UNIVERSIDADE DO EXTREMO SUL CATARINENSE, UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

MAYARA MADEIRA TREVISOL

MODELAGEM UWE (UML-BASED WEB ENGINEERING) DE UM *FRAMEWORK* PARA ELABORAÇÃO DE TESTES DE VERIFICAÇÃO DE APRENDIZAGEM *ON-LINE*.

CRICIÚMA, JULHO DE 2009

MAYARA MADEIRA TREVISOL

MODELAGEM UWE (UML-*BASED WEB ENGINEERING*) DE UM *FRAMEWORK* PARA ELABORAÇÃO DE TESTES DE VERIFICAÇÃO DE APRENDIZAGEM *ON-LINE*.

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do grau de bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense.

Orientador(a): Profª Msc. Leila Lais Gonçalves

CRICIUMA, JULHO DE 2009

MAYARA MADEIRA TREVISOL

Modelagem UWE (UML – Based Web Engineering) de um Framework para Elaboração de Testes de Verificação de Aprendizagem Online

Submetido ao corpo docente do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Prof. MSc. Rogério Antônio Casagrande
Coordenador Adjunto do Curso de Ciência da Computação

Banca Examinadora:

Profa. MSc. Leila Laís Gonçalves (UNESC)
Orientador

Prof. MSc. Gustavo Bisognin (UNESC)

Esp. Marcelo Mazon (Depto de TI – UNESC)

A minha família que me apoiou durante esses anos e a todas as pessoas que mesmo com apenas um sorriso me fizeram ir em frente e lutar pelos meus sonhos.

AGRADECIMENTOS

Agradeço a Deus por estar sempre ao meu lado mesmo nas horas que duvidei.

Aos meus pais Márcia e César que sempre colocaram a educação em primeiro lugar e me ensinaram que a honestidade está acima de qualquer coisa.

A minha orientadora Leila e sua família que cedeu manhãs de vários sábados para me orientar e tornar essa pesquisa possível.

Aos meus amigos que estavam sempre ao meu lado dando força e tornando os momentos difíceis mais amenos.

Ao meu namorado Ronan, que mesmo concluindo a graduação, esteve sempre me ajudando a vencer as preocupações e mostrando que juntos a gente chega mais longe.

A professora Jane que nos últimos dias cedeu seu final de semana para corrigir meu trabalho.

Por fim, a todos os professores e funcionários que fizeram parte dessa etapa da minha vida.

Hoje você terá a vitória sobre o que foi ontem; amanhã, triunfará sobre os menos preparados; depois, sobre os mais competentes.

Miyamoto Musashi

RESUMO

Nos últimos anos houve um crescente interesse no emprego de tecnologia de informação e comunicação (TICs) na educação, porém o uso destes recursos como suporte às práticas pedagógicas ainda é limitado. O acompanhamento da aprendizagem, realizado por diferentes instrumentos e métodos, fornece ao professor dados importantes a respeito do processo de ensino e aprendizagem como as dificuldades dos estudantes, deficiências do processo e níveis de desempenho. Um dos fatores importantes nesse processo é a mediação pedagógica, realizada na modalidade de ensino presencial pelo professor com intuito de sanar problemas apontados nos dados de acompanhamento de aprendizagem.

Essa pesquisa buscou contribuir no desenvolvimento de instrumentos de verificação do aprendizado *on-line* com uma abordagem pedagógica que visa dar suporte ao acompanhamento da aprendizagem possibilitando intervenções mediadoras pré-programadas.

Nesse trabalho é proposta a modelagem de um *framework* que oferece flexibilidade para o professor incorporar sua abordagem pedagógica nos Testes de Verificação de Aprendizado (T2A) podendo adicionar mediações e *feedbacks* adequados. Para que a modelagem possuísse um nível adequado de abstração e detalhe foi adotado a notação de modelagem UWE que é uma extensão da UML para aplicações *Web*.

Durante a pesquisa foram levantados os requisitos, definidos pontos de flexibilidade, desenvolvido os diagramas e roteiros que juntos compõem a modelagem do *framework* T2A.

Palavras chave: *Framework*, UWE, EVA, Exercícios *on-line*.

