom: entidade para representar o ambiente que está presente no FrameColab;
- Upi, VersaoUpi e Publicacao: entidades que retratam uma UPI , seu poder de sofrer versionamento e o ato de publicá-la em um Vcom.

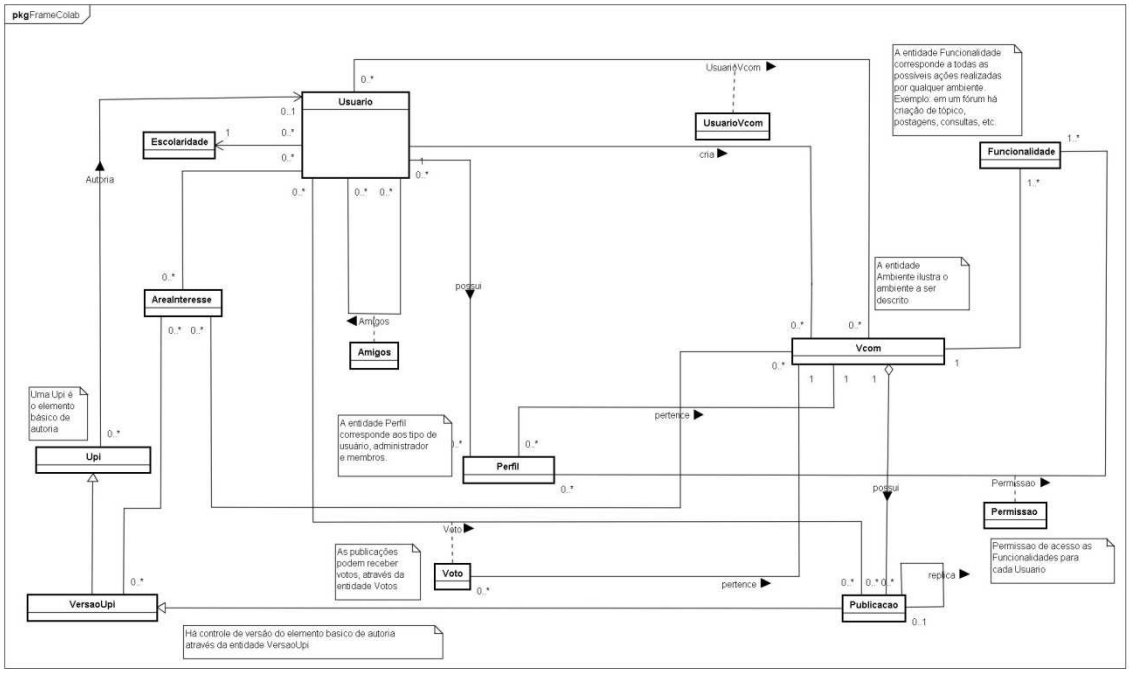


Figura 5.7 - Modelo conceitual do domínio

5.3.1.3 Projeto

Durante a fase de análise, o sistema é modelado sem levar em conta os aspectos tecnológicos a serem utilizados para materializá-lo. A fase de projeto tem o objetivo de definir todos os aspectos tecnológicos do sistema, tais como linguagem de programação utilizada na implementação, arquitetura do software, características de interface com o

usuário e formas de acesso, persistência de dados entre outros. É a etapa mais importante na elaboração desta proposta, uma vez que define toda a estruturação interna do FrameColab de modo a comportar o modelo conceitual apresentado anteriormente.

Arquitetura Geral Interna

A Figura 5.8 ilustra a arquitetura interna do FrameColab. As denominações *Model*, *View* e *Controller* vêm do padrão MVC (ver seção 2.3.3), que é um padrão largamente utilizado para desenvolvimento de aplicações na Web. A *View* é gerada através do *framework* de interface, que também provê suporte ao design de interfaces baseadas em *Web templates*[13] (um possível "Editor de *Template*" para o FrameColab ficaria acoplado em tais *templates*). A sigla HVM também é outro padrão adotado no projeto, comentado em capítulos anteriores (seção 2.3.3), que é representado pela forma de se acoplar Vcoms no FrameColab ("plugando-se" arquivos XMLs). Já o *Façade* é utilizado como forma de prover uma interface única e padronizada para comunicação com a camada de apresentação. No padrão DTO estão representadas as classes de negócio (que fazem uso de classes de domínio - *Model*) da aplicação o no DAO as classes que fazem uso da API[14] do *framework* de persistência e provêem acesso aos dados de maneira transparente (sem necessidade de elaboração de SQL).

O objetivo na incorporação de um *framework* de interface e um *framework* de persistência pode ser justificado por representarem uma solução pré-moldada para os dados domínios (interface e persistência). Um outro *framework* que poderia ser acoplado a arquitetura do FrameColab é um *framework* de *workflow*, como forma de prover mecanismos de representação facilitada de tarefas. No entanto, tal trabalho está fora do escopo desta proposta.

---

[13] São instrumentos utilizados para separar a apresentação do conteúdo em web *design*, e para a produção massiva de documentos web.

[14] API, de ***A**pplication **P**rogramming **I**nterface* (ou Interface de Programação de Aplicativos) é um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por aplicativos que não pretendem envolver-se em detalhes da implementação do software, mas apenas usar seus serviços (foldoc.org, 1995).

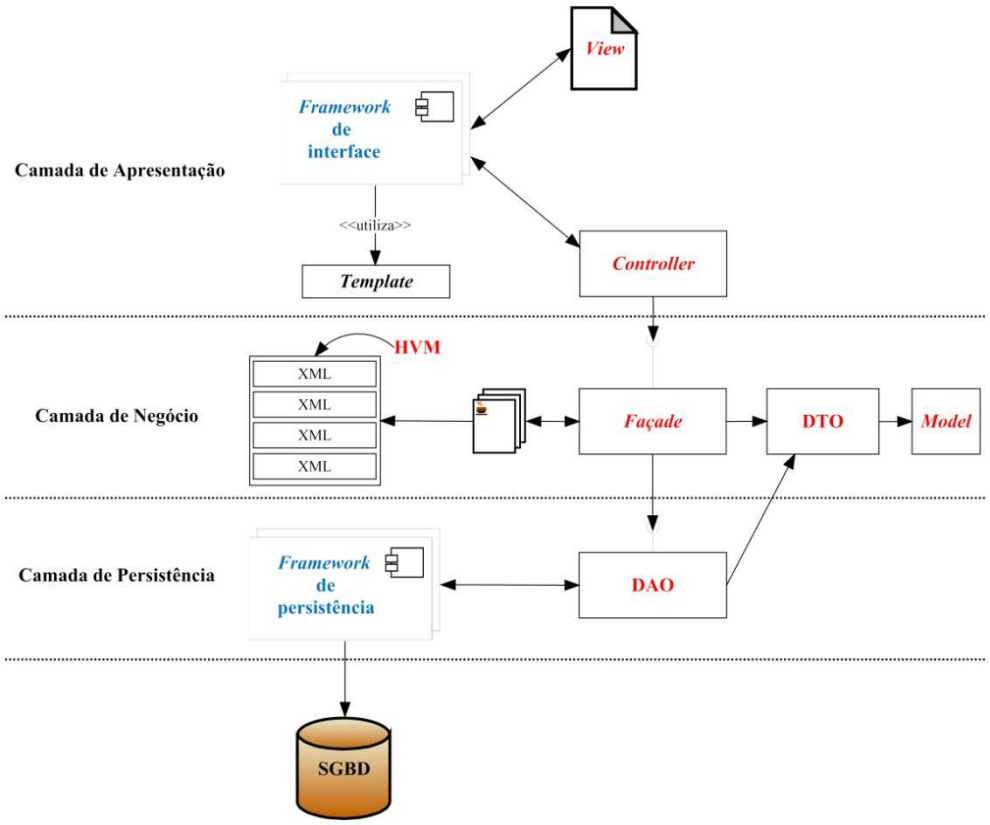


Figura 5.8 - Arquitetura Geral Interna do FrameColab

Ainda neste capítulo será descrito os principais recursos providos pela camada de negócio e de persistência. A seguir, é definido as escolhas tecnológicas, com base nas características arquiteturais definidas.

Tecnologias Envolvidas

A Tabela 8 sumariza as principais tecnologias definidas para o FrameColab. Escolheu-se a linguagem Java pois sua característica multi-plataforma introduz um diferencial no projeto, além de possuir inúmeros projetos em Java podendo ser acoplados a solução.

ZKoss[15] é um *framework* de interface, escrito em Java, que permite a criação de interfaces gráficas ricas para Web, fazendo uso de AJAX[16] sem a necessidade de

[15] http://www.zkoss.org/

[16] AJAX (acrônimo de ***A**synchronous **J**avascript **a**nd* **X**ML, em português "Javascript e XML Assíncronos") é o uso metodológico de tecnologias como Javascript e XML, providas por navegadores,

conhecimento em JavaScript (gera automaticamente componentes visuais com JavaScript e compatível com um grande leque de navegadores). Ele possui uma linguagem baseada em XML para descrever os componentes visuais de forma facilitada e tal recurso foi utilizado buscando representar *templates* do projeto MOrFEu (acoplados com CSS dos Vcoms).

Hibernate[17] é um *framework* para o mapeamento objeto-relacional, em Java, facilitando o mapeamento dos atributos entre uma base tradicional de dados relacionais e o modelo objeto de uma aplicação; tendo como objetivo a diminuição da complexidade de acesso (e atualização) da dados em bases relacionais. O Hibernate gera as chamadas SQL e libera o desenvolvedor do trabalho manual da conversão dos dados resultante, mantendo o programa portável para quaisquer bancos de dados SQL (tal característica permite transpor o projeto para qualquer banco de dados, que é uma limitação de ambientes convencionais). Também possui mecanismos relacionados ao gerenciamento de transações e tecnologias de acesso à base de dados, através de uma API bem definida (que são utilizadas nas classes DAO).

Por fim, o banco de dados e do servidor de aplicação podem ser alterados em qualquer momento, uma vez que o projeto está totalmente independente destes, sendo assim uma apenas escolha para exemplificar o uso.

Tabela 8 - Tecnologias utilizadas

Banco de dados	MySQL
Servidor de aplicação	Apache TomCat
Linguagem/plataforma	Java J2EE
Framework de interface	Zkoss
Framework de persistência	Hibernate

Principais recursos oferecidos pela camada de negócio e camada de persistência

A camada de negócio, retratada na Figura 5.8, busca transmitir as principais regras de negócio contidas no domínio. É com acesso a essa camada que os Vcoms terão suas primitivas básicas, com interfaces bem definidas e pronto para serem utilizadas. A

para tornar páginas web mais interativas com o usuário, utilizando-se de solicitações assíncronas de informações.

[17] http://www.hibernate.org/

segui, algumas das principais regras. (i) ações relacionadas a UPI, tais como: publicar UPI, editar UPI, versionar UPI, entre outros; (ii) ações relacionadas a Vcom: criar Vcom; (iii) ações relacionadas a usuários: criar usuário, testar senha, recuperar senha, recuperar perfil, recuperar amigos, entre outros.

Entre vários outros recursos oferecidos. No entanto, a camada de negócio depende da camada de persistência para seu funcionamento. Pois é a camada de persistência que abstrai todo o acesso à dados. A seguir, alguns exemplos de competências da camada de persistência. (i) inclusão, recuperação e alteração de objetos do domínio; (ii) mecanismos para controle de transação; (iii) abstração do banco de dados, entre vários outros.

Definição dos Pacotes do Domínio

Para uma maior organização, as classes são separadas em pacotes (Figura 5.9). Os diagramas de pacotes dão uma visão mais geral da divisão das classes e colaboração entre pacotes. Assim, na Figura 5.9, é retratado as definições dos pacotes para organizar as classes de domínio do FrameColab:

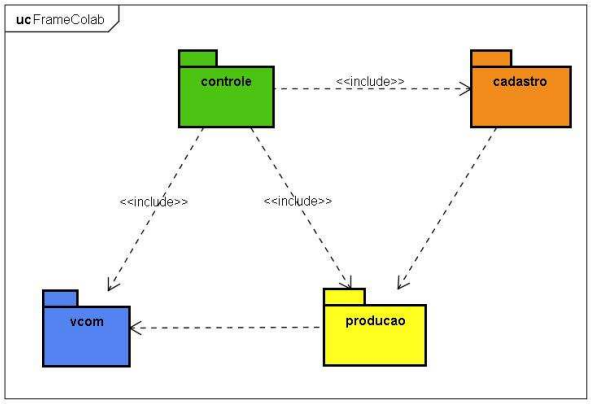


Figura 5.9 - Pacotes no FrameColab

- cadastro: envolve todas entidades relacionadas com o cadastro de informações de apoio, abrangendo amigos, usuários, áreas de interesse e escolaridade;

- controle: envolve todas entidades relacionadas com o controle de acesso, abrangendo perfil e permissão;

- producao: envolve todas entidades relacionadas com o produção de informação, abrangendo upi, versão de upi e voto[18];

- vcom: envolve todas entidades relacionadas com Vcom, abrangendo funcionalidade e vcom.

Diagrama de Classes do Domínio

Ao abstrair os conceitos do mundo real em classes de objetos, seus atributos e relacionamentos no mundo computacional, são formados diagramas de classes, que mostram as propriedades e as associações entre as classes do sistema. Tais propriedades e associações são detalhadas no dicionário de dados (ver no Apêndice IV: Dicionário de dados do diagrama de classes). A seguir, na Figura 5.10, é apresentado o diagrama de classes do domínio do FrameColab que é uma transposição do modelo conceitual apresentado na Figura 5.7.

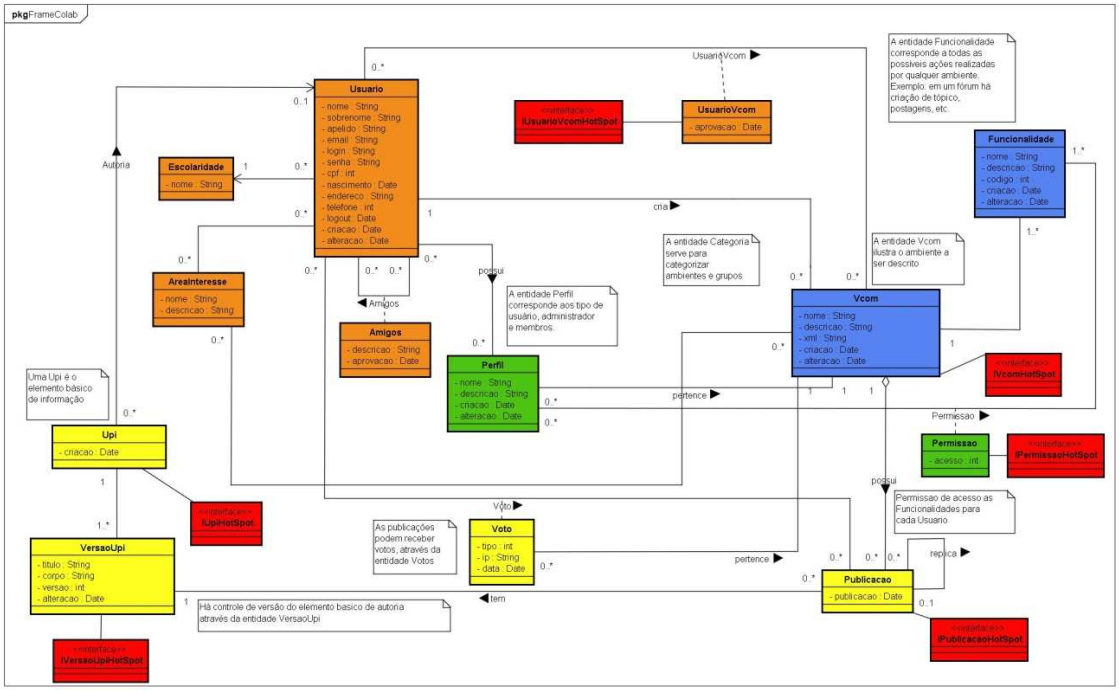


Figura 5.10 - Diagrama de classes de domínio do FrameColab. Em vermelho os *hot-spots*, em azul as classes pertencentes ao pacote vcom, em amarelo as classes pertencentes ao pacote producao, em

[18] UPIs em algumas situações podem receber votos, buscando representar recursos presentes em alguns ambientes, como por exemplo em enquetes, agregadores de notícias sociais, etc.

laranja as classes pertencentes ao pacote cadastro e em verde as classes pertencentes ao pacote controle

O objetivo do diagrama de classes acima é retratar todos os conceitos presentes no FrameColab, bem como seus relacionamentos e atributos.

Hot-Spots do FrameColab

Conforme relatado na seção 2.4.4, *hot-spot* é um importante componente de um *framework*, uma vez que permite o acoplamento de implementações em pontos estratégicos. Nesta parte do trabalho será descrito os *hot-spots* comportados pelo FrameColab. Na Figura 5.10, em vermelho, podem ser vistos os *hot-spots*. Eles estão presentes na classe de negócio do FrameColab buscando dar flexibilidade durante algumas operações relacionadas ao negócio:

- IUpiHotSpot: *hot-spot* ligado a UPI;
- IVersaoUpiHotSpot: *hot-spot* ligado ao versionamento de UPI;
- IPublicacaoHotSpot: *hot-spot* ligado a publicação de UPI;
- IPermissaoHotSpot: *hot-spot* ligado a concessão de acesso (permissão) de usuários;
- IVcomHotSpot: *hot-spot* ligado a Vcom;
- IUsuarioVcomHotSpot: *hot-spot* ligado ao acesso de usuários a Vcoms;

Todos *hot-spots* descritos acima estão na camada de negocio, presentes em rotinas de salvamento e de acesso a dados de suas classes de domínio. Alguns exemplos de possibilidades com implementações de *hot-spots*:

- Disparar email ao publicar uma UPI em resposta a uma UPI;
- Sinalizar que uma UPI foi criada ou versionada;

- Notificar usuário que teve acesso concedido a determinado Vcom;

- Notificar usuários de um determinado perfil que a permissão de acesso de uma funcionalidade foi alterada;

- Gerar informações de estatísticas de acesso a determinada informação;

- Informar usuários em caso de alteração no Vcom;

Entre várias outras possibilidades. A intenção do FrameColab prover *hot-spots* é permitir o acoplamento de implementações que customizem ações já definidas em sua camada de negócio.

Classificação do FrameColab

O FrameColab, segundo os conceitos descritos na seção 2.4.3, pode ser classificado como:

- Quanto ao domínio: *framework* de domínio, uma vez que ele encapsula conceitos do projeto MOrFEu e os utiliza para projetar Ambientes Colaborativos;

- Quanto à estrutura: *framework* com arquitetura em camadas e com arquitetura MVC, uma vez que utiliza tais conceitos em sua arquitetura;

- Quanto ao uso: *framework* caixa-cinza, uma vez que através de *hot-spots* usuários implementam interfaces para caracterizar as especificidades de um Vcom (característica de caixa branca); e também pode ser combinado diversas classes para projetar Vcoms (característica de caixa preta).

5.4 CONSDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo apresentou a formalização da proposta deste trabalho: FrameColab. Foi descrito a metodologia de desenvolvimento utilizada, passando pelas principais fases da Engenharia de Software, utilizando as boas práticas para a elaboração da proposta.

Vale ressaltar a grande importância do referencial teórico, descrito no Capítulo 2. Através de tais conceitos e teorias tiveram grande influência na elaboração e rumos da proposta deste trabalho.

A partir do projeto java gerado neste trabalho (*framework*), mostrou-se que o projeto de ambientes colaborativos é facilitado, uma vez que já possui toda uma arquitetura, API, classes, bibliotecas, persistência e outros definidos para serem utilizados.

Por fim, é importante enfatizar também a abordagem diferenciada que o FrameColab proporciona, conforme descrito os conceitos atrelados ao projeto MOrFEu, no Capítulo 4.

CAPÍTULO 6 UM PROTÓTIPO PARA O FRAMECOLAB

6.1 INTRODUÇÃO

Este capítulo retrata as fases de materialização da proposta formalizada no capítulo anterior, como forma de construir um protótipo funcional para elaboração de Ambientes Colaborativos. Também mostra em detalhes, um conjunto de espaços virtuais gerados a partir do protótipo construído.

6.2 DETALHAMENTO DO PROTÓTIPO

Conforme relatado na Metodologia de desenvolvimento descrita na seção 5.3.1 do capítulo anterior, este capítulo apresenta as fases de Implementação, Testes e Implantação da proposta do FrameColab, como forma de materializar um protótipo.

Os modelos conceituais, diagramas de classes do domínio, e todas outras definições descritas na Análise e no Projeto (seções 5.3.1.2 e 5.3.1.3, respectivamente) foram incorporadas na fase de Implementação para construção deste protótipo (incorporadas em menor escala, obviamente, por tratar-se de um protótipo). Assim, não faz sentido repetir tais fases nesta seção, uma vez que já foram detalhadas no capítulo anterior. Como forma de diminuir um pouco o escopo do protótipo, não foram implementadas algumas classes de domínio previstas na proposta do FrameColab:

- Amigos: o protótipo não materializa usuários se relacionando, formando redes sociais;

- AreaInteresse: trata-se de uma categorização que pode ser atribuída a usuários e Vcoms, que também não foi implementada;

- Voto: mecanismos de permitir usuários votarem em UPIs publicadas. Alguns Ambientes Colaborativos permitem tal feito, como por exemplo, agregadores de notícias sociais. Também não foi implementado como forma de enxugar o escopo do protótipo.

6.2.1 Implementação

A etapa de projeto fornece algumas das principais definições a serem seguidas para implementar um sistema, de modo que a implementação é considerada satisfatória desde que cumpra tudo o que estiver descrito no projeto. Nesta seção, a implementação, relata alguns dos principais detalhes de implementação relevantes a serem detalhados.

Diagrama de artefatos

Em (Booch, Rumbaugh, & Jacobson, 2005), é dito que diagramas de artefatos:

> *são empregados para modelagem da visão estática de implementação de um sistema, envolvendo bibliotecas, componentes, executáveis, entre outros artefatos. Mostra também a organização e as dependências existentes entre um conjunto de artefatos.*

Na Figura 6.1 é retratado alguns dos principais artefatos (e suas relações de dependência) do código fonte do protótipo e na Tabela 9, suas descrições.

Tabela 9 - Descrição dos principais artefatos do protótipo

Artefato	Descrição
web	Pacote Java que reúne todas classes que envolvem questões relacionadas a interface Web. Exemplo: criação dinâmica de componentes visuais, manipulação de *templates*, etc.
negocio	Pacote Java que reúne todas classes que definem regras de negócio. Exemplo: publicar UPI é uma implementação contida no pacote negocio.
persistencia	Pacote Java que reúne todas classes que tratam acesso a dado.
util	Pacote Java que reúne todas classes que fornecem utilidades ao protótipo. Exemplo: funções para ler XML, funções para ler parametrizações em arquivo, etc.
hotspot	Pacote Java que reúne todos *hot-spots*.
dominio	Pacote Java que reúne todas classes de domínio (Upi, VersaoUpi, Publicacao, etc).

log4j	Biblioteca Java para geração de *logs*[19].
XStream	Biblioteca Java para manipulação de XML.
jDOM	Biblioteca Java para manipulação de XML.
Hibernate	*Framework* de persistência.
Zkoss	*Framework* de interface.

Os artefatos controle, producao, vcom e cadastro são pacotes java que agrupam classes java relacionadas aos pacotes descritos no diagrama de pacotes (ver Figura 5.9, no Projeto do FrameColab). Exemplo: o subpacote vcom, do pacote negocio, conterá regras de negócio que envolvem todas as classes pertencentes ao pacote vcom (classes Vcom e Funcionalidade, conforme Figura 5.10).

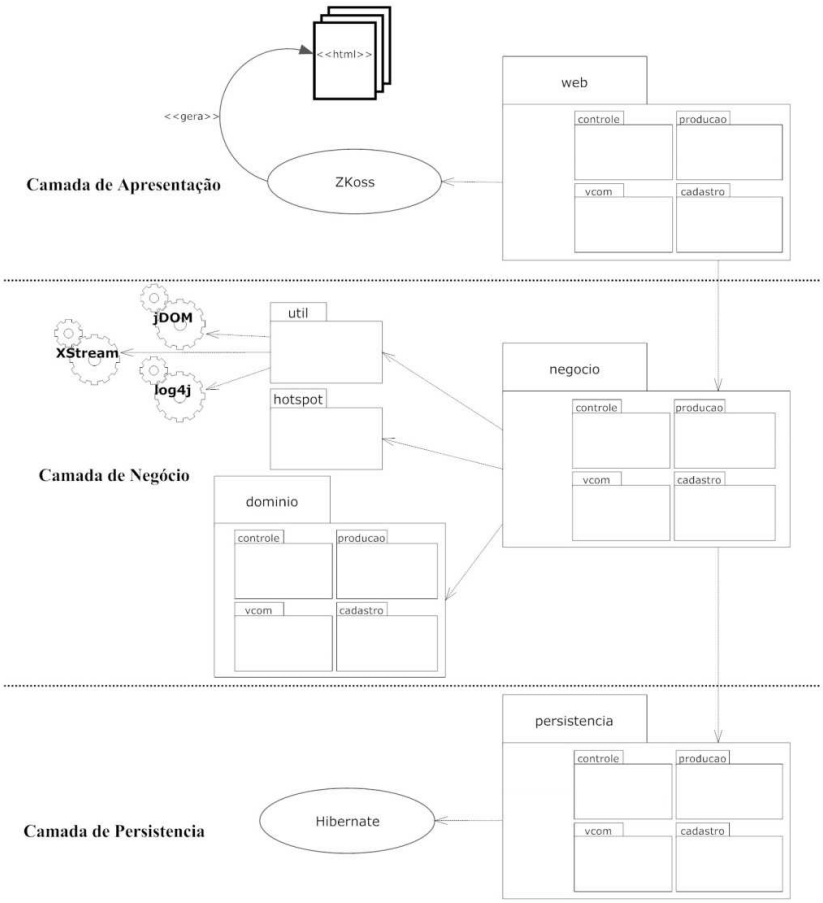


Figura 6.1 - Diagrama de artefatos do protótipo

[19] Expressão utilizada para descrever o processo de registro de eventos relevantes num sistema computacional. Esse registro pode ser utilizado para restabelecer o estado original de um sistema ou para que um administrador conheça o seu comportamento no passado. Um arquivo de *log* pode ser utilizado para auditoria e diagnóstico de problemas em sistemas computacionais.

API provida pelo protótipo

Conforme retratado na Figura 6.1, e definido na proposta do FrameColab, este possui a camada de apresentação, a camada de negócio e a camada de persistência, que são representadas no código fonte pelos pacotes java Web, negocio e persistencia, respectivamente. Em cada pacote, há uma API de modo a facilitar o reuso do código dentro do protótipo, além de permitir facilmente sua expansão e seu uso em *hot-spots*.

A Figura 6.2 ilustra um trecho da API presente no pacote negocio e subpacote producao. No trecho, é mostrado dois métodos relacionados ao negócio do pacote de produção: busca de publicações e ação de publicar. Vale citar que todo código do de todo protótipo foi comentado utilizando Javadoc[20].

```
package morfeu.negocio.producao;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import org.zkoss.zul.Messagebox;
import morfeu.dominio.cadastro.Usuario;
import morfeu.dominio.producao.Upi;
import morfeu.dominio.producao.VersaoUpi;
import morfeu.dominio.vcom.Vcom;
import morfeu.hotspot.producao.IPublicacaoHotSpot;
import morfeu.persistencia.HibernateUtil;
import morfeu.persistencia.dto.PublicacaoDto;
import morfeu.persistencia.dto.UpiDto;
import morfeu.persistencia.dto.VersaoUpiDto;
import morfeu.util.xml.XStreamUtil;
import morfeu.web.FrameworkUtil;

public class Publicacao {

    IPublicacaoHotSpot publicacaoHotSpot;

    /**
     * Retorna uma publicacao a partir de um id de um componente visual
     * @param idComponente id do componente visual a qual a publicação está amarrada
     * @vcom Vcom a ser usado
     * @return Publicação a ser buscada
     */
    public static morfeu.dominio.producao.Publicacao buscaPublicacao(int idComponente, Vcom vcom)

    /**
     * Publica uma versão de upi ou cadastra e publica uma versão de upi; A replica pode ser adicionada também.
     * @param upi
     * @param versaoUpi
     * @param replica
     * @param vcom
     * @param idComponente
     * @param substituir
     * @param dados
     */
    public static void publicaUpi(Upi upi, VersaoUpi versaoUpi, morfeu.dominio.producao.Publicacao replica, Vcom vcom, int idComponente, boolean substituir, Object dados)
```

Figura 6.2 - Exemplo de alguns métodos da API do pacote neogocio e subpacote poducao

Implementação de um *hot-spot*

A Figura 6.3 ilustra um esboço de implementação de um *hot-spot*. Cada *hot-spot* possui uma interface, que deve ser implementada. Na classe abaixo, é retratado a implementação do *hot-spot* referenciado na configuração global do *blog* descrito na Figura 5.3.

---

[20] É um gerador de documentação criado pela Sun Microsystems para documentar a API dos programas em Java, a partir do código-fonte. O resultado é expresso em HTML. É constituído, basicamente, por algumas marcações muitos simples inseridas nos comentários do programa.

```java
package exemploshotspots;

import morfeu.dominio.producao.VersaoUpi;
import morfeu.dominio.vcom.Vcom;
import morfeu.hotspot.producao.IPublucacaoHotSpot;

public class HotSpotPublicacaoImpl implements IPublucacaoHotSpot {

    @Override
    public void publicaUpi(VersaoUpi versaoUpi, Vcom vcom, Object dados) {
        // TODO implementação da rotina do hot-spot de Publicacao



    }
}
```

Figura 6.3 - Exemplo de implementação de um *hot-spot*

Com o uso de tal *hot-spot*, pode-se estender o comportamento da rotina de publicação de UPI. Colocando-se uma rotina de disparo de email, por exemplo, pode ser implementado para disparar email sempre que uma UPI for publicada.

Arquivo de configuração do protótipo (framework.properties)

A Figura 6.4 apresenta um trecho do arquivo de configuração do protótipo. Algumas das configurações de funcionamento são parametrizadas e, tais configurações estão dispostas em um arquivo (para aumentar a flexibilidade para ajustes). Exemplo de configurações: mensagens gerais produzidas *framework*, diretórios que contém os XMLs, formatação de textos, conteúdo de botões, dentre outros.

```
XMLConfig = C:\\Users\\Eleu\\workspace\\morfeuzk\\WebContent\\XML\\VcomConfig\\
XMLDados = C:\\Users\\Eleu\\workspace\\morfeuzk\\WebContent\\XML\\VcomDados\\
XMLVcom = C:\\Users\\Eleu\\workspace\\morfeuzk\\WebContent\\XML\\Vcom\\
CSS = C:\\Users\\Eleu\\workspace\\morfeuzk\\WebContent\\CSS\\
PrazoAte = Prazo de postagem até: %s
PrazoAPartirDe = Prazo de postagem a partir de: %s
Prazo = Prazo de postagem a partir de %s e até %s
TituloSucesso = Informação
Sucesso = Dados atualizados com sucesso!
TituloErro = Erro
Erro = Erro durante o processo de salvamento! Detalhes: %s
LimitePostagens = O limite de postagens, que é %s, já foi atingido!
TituloLimitePostagens = Atenção!
LoginSucesso = Login efetuado com sucesso!
LoginErro = Erro ao efetuar o login
MeusVcomsLogin = Faça login para visualizar seus vcoms
Logout = Logout realizado com sucesso!
TituloLogout = Logout
BotaoHome = Home
BotaoLogin = Login
BotaoPerfil = Meu Perfil
BotaoCriarVcom = Criar Vcom
BotaoAmigos = Amigos
BotaoLogout = Logout
DetalhesUsuario = Faça login para visualizar suas informações
```

Figura 6.4 - Trecho do arquivo de configuração do protótipo do FrameColab

Classe DAO Genérica

Buscando promover a reutilização, o protótipo contém um classe genérica DAO. Ela provê recursos básicos as classes de domínio (consultas, exclusões, alterações e outros) evitando que cada classe de domínio acrescentada implique em uma nova classe DAO a ser implementada. A Figura 6.5 representa a interface que tal classe implementa:

```
package morfeu.persistencia.dao;

import java.util.List;
/**
 * Interface referente a classe GenericoDao
 * @author Eleu
 *
 * @param <T> Tipo de objeto a ser usado para persistir
 */
abstract public interface IGenericoDao<T> {
    public T retornaUnico(Object id);
    public List<T> retornaLista(Map<String, Object> filtro, Map<String, ordenacaoEnum> ordenacao);
    public List<T> retornaTodos();
    public T salvar(Object objeto);
    public void excluir(Object objeto);
}
```

Figura 6.5 - Interface DAO genérica

Com isso, conforme Figura 6.6, para adição de uma nova classe de domínio basta utilizar herança. Isso é uma decisão de implementação importante, pois permite a fácil incorporação de novas classes de domínio no protótipo (facilitando trabalhos posteriores, que podem ser realizados buscando expandir o *framework*).

```
package morfeu.persistencia.dto;

import morfeu.dominio.producao.Upi;
import morfeu.persistencia.dao.GenericoDao;

public class UpiDto extends GenericoDao<Upi> {

}
```

Figura 6.6 - Exemplo de uma classe DTO, utilizando a classe genérica DAO

Diagrama de Navegação

A Figura 6.7 apresenta um diagrama de navegação tendo como objetivo relatar as páginas contidas no protótipo gerado e a navegabilidade entre elas. O acesso inicial dar-se-á através da página inicial do ambiente. Como visitante (sem fazer autenticação), o usuário pode acessar a página “Gera Vcom”, que é uma página que recebe o Vcom a ser visitado como parâmetro e o constrói em tempo de execução (controles visuais, eventos,

e outros componentes de uma página Web são gerado dinamicamente, em cada acesso) - criação de acordo com a descrição estrutural do Vcom, suas características globais e os *hot-spots*. A construção em tempo de execução do Vcom permite que com qualquer alteração em, por exemplo, na configuração global do Vcom, repercute instantaneamente no ambiente. Autenticando-se, o usuário tem mais opções de ações no ambiente: criar UPIs, passando a interagir com os vários Vcoms instanciados no ambiente.

Caso seja desejado criar um Vcom, deve-se começar através da página “Cadastro Vcom”. Após isso, deve-se criar as funcionalidades do Vcom em questão, através da página “Cadastro de Funcionalidade”. Após as funcionalidades cadastradas, é o momento de cadastrar os possíveis perfis presentes no Vcom, sendo feito através da página “Cadastro de Perfil”. Depois dos perfis previamente cadastrados, é o momento da associação entre perfil e funcionalidade, atribuindo um acesso para cada associação, sendo feito através da página “Cadastro de Permissão”. Após este passo, o Vcom já pode ser considerado instanciado e acessado (considerando-se que foi criado os arquivos de configurações - estrutural e global). Eventualmente usuários podem requerer acesso ao Vcom ou terem acessos revogados, que é provido pela página “Cadastro de Acesso”.

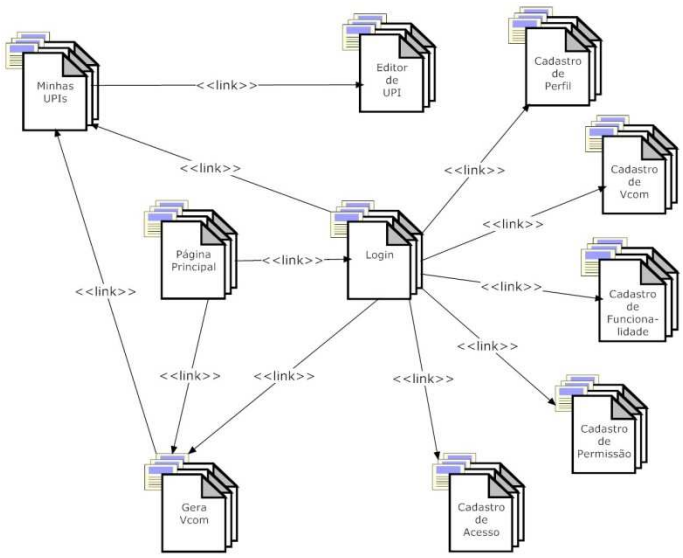


Figura 6.7 - Diagrama de navegação do protótipo do FrameColab

A próxima seção ilustra algumas telas do ambiente provido pelo protótipo, facilitando o entendimento do diagrama anteriormente citado.

Principais telas

Nesta parte é apresentado o visual de algumas das principais telas (páginas) do protótipo baseado na proposta do FrameColab.

A página inicial do ambiente provido pelo *framework* é retratado na Figura 6.8. A ação de visitar qualquer Vcom contido em “Todos Vcoms” implicará na chamada da página “Gera Vcom” que receberá o identificador do Vcom a ser gerado, e irá construir, em tempo de execução, o Vcom de acordo com suas configurações (estrutural e global), conforme já descrito anteriormente. Qualquer alteração nestas configurações, implica alterações instantâneas no Vcom.

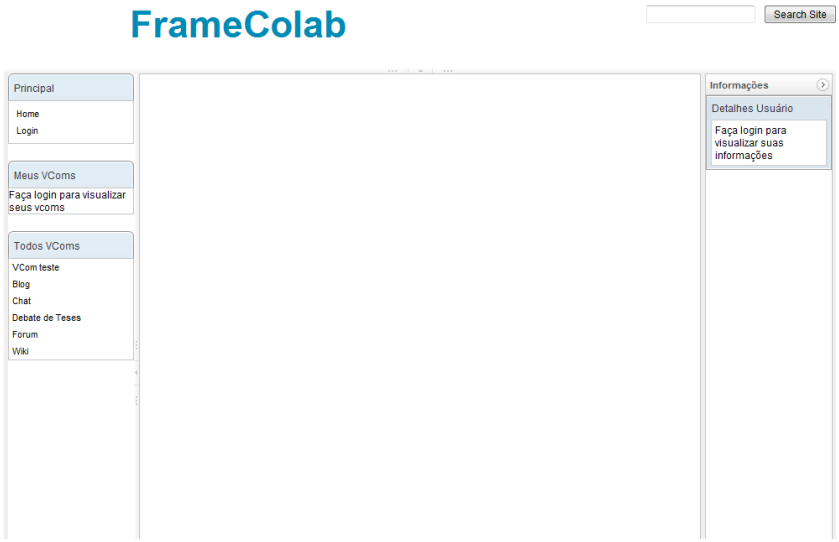


Figura 6.8 - Tela inicial

Para instanciar um Vcom, é preciso acessar a página “Cadastro de Vcom” (Figura 6.10) e preencher os campos. O campo XML é o nome do arquivo de configuração estrutural e global que o Vcom em questão deverá usar, e os caminhos que contém o XML é descrito no arquivo de configuração do protótipo - framework.properties (citado anteriormente).

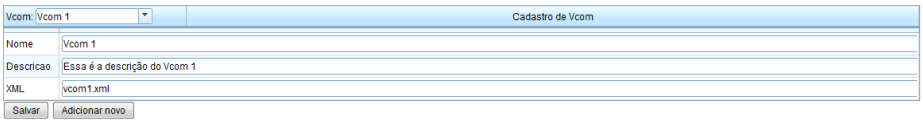


Figura 6.9 - Cadastro de Vcom

O cadastro das funcionalidades do Vcom é feito através do acesso a página “Cadastro da Funcionalidade” (Figura 6.10), onde o usuário cadastra o código de cada funcionalidade. Tal código é utilizado na linguagem de descrição de Vcoms (seção 5.2.2), através do atributo “codigo” da marcação “secao” (ver Tabela 7).

Figura 6.10 - Cadastro de funcionalidade de um Vcom

Os perfis (papéis) que os usuários poderão assumir no Vcom projetado são criados através da página “Cadastro de Perfil” (conforme Figura 6.11).

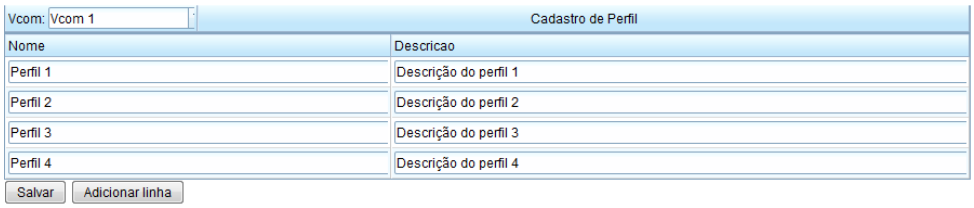


Figura 6.11 - Cadastro de Perfil

A amarração entre o perfil, a funcionalidade e o acesso é concedido através da página “Cadastro de Permissão” (ver Figura 6.12). Para cada funcionalidade de um dado perfil de um Vcom recebe o acesso. O acesso pode ser classificado em três opções (conforme previsto na Figura 4.5 - Uma representação da estrutura de um Vcom): consumidor (somente tem acesso a leitura), colaborador (acesso a leitura e escrita) e bloqueado (não tem acesso a leitura e nem a escrita).

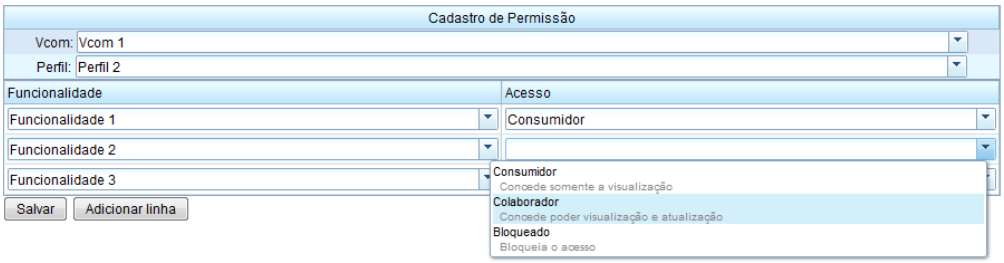


Figura 6.12 - Cadastro de Permissão

Vcoms podem ter livre acesso de usuários, ou podem ter acesso restrito. Tal restrição é controlada pela página "Cadastro de Acesso" (Figura 6.13). Em tal tela, concede-se acesso a usuários que requisitaram em algum momento, atribuindo-se um perfil de acesso.

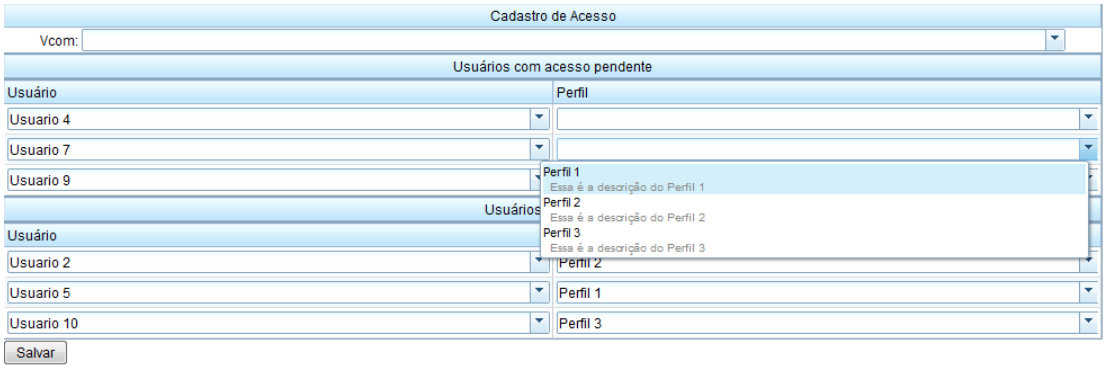


Figura 6.13 - Cadastro de Acesso

Projeto Web no Eclipse do protótipo desenvolvido

Por fim, a Figura 6.14 ilustra o projeto java do protótipo do FrameColab no Eclipse[21] (um conjunto de artefatos java). Observa-se na figura os pacotes descritos neste capítulo e também as pastas que contém artefatos para construção de Vcoms (XMLs, CSS, Imagens, *Templates*, etc).

[21] IDE (de ***I**ntegrated **D**evelopment **E**nvironment*) feita em Java, código aberto, e largamente utilizada em projetos Java.

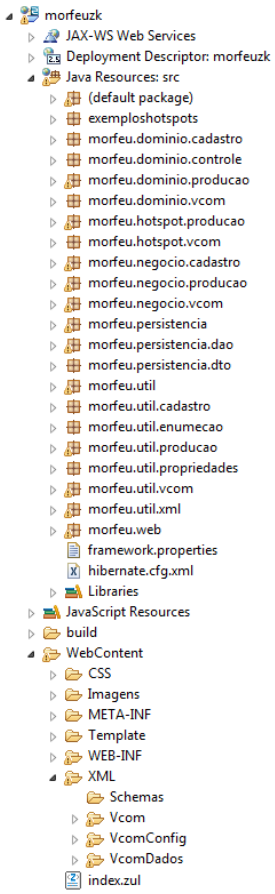


Figura 6.14 - Projeto Web Java no Eclipse do protótipo do FrameColab

6.2.2 Testes e Implantação

Testes são de grande importância em projetos Web, no entanto, não foi utilizado nenhum formalismo para executar os testes neste protótipo uma vez que foge do escopo do trabalho. Os testes realizados no trabalho foram focados em validar aspectos de negócio (conceitos do domínio e facilitadores para construção de ambientes), não abordando questões de desempenho e usabilidade, por exemplo.

Já o processo de implantação do protótipo é simples. Como o *framework* desenvolvido é um projeto Web java, basta exportá-lo como arquivo WAR[22] (conforme ilustrado na Figura 6.15, que é acessado através do botão direito do *mouse* no Eclipse). Feito isso, a implantação dar-se-á quando colocado o arquivo WAR gerado no diretório das

[22] WAR (de ***Web application ARchive***) é um arquivo JAR usado para distribuição de coleções de JavaServer Pages, servlets, classes Java, arquivos XML, tag libraries e páginas web estáticas (HTML e arquivos relacionados) que juntos constituem uma aplicação web.

aplicações Web do servidor de aplicação. No caso deste trabalho, foi adotado o TomCat, e tal diretório é o "*webapp*" que fica localizado na raiz da instalação do aplicativo.

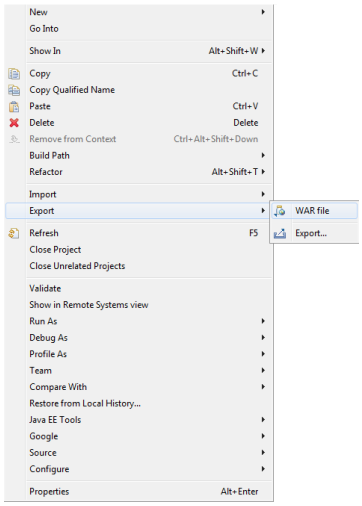


Figura 6.15 - Opção de gerar arquivo WAR, no Eclipse, acessada através do botão direito do *mouse*

Quanto ao banco de dados, como o protótipo desenvolvido utiliza Hibernate fazendo uso do recurso "hbm2ddl.auto", é garantido uma liberdade total quanto ao banco: todas tabelas são geradas automaticamente no primeiro acesso.

6.3 APLICAÇÕES DO PROTÓTIPO

Nesta seção será descrito alguns Ambientes Colaborativos criados a partir do protótipo do FrameColab desenvolvido.

6.3.1 Chat

Um *chat*, que em português significa conversação, ou bate-papo (termo usado no Brasil), é um neologismo para designar aplicações de conversação em tempo real. Trata-se de um ambientes tradicional e amplamente disseminado na Web.

A Figura 6.16 retrata uma possível descrição estrutural de um *chat*. A linha 3 descreve uma seção, com múltiplos usuários postando, com ordenação em ordem decrescente por data (mais recentes primeiro), sem prazo limite para postagem e que tem código de acesso 10.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <chat>
3      <secao titulo="Mensagens" codigo="10" cer="nao" ordem="data_decrescente">
4          <repeticao quantidade="9999">
5              <upi titulo="nao" editor="nao" textarea="nao" autorelacionamento="nao" />
6          </repeticao>
7      </secao>
8  </chat>
```

Figura 6.16 - Configuração estrutural para um *chat* exemplo

Dentro da seção da linha 3, pode ser colocada 9999 UPIs (conforme o atributo "quantidade" da linha 4). Tais UPIs não tem título (atributo "titulo" na linha 5) e não se auto-relacionam com outras UPIs (atributo "autorelacionamento" na mesma linha).

A Figura 6.17 apresenta uma possível configuração global para o chat criado anteriormente. Título é definido (linha 3), CSS utilizado (linha 4), formatação das mensagens de autoria (linha 5), taxa de atualização (linha 11 - no caso, 3 segundos), etc. Nesse caso não foi implementado nenhum *hot-spot*, por tratar-se de um Vcom muito simples.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <vcomconfig>
3      <titulo>Meu chat</titulo>
4      <css>chat.css</css>
5      <mensagemAutoria>[%s] %s disse: </mensagemAutoria>
6      <formatoData>dd/MM/yyyy, HH:mm:ss</formatoData>
7      <privado>nao</privado>
8      <permiteanonimo>nao</permiteanonimo>
9      <permiteadesao>sim</permiteadesao>
10     <ativado>sim</ativado>
11     <atualizacao>3</atualizacao>
12 </vcomconfig>
```

Figura 6.17 - Configuração global para um *chat* exemplo

A Figura 6.18 mostra o *chat* gerado de acordo com as configurações definidas. É possível verificar a formatação de datas, o título do Vcom, o nome da seção e a mensagem de autoria estão conforme descrito na configuração do Vcom, bem como sua estrutura que segue a configuração estrutural. O ato do usuário digitar uma mensagem e "clicar" no botão salvar significa que o *framework* irá criar uma UPI e publicá-la de maneira transparente. No entanto, caso o usuário queira re-publicar uma de suas UPIs, basta "clicar" no botão "Minhas UPIs", conceito retratado na seção 4.3.1.

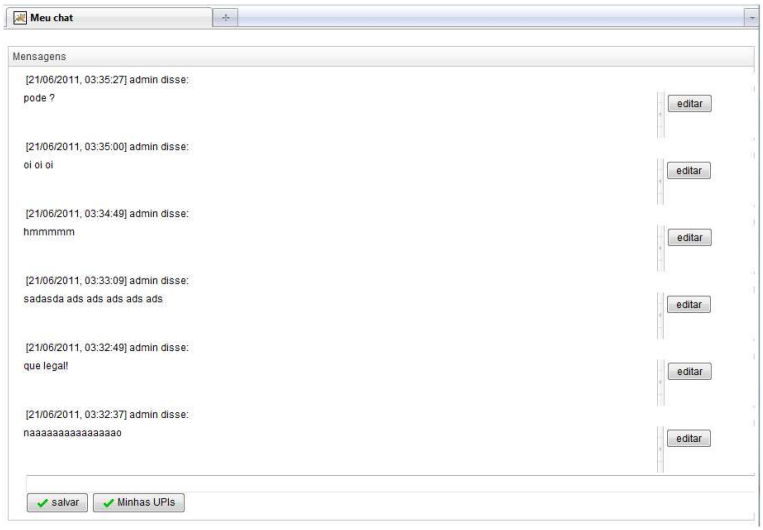


Figura 6.18 - Exibição de um *chat* exemplo gerado pelo protótipo do FrameColab

6.3.2 Fórum

Outro ambiente colaborativo tradicional que será abordado neste capítulo é o fórum. Também chamados "fórum de discussão", buscam promover debates através de mensagens publicadas abordando uma mesma questão ("tópico"). Basicamente possuem duas divisões hierárquicas a primeira faz a divisão por tópicos e a segunda uma divisão mensagens de cada tópico (que são os "*posts*"). As mensagens ficam ordenadas decrescentemente por data, e os tópicos ficam ordenados pelo título. Tal organização hierárquica, é representado pelas linhas 4 e 6, na Figura 6.19, da configuração estrutural do Vcom criado.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <forum>
3      <repeticao quantidade="9999">
4          <secao titulo="Topicos" codigo="40" cer="nao" ordem="titulo_crescente">
5              <upi titulo="sim" editor="sim" textarea="sim" autorelacionamento="nao" />
6              <secao titulo="Mensagens" codigo="41" cer="nao" ordem="arvore">
7                  <repeticao quantidade="9999">
8                      <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="sim" />
9                  </repeticao>
10             </secao>
11         </secao>
12     </repeticao>
13 </forum>
```

Figura 6.19 - Configuração estrutural para um fórum exemplo

Já a Figura 6.20 ilustra a configuração global do fórum projetado. Algumas diferenças foram feitas, em comparação ao *chat* anteriormente descrito: título, mensagem de

autoria, formato de data, permissão de adesão e taxa de atualização (não atualiza - ambiente síncrono).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<vcomconfig>
    <titulo>Meu forum</titulo>
    <css>forum.css</css>
    <mensagemAutoria>[%s] %s escreveu: </mensagemAutoria>
    <formatoData>EEE, dd/MM/yyyy, HH:mm:ss</formatoData>
    <privado>nao</privado>
    <permiteanonimo>nao</permiteanonimo>
    <permiteadesao>nao</permiteadesao>
    <ativado>sim</ativado>
    <atualizacao>-1</atualizacao>
</vcomconfig>
```

Figura 6.20 - Configuração global para um fórum exemplo

E a Figura 6.21 representa o fórum gerado pelo protótipo do FrameColab a partir de tais configurações. Nota-se os tópicos e as mensagens com sua hierarquia realçada, editor WYSIWYG na criação de tópicos e possibilidade de mensagens se auto-relacionarem.

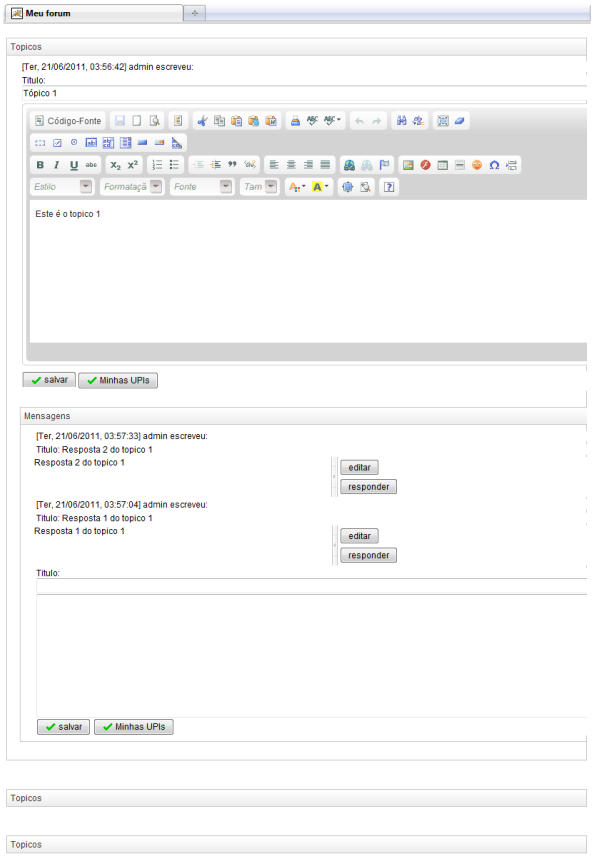


Figura 6.21 - Exibição de um fórum exemplo gerado pelo protótipo do FrameColab

6.3.3 Júri Simulado

Com a flexibilidade da descrição estrutural de Vcoms, fica viável e mais simples a criação de soluções inovadoras e diferenciadas. Um exemplo é o ambiente de um “Júri Simulado”. Trata-se da Arquitetura Pedagógica que propõem a simulação de um júri presencial na Web, proposta por (Nevado, Carvalho, & Menezes, 2007). Em um júri real, há a figura do réu, dos jurados, do juiz, da promotoria e do advogado de defesa. No Júri Simulado há atores semelhantes (Real & Menezes, 2008):

- Réu: não é um ator, e sim o que será estudado. Sendo assim, pode ser uma proposta, uma teoria, etc;

- Acusação: é um grupo de usuários designados a serem contra o réu;

- Defesa: é um grupo de usuários designados a serem a favor do réu;

- Jurados: também sendo um grupo de usuários que analisam as ponderações dos dois lados (acusação e defesa) e votam favoráveis a um lado;

- Juiz: com base no que foi discutido, na exposição dos jurados, o usuário juiz apresenta seu veredicto.

A Figura 6.22 apresenta um diagrama de atividades de exemplo de um Júri Simulado simplificado. O diagrama foi construído enfatizando os 3C do Modelo de Colaboração descrito em seções anteriores, como forma de evidenciar a abordagem adotada neste trabalho.

Inicialmente, para criar-se o Júri, o professor define os participantes dos grupos: Juiz, Acusação, Defesa e Jurados. Definidos os grupos, O grupo da acusação apresenta os argumentos e o grupo de defesa também. Quando o prazo de apresentação de tais argumentos termina, inicia-se uma discussão, com intervenções do juiz (quando necessário). Após o fim do prazo da discussão da acusação e defesa, os jurados passam a ter um prazo para leitura do material e apresentarem seus posicionamentos. Após o prazo dos jurados esgotarem, o juiz lê tudo o que foi publicado e produz um veredicto.

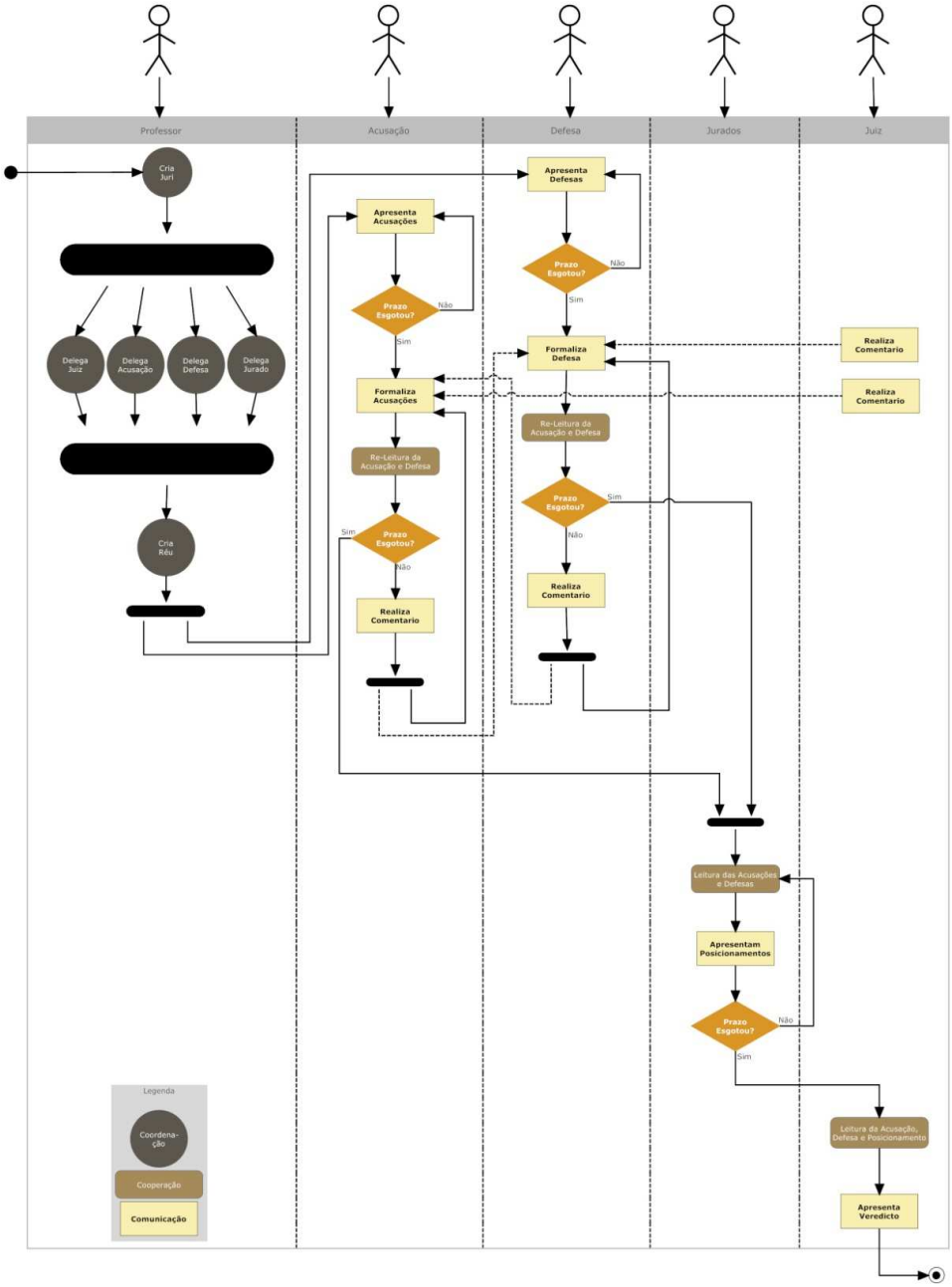


Figura 6.22 - Diagrama de atividades de um Júri Simulado exemplo

Nas linhas 7, 15, 24 e 29 da Figura 6.23, é caracterizado os locais de interação da Acusação, da Defesa, dos Jurados e do Juiz, respectivamente.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<jurisimulado>
    <secao titulo="Réu" codigo="20" cer="nao" prazo_inicio="13/06/2011" prazo_fim="18/06/2011">
        <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="nao" />
    </secao>
    <repeticao quantidade="9999">
        <secao titulo="Acusação" codigo="21" cer="nao" prazo_inicio="19/06/2011" prazo_fim="22/06/2011" ordem="titulo_crescente">
            <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="nao" />
            <secao titulo="Discussão acusação" codigo="22" cer="nao" prazo_inicio="23/06/2011" prazo_fim="25/06/2011" ordem="arvore">
                <repeticao quantidade="9999">
                    <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="sim" />
                </repeticao>
            </secao>
        </secao>
    </repeticao>
    <repeticao quantidade="9999">
        <secao titulo="Defesa" codigo="23" cer="nao" prazo_inicio="19/06/2011" prazo_fim="22/06/2011" ordem="titulo_crescente">
            <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="nao" />
            <secao titulo="Discussão acusação" codigo="24" cer="nao" prazo_inicio="23/06/2011" prazo_fim="25/06/2011" ordem="arvore">
                <repeticao quantidade="9999">
                    <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="sim" />
                </repeticao>
            </secao>
        </secao>
    </repeticao>
    <secao titulo="Jurados" codigo="25" cer="nao" prazo_inicio="25/06/2011" prazo_fim="04/07/2011">
        <repeticao quantidade="7">
            <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="nao" />
        </repeticao>
    </secao>
    <secao titulo="Veredicto" codigo="26" cer="nao" prazo_inicio="05/07/2011" prazo_fim="10/07/2011">
        <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="nao" />
    </secao>
</jurisimulado>
```

Figura 6.23 - Configuração estrutural para um júri simulado exemplo

A Figura 6.24 representa as configuração do Júri Simulado projetado. Vale ressaltar que adesões não são permitidas, uma vez que o administrador do Júri que é o responsável por especificar quem irá participar do júri e em qual grupo (perfil) fará parte.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<vcomconfig>
    <titulo>Meu júri simulado</titulo>
    <css>juri.css</css>
    <mensagemAutoria>[%s] %s disse: </mensagemAutoria>
    <formatoData>EEE, dd/MM/yyyy, HH:mm:ss</formatoData>
    <privado>nao</privado>
    <permiteanonimo>nao</permiteanonimo>
    <permiteadesao>nao</permiteadesao>
    <ativado>sim</ativado>
    <atualizacao>-1</atualizacao>
</vcomconfig>
```

Figura 6.24 - Configuração global para o júri simulado exemplo

A Figura 6.25 ilustra as funcionalidade providas pelo Júri Simulado. Merece destaque a coluna “código”, que deve ser os mesmo referenciados no código estrutural do Vcom projetado.

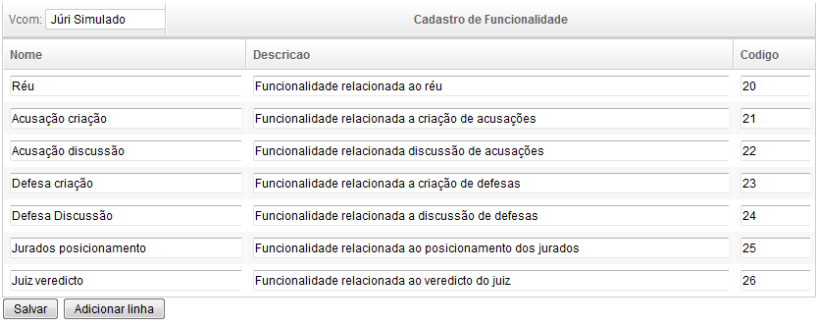


Figura 6.25 - Cadastro de funcionalidade no Júri Simulado

A Figura 6.26 representa as permissões de acesso de um dos perfis do Júri Simulado. A Defesa tem acesso a leitura do réu, leitura da acusação, escrita na discussão das acusações, escrita na criação de defesas, escrita na discussão das defesas, leitura nos posicionamentos dos jurados e leitura no veredicto do juiz.

Figura 6.26 - Permissões de acesso no perfil Defesa para o Júri Simulado

6.3.4 Jigsaw

O Jigsaw (Slavin, 1995), foi desenvolvido por Elliot Aronson em um projeto educacional no Texas. Esta abordagem foi criada para ajudar a construir um ambiente de estudo como uma comunidade onde todos os aprendizes são valorizados, procurando-se eliminar aspectos indesejáveis tal como a competição excessiva entre os participantes primando por aumentar o interesse na cooperação mútua (Sharan, 1999) (Pereira, Castro Júnior, Souza, Mendonça, & Silva, 2002). O foco recai, então, sobre o compartilhamento dos recursos. Pode-se descrever tal método através de quatro estágios genéricos:

- Introdução: nesta fase o professor organiza os grupos Jigsaw, introduz os tópicos, textos, informações ou materiais para ajudar os estudantes no entendimento dos tópicos que vão ser trabalhados e verificar como eles se encaixam com o que foi estudado e como serão importantes no futuro;

- Exploração: os estudantes se reorganizam em outros grupos, chamados de especialistas, para estudar os tópicos em maior profundidade. Nesta fase o professor deve utilizar ferramentas para incentivar os alunos a interagir com os outros;

- Relato e transformação: os estudantes voltam ao grupo original para explicar os tópicos para os companheiros. Deve-se entender primeiro as partes, para ter uma compreensão melhor do todo;

- Integração e avaliação: o que foi obtido pelos alunos em grupos pequenos é integrado com as outras pessoas envolvidas no ambiente de estudo. O resultado é então avaliado.

O diagrama de atividades exposto na Figura 6.27 representa uma possível do método de aprendizagem Jigsaw. Inicialmente o professor cria os grupos de discussão, submete os materiais a serem discutidos e escolhe os participantes de cada grupo. Após isso, cada grupo (dito grupo de “especialistas”) irá discutir o assunto entre si durante um prazo. Após o término do prazo cada grupo expõe suas conclusões sobre o assunto debatido e é iniciada a leitura por todos. Todos grupos, agora unidos, debatem sobre as questões envolvidas e cada grupo “especialista” responde na medida do possível. Após um prazo, os participantes fazer uma avaliação do aprendizado e termina-se o estudo.

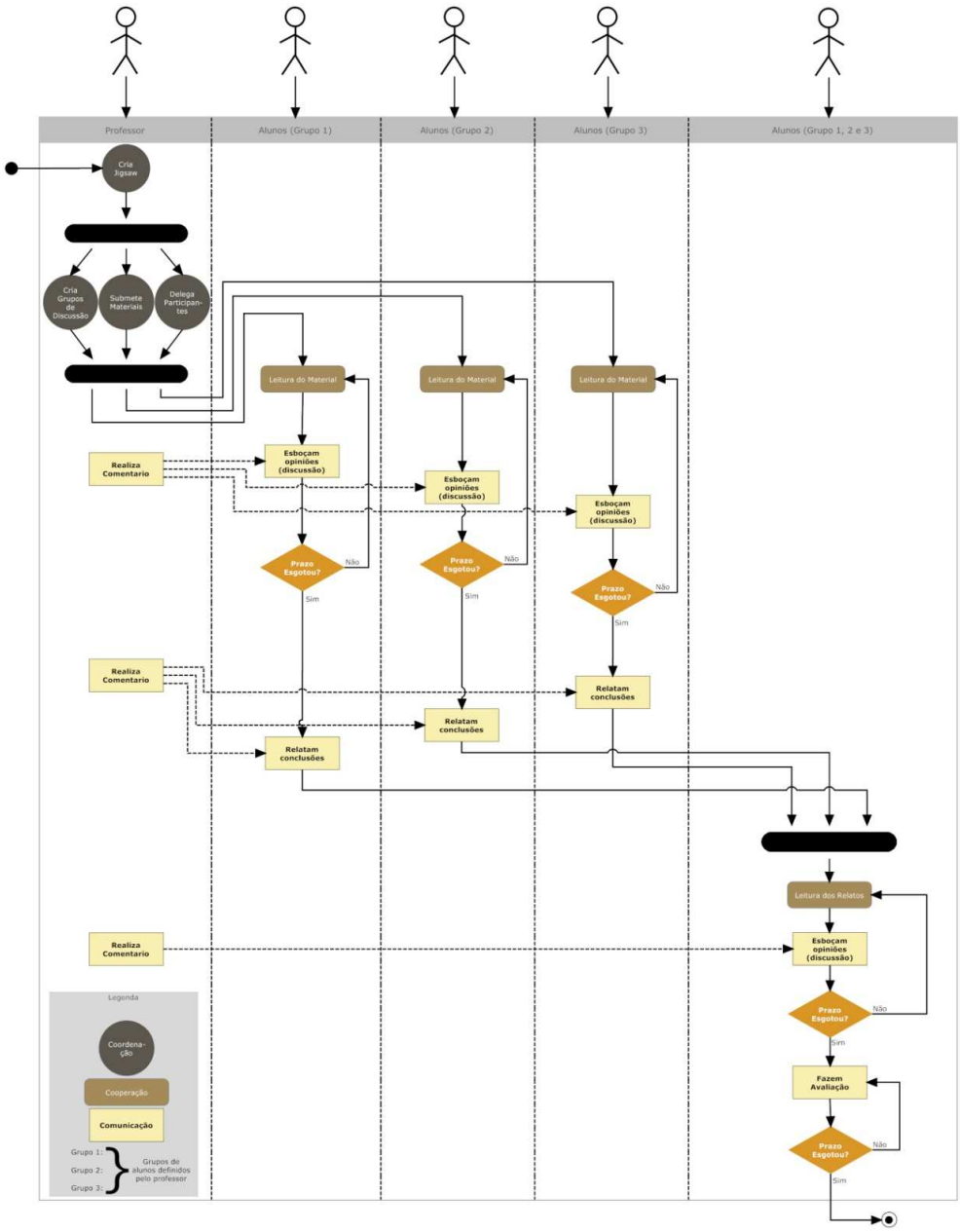


Figura 6.27 - Diagrama de atividades de um possível Jigsaw

A Figura 6.28 representa uma configuração estrutural de uma possível representação de um Jigsaw através do protótipo. Em tal Jigsaw, há três grupos “especialistas”, que são representados pelas linhas 3, 16 e 29. Nestas seções são realizadas discussões sobre o assunto em pauta durante um prazo e, terminando o prazo, os membros do grupo registram suas conclusões acerca do estudo. Após tal etapa, na linha 42, usuários consomem as informações geradas e discutem durante um período. Após tais discussões, os usuários relatam suas avaliações acerca do aprendizado realizando, na linha 47.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<jigsaw>
    <secao titulo="Topico de Estudo 1" codigo="30" cer="nao" prazo_inicio="19/06/2011" prazo_fim="22/06/2011">
        <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="nao" />
        <secao titulo="Discussão do Tópico de Estudo 1" codigo="31" cer="nao" prazo_inicio="23/06/2011" prazo_fim="25/06/2011" ordem="arvore">
            <repeticao quantidade="9999">
                <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="sim" />
            </repeticao>
        </secao>
        <secao titulo="Conclusões do Tópico de Estudo 1" codigo="32" cer="nao" prazo_inicio="26/06/2011" prazo_fim="27/06/2011" ordem="arvore">
            <repeticao quantidade="9999">
                <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="sim" />
            </repeticao>
        </secao>
    </secao>
    <secao titulo="Topico de Estudo 2" codigo="33" cer="nao" prazo_inicio="19/06/2011" prazo_fim="22/06/2011">
        <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="nao" />
        <secao titulo="Discussão do Tópico de Estudo 2" codigo="34" cer="nao" prazo_inicio="23/06/2011" prazo_fim="25/06/2011" ordem="arvore">
            <repeticao quantidade="9999">
                <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="sim" />
            </repeticao>
        </secao>
        <secao titulo="Conclusões do Tópico de Estudo 2" codigo="35" cer="nao" prazo_inicio="26/06/2011" prazo_fim="27/06/2011" ordem="arvore">
            <repeticao quantidade="9999">
                <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="sim" />
            </repeticao>
        </secao>
    </secao>
    <secao titulo="Topico de Estudo 3" codigo="36" cer="nao" prazo_inicio="19/06/2011" prazo_fim="22/06/2011">
        <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="nao" />
        <secao titulo="Discussão do Tópico de Estudo 3" codigo="37" cer="nao" prazo_inicio="23/06/2011" prazo_fim="25/06/2011" ordem="arvore">
            <repeticao quantidade="9999">
                <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="sim" />
            </repeticao>
        </secao>
        <secao titulo="Conclusões do Tópico de Estudo 3" codigo="38" cer="nao" prazo_inicio="26/06/2011" prazo_fim="27/06/2011" ordem="arvore">
            <repeticao quantidade="9999">
                <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="sim" />
            </repeticao>
        </secao>
    </secao>
    <secao titulo="Esboço de Opiniões" codigo="39" cer="nao" prazo_inicio="28/06/2011" prazo_fim="30/06/2011" ordem="arvore">
        <repeticao quantidade="9999">
            <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="sim" />
        </repeticao>
    </secao>
    <secao titulo="Avaliações" codigo="40" cer="nao" prazo_inicio="01/07/2011" prazo_fim="03/07/2011" ordem="data_decrescente">
        <repeticao quantidade="9999">
            <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="nao" />
        </repeticao>
    </secao>
</jigsaw>
```

Figura 6.28 - Configuração estrutural de um Jigsaw exemplo

A Figura 6.29 esboça um exemplo de configuração global para o Jigsaw a ser projetado. A linha 9 indica que não é permitido a adesão ao Vcom, uma vez que quem cria os grupos de acesso é o professor. As linhas 12 e 13 indicam possíveis implementações de *hot-spots* de UsuarioVcom e Permissao, respectivamente. Em tais implementação, pode-se, por exemplo, colocar rotinas de disparo de emails como forma de avisar que determinado usuário teve acesso permitido no Jigsaw e com determinado perfil.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<vcomconfig>
    <titulo>Meu jigsaw</titulo>
    <css>jigsaw.css</css>
    <mensagemAutoria>[%s] %s disse: </mensagemAutoria>
    <formatoData>EEE, dd/MM/yyyy, HH:mm:ss</formatoData>
    <privado>nao</privado>
    <permiteanonimo>nao</permiteanonimo>
    <permiteadesao>nao</permiteadesao>
    <ativado>sim</ativado>
    <atualizacao>-1</atualizacao>
    <hotspotusuariovcom>exemploshotspots.HotSpotUsuarioVcom</hotspotusuariovcom>
    <hotspotpermissao>exemploshotspots.HotSpotPermissao</hotspotpermissao>
</vcomconfig>
```

Figura 6.29 - Configuração global de um Jigsaw exemplo

6.3.5 Debate de teses

Outra Arquitetura Pedagógica a ser projetada e analisada através do protótipo é o Debate de Teses (Nevado, Dalpiaz, & Menezes, 2009). Uma possível representação é apresentada na Figura 6.30; inicialmente o administrador cria o debate, delega os usuários pertencentes aos grupos de mediador, argumentador e revisor.

O processo se inicia com o mediador apresentando as teses a serem analisadas. Os argumentadores fazer a leitura, posicionam-se e apresentam sua argumentação para justificar tal fato. Quando o prazo de argumentação terminar, os revisores passam a atuar questionando os argumentos feitos pelos argumentadores. Quando o prazo da revisão termina, os argumentatores tem um novo período para se re-posicionarem acerca das teses e apresentarem sua réplica. Com o fim desse passo, o processo termina. Vale ressaltar que o mediador pode ter participação em qualquer momento, inserindo comentários. O mediador, portanto, é a figura do professor e o restante a dos aprendizes.

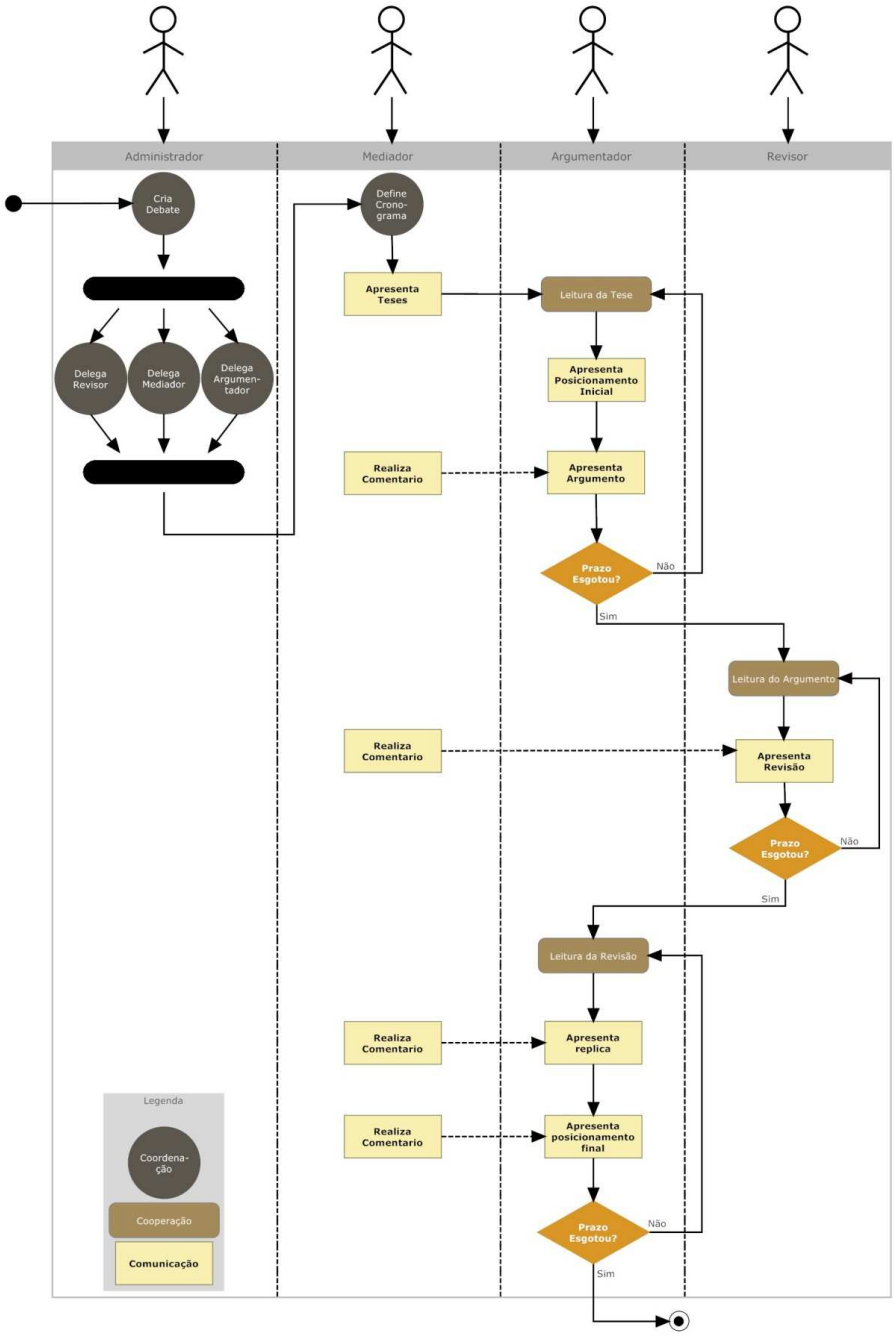


Figura 6.30 - Diagrama de atividades de um possível Debate de Teses

A Figura 6.31 apresenta a descrição estrutural de um Debate de Teses exemplo. As linhas 4, 7, 15 e 23 representam as fases de “apresentação das teses”, “posicionamento”, “revisão” e “réplica”, respectivamente. Cada etapa possui restrição de acesso, previamente definido de acordo com os perfis criados, conforme meio apresentado a pouco.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<debateteses>
    <repeticao quantidade="5">
        <secao titulo="Tese" codigo="10" cer="nao" prazo_inicio="13/06/2011" prazo_fim="15/05/2011">
            <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="nao" />
        </secao>
        <secao titulo="Posicionamento Inicial" codigo="11" cer="sim" prazo_inicio="16/06/2011" prazo_fim="22/06/2011">
            <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="nao" />
            <secao titulo="Comentários" codigo="12" cer="nao" prazo_inicio="16/06/2011" prazo_fim="22/06/2011" ordem="arvore">
                <repeticao quantidade="10">
                    <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="sim" />
                </repeticao>
            </secao>
        </secao>
        <secao titulo="Revisão do Posicionamento" codigo="13" cer="sim" prazo_inicio="23/06/2011" prazo_fim="27/06/2011">
            <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="nao" />
            <secao titulo="Comentários" codigo="14" cer="nao" prazo_inicio="23/06/2011" prazo_fim="27/06/2011" ordem="arvore">
                <repeticao quantidade="10">
                    <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="sim" />
                </repeticao>
            </secao>
        </secao>
        <secao titulo="Réplica" codigo="15" cer="sim" prazo_inicio="28/06/2011" prazo_fim="01/07/2011">
            <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="nao" />
            <secao titulo="Comentários" codigo="16" cer="nao" prazo_inicio="28/06/2011" prazo_fim="01/07/2011" ordem="arvore">
                <repeticao quantidade="10">
                    <upi titulo="sim" editor="nao" textarea="sim" autorelacionamento="sim" />
                </repeticao>
            </secao>
        </secao>
    </repeticao>
</debateteses>
```

Figura 6.31 - Configurações estruturais de um debate de teses exemplo

Já a Figura 6.32, esboça a configuração global do Debate de Teses exemplo. No *hot-spot* de publicação, pode por exemplo, disparar emails para os usuários do Debate de Teses sempre que uma tese for publicada. No hot-spot de UsuarioVcom pode, por exemplo, implementar uma rotina para avisar o determinado usuário assim que ele for adicionado ao Debate de Teses, por parte do administrador/professor.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<vcomconfig>
    <titulo>Meu debate de teses</titulo>
    <css>debatedeteses.css</css>
    <mensagemAutoria>[%s] %s disse: </mensagemAutoria>
    <formatoData>EEE, dd/MM/yyyy, HH:mm:ss</formatoData>
    <privado>nao</privado>
    <permiteanonimo>nao</permiteanonimo>
    <permiteadesao>nao</permiteadesao>
    <ativado>sim</ativado>
    <atualizacao>-1</atualizacao>
    <hotspotusuariovcom>exemploshotspots.HotSpotUsuarioVcom</hotspotusuariovcom>
    <hotspotpermissao>exemploshotspots.HotSpotPermissao</hotspotpermissao>
    <hotspotpublicacao>exemploshotspots.HotSpotPublicacao</hotspotpublicacao>
</vcomconfig>
```

Figura 6.32 - Configurações globais de um debate de teses exemplo

6.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Esse capítulo apresentou uma abordagem prática do trabalho proposto. O objetivo foi materializar a proposta deste trabalho e evidenciar que através de um protótipo baseado na proposta FrameColab a possibilidade elaborar propostas colaborativas sem grandes

esforços de programação e infra-estrutura geral. Também mostrou um esboço de propostas diferenciadas das convencionais, não contempladas por ambientes tradicionais, vide Júri Simulado, Jigsaw e Debate de Teses (mesmo que descritos simploriamente).

Obviamente os ambientes retratados foram abordados com uma visão simplificada, no entanto, fica claro o potencial da abordagem descrita neste trabalho.

Por fim, esse capítulo retrata que o estudo descrito em capítulos anteriores foi de grande importância para moldar a elaboração da proposta, deixando-a com aspectos altos de flexibilidade e, também, escalabilidade.

CAPÍTULO 7 CONCLUSÕES E TRABALHOS FUTUROS

7.1 CONCLUSÕES E CONTRIBUIÇÕES DA DISSERTAÇÃO

O surgimento do projeto MOrFEu nasceu através de longas experiência em ambientes virtuais e foi inicialmente retratado em (Menezes, Nevado, Castro Junior, & Santos, 2008). Tal projeto teve forte influência no desenvolvimento deste trabalho, uma vez que propõem uma maneira diferenciada para se moldar ambientes colaborativos.

Para este material ser construído houve um estudo aprofundado em diversas áreas, sendo uma etapa de grande aquisição de conhecimento. Em caráter de exemplo, algumas áreas estudadas: padrões de software, *frameworks*, metodologias de desenvolvimento de *frameworks*, arquitetura de software, sistemas colaborativos, desenvolvimento de ambientes colaborativos, análise e projeto de sistemas, dentre várias outras.

O trabalho aqui exposto propôs um conjunto de artefatos de software que auxiliem desenvolvedores no projeto de ambientes virtuais inovadores. E para isso, foi acoplado os conceitos pressupostos pelo MOrFEu.

O FrameColab fornece mecanismos importantes aos desenvolvedores para a criação de novos Ambientes Colaborativos: (i) possibilidade de descrever estruturalmente Vcoms através de uma linguagem, também proposta neste trabalho, baseada em XML; (ii) possibilidade de acoplar implementações em pontos específicos (*hot-spots*); (iii) definição de um padrão arquitetural interno para o desenvolvimento de aplicações colaborativas; e (iv) acoplamento dos principais conceitos presentes no projeto MOrFEu, que representa uma forma inovadora, conforme foi descrito.

Pode-se citar como contribuições deste trabalho:

- **Proposta do *framework*:** a proposta principal deste trabalho é um *framework*, o FrameColab, que por si só já é uma grande contribuição;

- **Linguagem para descrição estrutural de Vcoms:** criação de uma linguagem, embora simplificada, para descrever estruturalmente ambientes de colaboração;

- **Definição de uma arquitetura interna de aplicações:** definição de uma arquitetura geral interna para aplicações projetadas a partir do FrameColab, fazendo uso de boas práticas da Engenharia de Software;

- **Arcabouço teórico:** fornecimento de um arcabouço teórico para evolução da proposta deste trabalho ou o surgimento de novos trabalhos;

- **Elucidação das limitações nos ambientes tradicionais:** retratação das limitações presentes nas propostas convencionais;

- **Protótipo funcional baseado no FrameColab:** construção de um protótipo funcional baseado na proposta deste trabalho;

- **Aplicações do protótipo:** criação de ambientes colaborativos e algumas Arquiteturas Pedagógicas através do protótipo desenvolvido, evidenciando a viabilidade da proposta.

7.2 TRABALHOS FUTUROS

Os trabalhos futuros, frutos deste trabalho, são inúmeros. Abaixo alguns de maior relevância:

- Formalização dos conceitos do projeto MOrFEu através de uma abordagem ontológica;

- Expandir a linguagem de descrição estrutural de Vcom, proposta neste trabalho, incorporando mais recursos, como forma de moldas ambientes mais complexos;

- Aumentar o número de *hot-spots* disponíveis no FrameColab;

- Acoplar mecanismos que ofereçam suporte a *workflow* ao FrameColab, como foram de facilitar na descrição de atividades e seus relacionamentos de dependências;

- Disponibilizar o *framework* para desenvolvedores testarem e relatarem suas constatações, críticas e sugestões;

- Disponibilizar os ambientes originados a partir do FrameColab para usuários testarem;

- Criação de uma interface Web que, com o uso dos artefatos provindos pelo FrameColab, permita criação de Vcoms pela online;

- Construção de um “Editor de Vcom” e acoplá-lo ao FrameColab, como forma de ajudar a descrição de Vcoms. Este editor pode, por exemplo, exportar a descrição estrutural do Vcom de acordo com a linguagem de descrição proposta neste trabalho e, esta ser utilizada pelo FrameColab para geração do dado Vcom;

- Construção de um “Editor de *Template*” e acoplá-lo ao FrameColab, uma vez que este já prevê tal fato;

REFERÊNCIAS

Almeida, J. P. (2006). Requirements Traceability and Transformation Conformance in Model-Driven Development. *Enterprise Distributed Object Computing Conference, 2006. EDOC '06. 10th IEEE International* , pp. 355-366.

Almeida, J. P., Dijkman, R., Pires, L. F., Quartel, D., & Sinderen, M. V. (2006). Model-driven design, refinement and transformation of abstract interactions. *International Journal of Cooperative Information Systems (IJCIS), Vol.15,. n. 4* , pp. 599-632.

Appleton, B. (1997). *Patterns and Software: Essential Concepts and Terminology*. Retrieved Maio 2010, from http://www.cmcrossroads.com/bradapp/docs/patterns-intro.html

Barros, L. A. (1994). Suporte a Ambientes Distribuídos para Aprendizagem Cooperativa. *Tese de Doutorado, COPPE/UFRJ, .* Rio de Janeiro, Brasil.

Beltrame, W., Monteiro, E. R., Rangel, V. G., & Cury, D. (2008). Multi-Organizador Flexível de Espaços Virtuais. *XIX SBIE.* Fortaleza/CE.

Booch, G., Rumbaugh, J., & Jacobson, I. (2005). *UML Guia do Usuário.* Rio de Janeiro: Elsevier.

Bosch, J., Molin, P., Mattsson, M., Bengtsson, P., & Fayad, M. E. (1999). Framework problem and experiences. In M. E. Fayad, R. E. Johnson, & D. E. Schmidt, *Building Application Frameworks: Object-Oriented Foundations of Framework Design* (pp. 55-82). John Willey and Sons.

Brna, P. (1998). Modelos de colaboração. *Revista Brasileira de Informática e Educação, 3* , 1-15.

Brooke, J. (1993). User interface for CSCW systems. In D. Dapier, & C. Sanger, *CSCW in pratice: Introduction and cases studies.* Springer-Verlag.

Buschmann, F. (1995). Pattern languages of program design. *Proceedings of the First Conference on Pattern Languages and Programming, ACM Press/Addison-Wesley Publishing Co* , pp. 133-142.

Buschmann, F. (1996). *System of Patterns.* Wiley.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). *Pattern-Oriented Software Architecture Volume 1: A System of Patterns.* Wiley.

Carvalho, M. J., Nevado, R. A., & Menezes, C. S. (2005). Arquiteturas pedagógicas para educação à distância: concepções e suporte telemático. *Anais – XVI Simpósio Brasileiro de Informática na Educação, 1* , 362-372.

Casais, E. (1996). An experiment in framework development. *Theory and Practice of Object Systems, Volume 1, Issue 4* , pp. 269-280.

Coad, P. (1992). Object-oriented patterns. *ACM, Volume 35, Issue 9* , pp. 152 -159.

Coplien, J. O., & Schmidt, D. C. (1995). *Pattern Languages of Program Design.* Addison-Wesley Professional.

Daft, R. L., & Lengel, R. H. (1986). Organizational information requirements, media richness and structural design. *Organizational Science* , 554-571.

Delvin, K., & Rosemberg, D. (1996). Language at Work: analyzing communication. *CSLI lecture notes no. 66* .

Dijkstra, E. W. (1976). *A Discipline of Programming.* Prentice Hall.

Dillenbourg, P. (1999). What do you mean by collaborative learning? *Collaborative-learning: Cognitive and Computational Approaches, Oxford, Elsevier* , pp. 1-19.

Dillenbourg, P., & Self, J. A. (1992). A computational approach to socially distributed cognition. *European Journal of Psychology of Education, Vol VII, No 4* , 252-273.

Dodge, B. (2004). Retrieved from The WebQuest Design Patterns: http://webquest.sdsu.edu/designpatterns/all.htm

Dougiamas, M. (2009). Retrieved from Moodle – a free, open source course management system for online learning: http://moodle.org/

Dourish, P., & Belloti, V. (1992). Awareness and coordination in shared workspaces. *Proceedings of Computer Supported Cooperative Work 1992, Toronto, Ontario, ACM Press, USA*, (pp. 107-114). Toronto.

Ellis, C. A., Gibbs, S. J., & Rein, G. L. (1991). Groupware - Some Issues and Experiences. *Communications of the ACM, Vol. 34, No. 1* , pp. 38-58.

Fagundes, L. C., Nevado, R. A., Basso, M., Bitencourt, J., Menezes, C. S., & Monteiro, V. C. (2006). Projetos de Aprendizagem – Uma experiência mediada por ambientes Telemáticos. *Revista Brasileira de Informática na Educação.*

Fayad, M. E., & Schmidt, D. C. (1997). Object-oriented application frameworks. *Communications of the ACM, Volume 40, Issue 10* , pp. 32-38.

Fayad, M. E., Johnson, R. E., & Schmidt, D. C. (1999). *Building Application Frameworks: Object-Oriented Foundations of Framework Design.* John Wiley & Sons.

Fioravanti, M. L., Nakagawa, E. Y., & Barbosa, E. F. (2010). EDUCAR: Uma Arquitetura de Referência para Ambientes Educacionais. *XXI Simpósio brasileiro de Informática na Educação (SBIE)* . João Pessoa, Pernambuco, Brasil.

foldoc.org. (1995). *Application Program Interface*. Retrieved abril 2011, from FOLDOC - Free On-Line Dictionary Of Computing: http://foldoc.org/Application+Program+Interface

Froehlich, G., Hoover, H. J., Liu, L., & Sorenson, P. G. (1997). Hooking into object-oriented application frameworks. *International Conference on Software Engineering, Proceedings of the 19th international conference on Software engineering, ACM* , pp. 491-501.

Froehlich, G., Hoover, H. J., Liu, L., & Sorenson, P. G. (1997). Reusing Application Frameworks Through Hooks Reusing Application Frameworks Through. *Communications of the ACM special issue on object-oriented frameworks* .

Fuks, H., Raposo, A. B., & Gerosa, M. A. (2003). Do Modelo de Colaboração 3C à Engenharia de Groupware. *Simpósio Brasileiro de Sistemas Multimídia e Web – Webmidia, Trilha especial de Trabalho Cooperativo Assistido por Computador* .

Fuks, H., Raposo, A. B., & Gerosa, M. A. (2002). Engenharia de Groupware: Desenvolvimento de Aplicações Colaborativas. *XXI Jornada de Atualização em Informática, Anais do XXII Congresso da Sociedade Brasileira de Computação, V2, Cap. 3, ISBN 85-88442-24-8* , pp. 89-128.

Fuks, H., Raposo, A. B., Gerosa, M. A., & Lucena, C. J. (2005). Applying the 3c-model to groupware engineering. *International Journal of Cooperative Information Systems (IJCIS), v. 14, n. 2-3* , pp. 299–328.

Fussell, S. R., Kraut, R. E., Learch, F. J., Scherlis, W. L., McNally, M. M., & Cadiz, J. J. (1998). Coordination, overload and team performance: effects of team communication strategies. *Proceedings of CSCW '98, Seattle, USA, ISBN 1-58113-009-0* , pp. 275-284.

Gall, H. C., Klösch, R. R., & Mittermeir, R. T. (1996). Application Patterns in Re-Engineering: Identifying and Using Reusable Concepts. *Proceedings of 6th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems* , pp. 1099-1106.

Gamma, E., Helm, R., Johnson, R. E., & Vlissides, J. M. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley Professional.

Gerosa, M. A., Raposo, A. B., Fuks, H., & Lucena, C. J. (2006). Component-Based Groupware Development Based on the 3C Collaboration Model. *Anais do XX Simpósio Brasileiro de Engenharia de Software – SBES, ISBN 85-7669-079-9* , pp. 129-144.

González, L. A. (2005). Um modelo conceitual para aprendizagem colaborativa baseada na execução de projetos pela web. *Tese de doutorado em Engenharia, USP* . São Paulo.

Grosz, B. J. (1996). Collaborative systems. *AI Magazine 17 (2) - Groupware Patterns Swiki (2005) http://www.groupware-patterns.org* , 67–85.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies, Volume 43, Issue 5-6* , pp. 907-928.

Guarino, N. (1998). Formal Ontology in Information Systems. *Proceedings of the 1st International Conference June 6-8, 1998, Trento, Italy, 1st edition* , p. 337.

Guizzardi, G. (2000). *Desenvolvimento para e com reuso: um estudo de caso no domínio de vídeo sob demanda, Dissertação de mestrado, Programa de Mestrado em Informática do Centro Tecnológico da Universidade Federal do Espírito Santo, Universidade Federal do Espírito Santo.*

Gutwin, C., & Greenberg, S. (1999). *A framework of awareness for small groups in shared-workspace groupware.* Saskatchewan University, Canada: Technical Report 99-1.

Hammer, M. (1990). Reengineering Work: Don't Automate, Obliterate. Harvard Business Review.

International Organization for Standard ISO. (1986). *Information processing - Text and office systems - Standard Generalized Markup Language (SGML)*. Retrieved from International Organization for Standard ISO: http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=16387

Jacobsen, E. E., Kristensen, B. B., & Nowack, P. (1997). Patterns in the Analysis, Design and Implementation of Frameworks. *Proceedings of the Twenty-First Annual International Computer Software and Application Conference, (COMPSAC'97)* .

Jacobson, I., Booch, G., & Rumbaugh, J. (1999). *The Unified Software Development Process.* Addison-Wesley Professional.

Johnson, R. E. (1993). *How to Design Frameworks*. Retrieved abril 2010, from Notes for CS497REJ: Object-Oriented Programming and Design: How to Design Frameworks

Johnson, R. E., & Foote, B. (1988). Designing reusable classes. *Journal of Object Oriented Programming, v. 1, n. 2* , 22-35.

Johnson, R. E., & Foote, B. (1988). Designing Reusable Classes. *Journal of Object Oriented Programming, v. 1, n. 2* , pp. 22–35.

Johnson, R. E., & Russo, V. F. (1991). *Reusing Object-Oriented Designs.* Rel. Téc. UIUCDCS 91-1696, University of Illinois.

Kruchten, P. (1998). Modeling Component Systems with the Unified Modeling Language. *International Component-Based Software Engineering - Workshop* .

Lee, T. B. (1999). *Weaving the Web: the original design of the World Wide Web by its inventor.* Harper Business.

Lévy, P. (2000). A emergência dos cyberspace e as mutações culturais. In N. M. Pellanda, & E. Campos, *Ciberespaço um hipertexto.* Porto Alegre: Artes e Ofícios.

Lyytinen, K. J., & Ngwenyama, O. K. (1992). What does computer support for cooperative work mean? A structurational analysis of computer supported cooperative work. *Accounting, Management and Information Technologies, Vol. 2, N. 1* , pp. 19-37.

Malone, T. W., & Crowston, K. (1990). What Is Coordination Theory and How Can It Help Design Cooperative Work Systems? *Proceedings of the Conference on Computer-Supported Cooperative Work, ISBN 0-89791-402-3,* (pp. 357-370). Los Angeles, USA.

Martin, R. C., Riehle, D., & Buschmann, F. (1998). *Pattern Languages of Program Design 3.* Addison-Wesley Professional.

Mattsson, M. (2000). Evolution and Composition Object-Oriented Frameworks. *PhD Thesis, University of Karlskrona/Ronneby, Department of Software Engineering and Computer Science* .

Mattsson, M. (1996). *Object-Oriented Frameworks - A survey of methodological issues.* Licentiate Thesis, Department of Computer Science, Lund University, CODEN: LUTEDX/(TECS-3066)/1-130/(1996), also as Technical Report, LU-CS-TR: 96-167, Department of Computer Science, Lund University.

Mattsson, M., Bosch, J., & Fayad, M. E. (2000). Framework integration problems, causes, solutions. *Communications of the ACM, Volume 42, Issue 10* , pp. 80-87.

Menezes, C. S., Nevado, R. A., Castro Junior, A. N., & Santos, L. N. (2008). MOrFEU Multi-Organizador Flexível de Espaços VirtUais para Apoiar a Inovação Pedagógica em EAD. *XIX SBIE*. Fortaleza/CE.

Monteiro, V. C., Menezes, C. S., Nevado, R. A., & Fagundes, L. C. (2005). Ferramenta de Autoria e Interação para apoio ao desenvolvimento de Projetos de Aprendizagem. *Renote Revista Novas Tecnologias na Educação V3, v. 3, n. 2* .

Mowbray, T. J., & Malveau, R. C. (1997). *Corba Design Patterns.* Chichester, England: John Wiley & Sons.

Natalli, E. L., & Menezes, C. S. (2010). Um framework para construção de ambientes colaborativos para mediação da Aprendizagem. *Congresso Iberoamericano de Informática Educativa.* Santiago/Chile.

Nevado, R. A., Carvalho, M. J., & Menezes, C. S. (2007). *Aprendizagem em rede na educação a distância: estudos e recursos para formação de professores.* Porto Alegre: Ricardo Luz.

Nevado, R. A., Dalpiaz, M. M., & Menezes, C. S. (2009). Arquitetura Pedagógica para Construção Colaborativa de Conceituações. *Workshop Sobre Educação na Escola, XV WIE.* Bento Gonçalves/RS.

Oliveira, F. F., Antunes, J. C., & Guizzardi, R. S. (2007). Towards a collaboration ontology. *In: GUIZZARDI, G.; FARIAS, C. (Ed.). Proceedings of the 2nd Workshop on Ontologies and Metamodels in Software and Data Engineering (WOMSDE'07)* . João Pessoa, Brazil.

OMG. (2003). *MDA Guide Version 1.0.1*. Retrieved 2010 Maio, from OMG: http://www.omg.org/docs/omg/03-06-01.pdf

Peirce, C. S. (1977). *Semiótica.* São Paulo: Editora Perspectiva.

Pereira, V. L., Castro Júnior, A. N., Souza, F. F., Mendonça, A. P., & Silva, L. S. (2002). Análise do método Jigsaw de aprendizagem cooperativa através da utilização de mapas conceituais. *XXII Congresso da Sociedade Brasileira de Computação - VIII Workshop de Informática na Escola - XXII Congresso da SBC*, (pp. 181-188). Florianópolis-SC.

Pessoa, J. M., & Menezes, C. S. (2003). Um Framework para Construção Cooperativa de Ambientes Virtuais de Aprendizagem na Web. *Simpósio Brasileiro de Informática na Educação* .

Pimentel, M., Gerosa, M. A., Filippo, D., Barreto, C. G., Raposo, A. B., Fuks, H., et al. (2005). AulaNet 3.0: Desenvolvendo Aplicações Colaborativas Baseadas em Componentes 3C. *XVI Simpósio Brasileiro de Informática na Educação, SBIE*, (pp. 761-770). Juiz de Fora-MG.

Pree, W. (1994). *Design Patterns for Object-Oriented Software Development.* Addison Wesley Longman.

Pree, W., & Koskimies, K. (2000). Framelets—small and loosely coupled frameworks. *ACM Computing Surveys (CSUR), Volume 32, Issue 1es* , pp. 6-10.

Prieto-Diaz, R., & Arango, G. (1991). *Domain Analysis and Software Systems Modeling.* IEEE Computer Society.

Rangel, V. G., Beltrame, W. A., Cury, D., & Menezes, C. S. (2009). MOrFEu: Towards the Design of an Environment for Flexible Virtual Spaces Organization. *WCCE – World Conference on Computer in Education.* Bento Gonçalvez/RS.

Raposo, A. B., Magalhães, L. P., Ricarte, I. L., & Fuks, H. (2001). Coordination of collaborative activities: A framework for the definition of tasks interdependencies. *Proceedings of the 7th International Workshop on Groupware - CRIWG, Darmstadt, Germany, IEEE Computer Society, USA, ISBN 0-7695-1351-4* , 170-179.

Real, L. M., & Menezes, C. S. (2008). Júri simulado: possibilidade de construção de conhecimento a partir de interações em um grupo. In R. A. Nevado, M. J. Carvalho, & C. S. Menezes, *Aprendizagem em rede na educação a distancia* (pp. 93-102). Porto Alegre: Ricardo Lenz.

Roschelle, J., & Teasley, S. (1995). The construction of shared knowledge in collaborative problem solving. *O'Malley, C.E., (ed.), Computer Supported Collaborative learning* . Springer-Verlag/Heildelberg, Germany.

Sametinger, J. (2001). *Software Engineering with Reusable Components.* Springer.

Schmidt, D. C., & Stephenson, P. (1995). Experience Using Design Patterns to Evolve Communication Software Across Diverse OS Platforms. *Proceddings of the 9th European Conference on Object-Oriented Programming* .

Schmidt, K. (1991). Riding a Tiger, or Computer Supported Cooperative Work. *Proc. of the 2nd European Conf. on Computer-Supported Cooperative Work (ECSCW),* (pp. 1-16).

Sharan, S. (1999). *Handbook of cooperative learning methods.* Praeger Publishers.

Silva, R. P. (2000). Suporte ao desenvolvimento e uso de frameworks e componentes. *tese de doutorado, Instituto de Informática-Programa de Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul* .

Slavin, R. E. (1995). *Cooperative learning: theory, research, and practice.* Allyn & Bacon.

Sparks, S., Benner, K. M., & Faris, C. (1996). Managing Object-Oriented Framework Reuse. *IEEE Computer Society Press, Volume 29, Issue 9* , pp. 52-61.

Sun. (2007). Retrieved from Core J2EE patterns: data access object: http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html

Taligent. (1995). *Leveraging Object-Oriented.* Retrieved Abril 2010, from LHCb Computing Home Page: https://lhcb-comp.web.cern.ch/lhcb-comp/Components/postscript/leveragingoo.pdf

Tapscoot, D., Ticoll, D., & Lowy, A. (2000). *Digital Capital: Harnessing the Power of Business Webs.* USA: Harvard Business Press.

Vieira Júnior, R. R., Santos, O. L., Rafalski, J. P., Bada, E. M., Silva, H. F., & Menezes, C. S. (2011). Coordenação nas Atividades Colaborativas em Ambientes de Aprendizagem - Uma Avaliação na Implementação de Arquiteturas Pedagógicas. *XVII Ciclo de Palestras sobre Novas Tecnologias na Educação* .

Vlissides, J. M., Coplien, J. O., & Kerth, N. L. (1996). *Pattern Languages of Program Design 2.* Addison-Wesley Professional.

Watson, W. R., & Watson, S. L. (2007). *An Argument for Clarity: What are Learning Management Systems, What are They Not, and What Should They Become?* echTrends. Springer Boston, Volume 51, Number 2. ISSN 8756-3894 (Print) 1559-7075 (Online).

Weinand, A., Gamma, E., & Marty, R. (1989). Design and Implementation of ET++, a Seamless Object-Oriented Application Framework. *Structured Programming, Vol.10, No2* .

Werner, C. M., & Braga, R. M. (2000). Desenvolvimento baseado em componentes. *Anais do XIV Simpósio Brasileiro de Engenharia de Software - Minicurso* , pp. 297–329.

Winograd, T. (1988). A Language/Action Perspective on the Design of Cooperative Work. In I. Greif, *Computer Supported Cooperative Work - A book of readings.* USA: Morgan Kaufmann Publishers, ISBN 0-934613-57-5.

Winograd, T., & Flores, F. (1987). *Understanding Computers and Cognition.* USA: Addison-Wesley, ISBN 0-201-11297-3.

Wirfs-Brock, A., Vlissides, J. M., Cunningham, W., Johnson, R. E., & Bollette, L. (1990). Designing reusable designs (panel session): experiences designing object-oriented frameworks. *Conference on Object Oriented Programming Systems Languages and Applications; Proceedings of the European conference on Object-oriented programming addendum: systems, languages, and applications: systems, languages, and applications, ACM* , pp. 19-24.

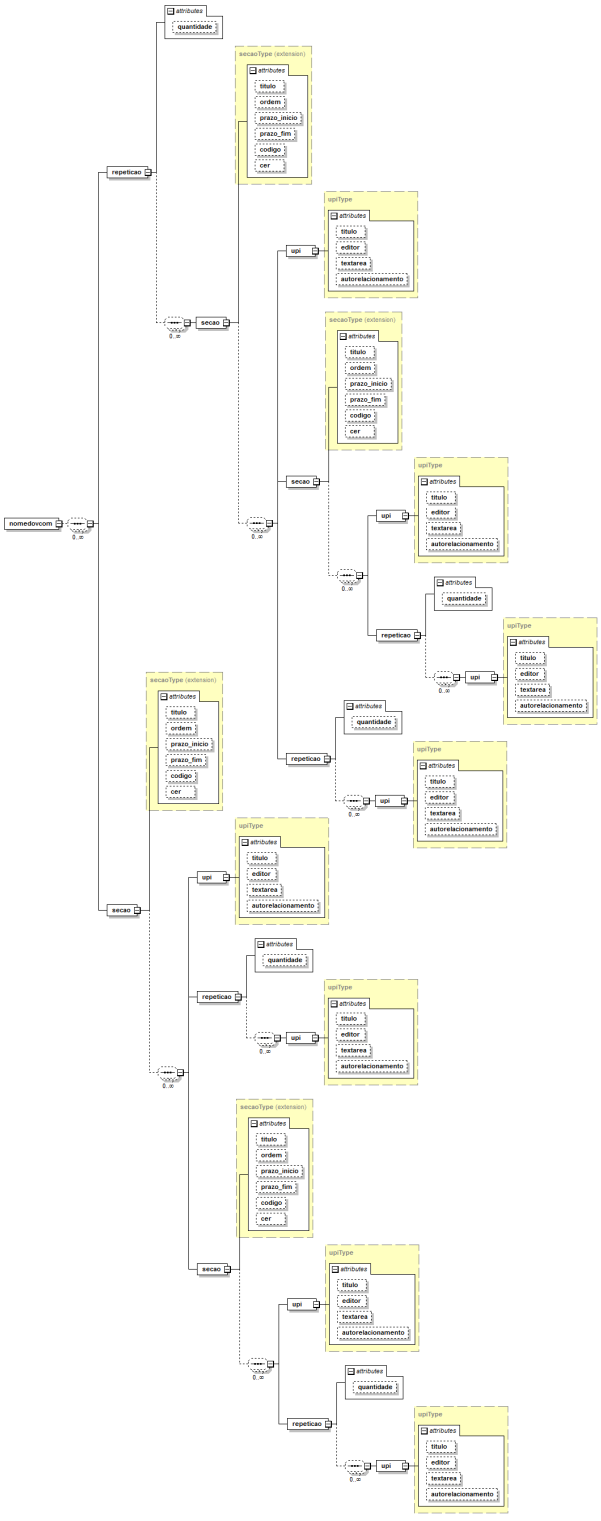
Wirfs-Brock, R. J., & Johnson, R. E. (1990). Surveying current research in object-oriented design. *Communications of the ACM, Volume 33, Issue 9* , pp. 104-124.

Wooldridge, M. (1999). Intelligent Agents. In G. Weiss, *Multiagent Systems.* The MIT Press.

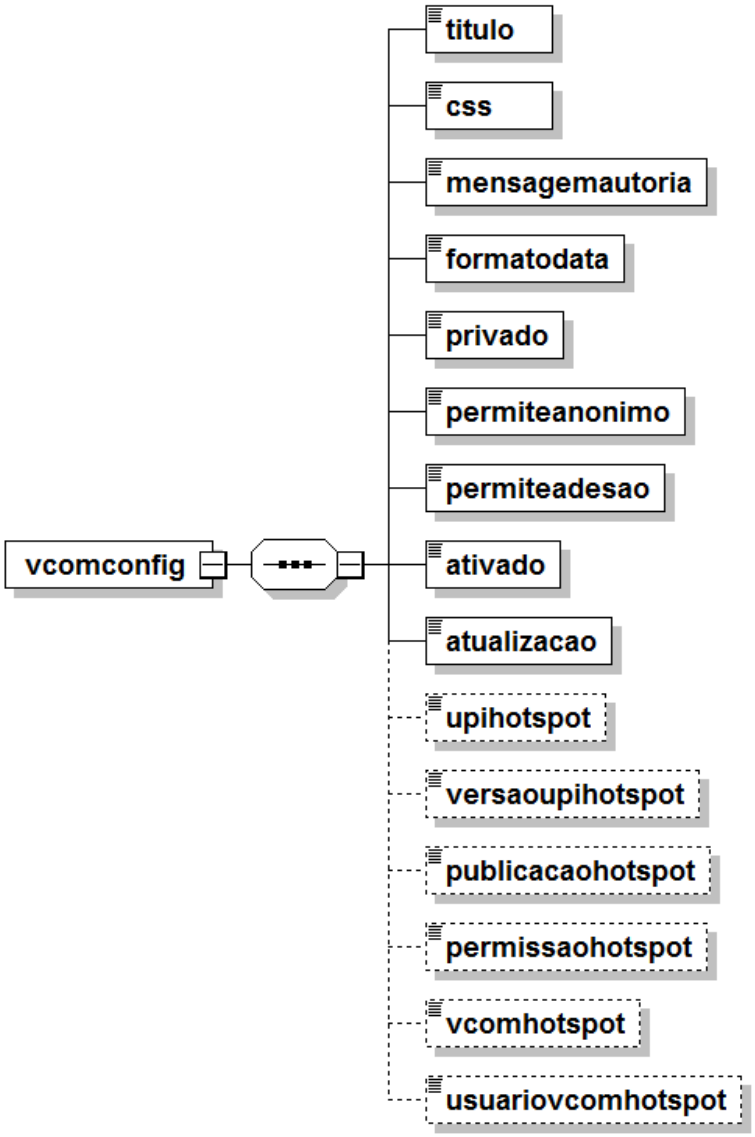
Woolf, H. B. (1981). *Webster's New Collegiate Dictionary.* Springfield. Mass: G&C.

Yassin, A. H., & Fayad, M. E. (1999). Application Frameworks: A Survey, cap. 29. In M. E. Fayad, & R. E. Johnson, *Domain-Specific Application Frameworks: Frameworks Experience by Industry* (pp. 615-632). John Wiley & Sons.

APÊNDICE I: REPRESENTAÇÃO DO XMLSCHEMA DA LINGUAGEM DE DESCRIÇÃO SEGUNDO O PADRÃO DA W3C



APÊNDICE II: REPRESENTAÇÃO DO XMLSCHEMA DO XML DE CONFIGURAÇÃO DO VCOM SEGUNDO O PADRÃO DA W3C



APÊNDICE III: XMLSCHEMA DA LINGUAGEM DE DESCRIÇÃO UTILIZADA E DAS CONFIGURAÇÕES DO VCOM

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://www.w3schools.com"
targetNamespace="http://www.w3schools.com" elementFormDefault="qualified">
  <xs:element name="nomedovcom">
    <xs:complexType>
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:element name="repeticao">
          <xs:complexType>
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
              <xs:element name="secao">
                <xs:complexType>
                  <xs:complexContent>
                    <xs:extension base="secaoType">
                      <xs:sequence minOccurs="0" maxOccurs="unbounded">
                        <xs:element name="upi" type="upiType"/>
                        <xs:element name="secao">
                          <xs:complexType>
                            <xs:complexContent>
                              <xs:extension base="secaoType">
                                <xs:sequence minOccurs="0" maxOccurs="unbounded">
                                  <xs:element name="upi" type="upiType"/>
                                  <xs:element name="repeticao">
                                    <xs:complexType>
                                      <xs:sequence minOccurs="0" maxOccurs="unbounded">
                                        <xs:element name="upi" type="upiType"/>
                                      </xs:sequence>
                                      <xs:attribute name="quantidade" type="xs:positiveInteger"/>
                                    </xs:complexType>
                                  </xs:element>
                                </xs:sequence>
                              </xs:extension>
                            </xs:complexContent>
                          </xs:complexType>
                        </xs:element>
                        <xs:element name="repeticao">
                          <xs:complexType>
                            <xs:sequence minOccurs="0" maxOccurs="unbounded">
                              <xs:element name="upi" type="upiType"/>
                            </xs:sequence>
                            <xs:attribute name="quantidade" type="xs:positiveInteger"/>
                          </xs:complexType>
                        </xs:element>
                      </xs:sequence>
                    </xs:extension>
                  </xs:complexContent>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="quantidade" type="xs:positiveInteger"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="secao">
          <xs:complexType>
            <xs:complexContent>
              <xs:extension base="secaoType">
                <xs:sequence minOccurs="0" maxOccurs="unbounded">
                  <xs:element name="upi" type="upiType"/>
                  <xs:element name="repeticao">
                    <xs:complexType>
                      <xs:sequence minOccurs="0" maxOccurs="unbounded">
                        <xs:element name="upi" type="upiType"/>
```

```
                        </xs:sequence>
                        <xs:attribute name="quantidade" type="xs:positiveInteger"/>
                     </xs:complexType>
                  </xs:element>
                  <xs:element name="secao">
                     <xs:complexType>
                        <xs:complexContent>
                           <xs:extension base="secaoType">
                              <xs:sequence minOccurs="0" maxOccurs="unbounded">
                                 <xs:element name="upi" type="upiType"/>
                                 <xs:element name="repeticao">
                                    <xs:complexType>
                                       <xs:sequence minOccurs="0" maxOccurs="unbounded">
                                          <xs:element name="upi" type="upiType"/>
                                       </xs:sequence>
                                       <xs:attribute name="quantidade" type="xs:positiveInteger"/>
                                    </xs:complexType>
                                 </xs:element>
                              </xs:sequence>
                           </xs:extension>
                        </xs:complexContent>
                     </xs:complexType>
                  </xs:element>
               </xs:sequence>
            </xs:extension>
         </xs:complexContent>
      </xs:complexType>
   </xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:complexType name="secaoType">
   <xs:attribute name="titulo" type="xs:string"/>
   <xs:attribute name="ordem">
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="data_crescente"/>
            <xs:enumeration value="data_decrescente"/>
            <xs:enumeration value="titulo_crescente"/>
            <xs:enumeration value="titulo_decrescente"/>
            <xs:enumeration value="arvore"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:attribute>
   <xs:attribute name="prazo_inicio" type="xs:date"/>
   <xs:attribute name="prazo_fim" type="xs:date"/>
   <xs:attribute name="codigo" type="xs:integer"/>
   <xs:attribute name="cer">
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:pattern value="sim|nao"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:attribute>
</xs:complexType>
<xs:complexType name="upiType">
   <xs:attribute name="titulo">
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:pattern value="sim|nao"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:attribute>
   <xs:attribute name="editor">
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:pattern value="sim|nao"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:attribute>
```

```
      <xs:attribute name="textarea">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:pattern value="sim|nao"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="autorelacionamento">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:pattern value="sim|nao"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
</xs:schema>
```

Figura 0.1 Esquema da linguagem de descrição de Vcom

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://www.w3schools.com"
targetNamespace="http://www.w3schools.com" elementFormDefault="qualified">
    <xs:element name="vcomconfig">
        <xs:complexType>
            <xs:sequence minOccurs="1" maxOccurs="1">
                <xs:element name="titulo" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="css" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="mensagemautoria" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="formatodata" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="privado" minOccurs="1" maxOccurs="1">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:pattern value="sim|nao"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="permiteanonimo" minOccurs="1" maxOccurs="1">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:pattern value="sim|nao"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="permiteadesao" minOccurs="1" maxOccurs="1">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:pattern value="sim|nao"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="ativado" minOccurs="1" maxOccurs="1">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:pattern value="sim|nao"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="atualizacao" minOccurs="1" maxOccurs="1">
                    <xs:simpleType>
                        <xs:restriction base="xs:integer">
                            <xs:minInclusive value="-1"/>
                            <xs:maxInclusive value="300"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:element>
                <xs:element name="upihotspot" type="xs:string" minOccurs="0" maxOccurs="1"/>
                <xs:element name="versaoupihotspot" type="xs:string" minOccurs="0" maxOccurs="1"/>
                <xs:element name="publicacaohotspot" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

```
            <xs:element name="permissaohotspot" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="vcomhotspot" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="usuariovcomhotspot" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figura 0.2 - Esquema de configuração de Vcom

APÊNDICE IV: DICIONÁRIO DE DADOS DO DIAGRAMA DE CLASSES

Tabela 10 - Dicionário de dados completo do diagrama de classes

Pacote	**Classes**			
Controle	**Perfil**	**Atributo**	**Descrição**	**Tipo**
	Perfil do usuário. Cada Vcom possui um conjunto de perfis, que contém as permissões de suas funcionalidades.	nome	nome do perfil	String
		descricao	descrição do perfil	String
		criacao	data de criação do perfil	Date
		alteracao	data de alteração do perfil	Date
	Permissao	**Atributo**	**Descrição**	**Tipo**
	Acesso propriamente dito de uma funcionalidade a um perfil de um dado Vcom.	acesso	Código do acesso (controlado e interpretado internamente pelo FrameColab)	int
Cadastro	**Amigos**	**Atributo**	**Descrição**	**Tipo**
	Amigos dos usuários	descricao	descrição da amizade	String
		aprovacao	data de aprovação da amizade	Date
	AreaInteresse	**Atributo**	**Descrição**	**Tipo**
	Áreas de interesse do usuário e do Vcom	nome	nome da área de interesse	String
		descricao	descrição da área de interesse	String
	Escolaridade	**Atributo**	**Descrição**	**Tipo**
	Grau de escolaridade do usuário	nome	nome do grau de escolaridade	String
	Usuario	**Atributo**	**Descrição**	**Tipo**
	Usuário do ambiente	nome	nome do usuário	String
		sobrenome	sobrenome do usuário	String
		apelido	apelido do usuário	String
		email	email do usuário	String
		login	login do usuário	String
		senha	senha do usuário	String
		cpf	CPF do usuário	String

		nascimento	data de dascimento do usuário	Date
		endereco	endereço do usuário	String
		telefone	telefone do usuário	int
		logout	data que o usuário fez logout	Date
		criacao	data de criação do usuário	Date
		alteracao	data de alteração do usuário	Date
	UsuarioVcom	**Atributo**	**Descrição**	**Tipo**
	Indicações de acesso de usuários a Vcoms	aprovacao	data de aprovação do usuário ao Vcom	Date
Produção	**Publicacao**	**Atributo**	**Descrição**	**Tipo**
	Publicação de uma UPI, atralada ao Vcom	publicacao	data de publicação da UPI ao Vcom	Date
	Upi	**Atributo**	**Descrição**	**Tipo**
	UPI, que é o elemento básico de informação	criacao	data de criação da UPI	Date
	VersaoUpi	**Atributo**	**Descrição**	**Tipo**
	Versionamento de uma UPI	titulo	título da UPI	String
		corpo	corpo da UPI	String
		versao	versão da UPI	int
		alteracao	data de alteração da UPI	Date
	Voto	**Atributo**	**Descrição**	**Tipo**
	Possibilidade de UPIs receberem votos	tipo	código do tipo de voto	int
		ip	IP da máquina que fez o voto	String
		data	data do voto	Date
Vcom	**Funcionalidade**	**Atributo**	**Descrição**	**Tipo**
	Funciondade que o Vcom possui	nome	nome da funcionalidade	String
		descricao	descrição da funcionalidade	String
		codigo	código da funcionalidade	int
		criacao	data de criação da funcionalidade	Date

		alteracao	data de alteração da funcionalidade	Date
	Vcom	**Atributo**	**Descrição**	**Tipo**
	Ambiente de colaboração projetado	nome	nome do Vcom	String
		descricao	descrição do Vcom	String
		xml	arquivos xmls utilizados	String
		criacao	data de criação do Vcom	Date
		alteracao	data de alteração do Vcom	Date