ABSTRACT

In recent years there has been a growing interest in the use of technology information and communication (ICT) in education, but the use of ICTs in most do not take into account the pedagogical practices that are fundamental in the process of teaching learning. Monitoring learning has the function of providing important data to the teacher with the skills, preferences and problems of students, besides give opportunity to the students to learn to overcome their difficulties, improve and reflect on their performance.

This research was based on the lack of instruments of monitoring learning that reflect a pedagogical approach. One of factors as important and as difficult to find the tools mediation is available to educational means that the intervention of teacher who is placed as a facilitator to work for growth of the student.

This research proposes a modeling *framework* that provides flexibility for the teacher incorporate their pedagogical approach in years of auditing and mediation learning can add appropriate *feedback*. For the modeling have an appropriate level of abstraction and detail has adopted the modeling notation that UWE is a extension of UML for *Web* applications During the research were presented requirements, set-points flexibility, developed the *scripts* and diagrams that together make up the modeling *framework* of T2A.

Keywords: *Framework*, UWE, EVA, exercises *on-line.*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

EJB	Enterprise Java Beans
EVA	Exercícios de Verificação de Aprendizagem
EW	Engenharia *Web*
IoC	Inversion of Controll
MVC	Model View Controller
OMG	Object Management Group
OOHDM	Object Oriented Hipermídia Design Method
OOWS	Object Oriented *Web* Solution
ORB	Object Request Broker
T2A	Teste de Avaliação de Aprendizagem
TIC	Tecnologia de Informação e Comunicação
UML	Linguagem de Modelagem Unificada
UNESC	Universidade do Extremo Sul Catarinense
UWE	UML-based *Web* Engineering
WAE	*Web* Application Extension
WebML	*Web* Modeling Language

SUMÁRIO

1 INTRODUÇÃO

A crescente demanda de desenvolvimento de conteúdos e recursos digitais para atender a diferentes práticas pedagógicas motivou essa pesquisa e a direcionou ao contexto *Web*, devido ao grande uso de tecnologia de informação e comunicação (TICs) na educação.

O emprego de TICs apenas não é suficiente no acompanhamento do processo ensino-aprendizagem, dessa forma se faz necessário associar a tecnologia às metodologias pedagógicas, adaptando-as à realidade virtual para se obter sucesso no processo de aprendizagem.

Juntamente a essa necessidade tem-se a grande questão na educação que é o acompanhamento da aprendizagem, cujo papel é o de fornecer ao professor dados importantes a respeito das aptidões, preferências e dificuldades dos estudantes, e ainda, gerar nos estudantes a oportunidade de aprender, de superar as suas dificuldades, melhorar e refletir sobre o seu desempenho.

Esse fator chamou a atenção para a carência de instrumentos de acompanhamento do aprendizado, que refletem uma abordagem pedagógica, e devem ser escolhidos a partir de objetivos de aprendizagem e avaliação definidos pelo professor. Na utilização da maioria desses instrumentos é importante a mediação pedagógica que significa a intervenção do professor que se coloca como um facilitador, incentivador ou motivador da aprendizagem, que colabora para que o estudante chegue aos seus objetivos.

Dessa forma pode-se afirmar que um dos maiores desafios, no que diz respeito ao acompanhamento do aprendizado mediado pelas TICs, é articular as inúmeras vantagens criadas pelas tecnologias modernas e utilizá-las de forma efetiva para apoiar, desde a elaboração de recursos, como também dar suporte nos processos de verificação e *feedback* de resultados.

1.1 OBJETIVO GERAL

Modelar um *framework* para elaboração de exercícios de verificação *on-line* para acompanhamento da aprendizagem com o uso de UWE.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos dessa pesquisa são:

a) compreender *framework*: conceito, tipos e estrutura;
b) abranger UWE: artefatos, aplicação e modelagem de aplicação *Web*;
c) entender as formas de acompanhamento de aprendizagem, seus instrumentos e estratégias de aplicação *on-line*;
d) identificar e descrever os tipos de exercícios de verificação de aprendizagem (EVA) *on-line*;
e) gerar artefatos UML estendidos para *Web* (UWE) que modelem um *framework* para elaboração de EVAs *on-line* especificando sua estrutura, esquema de comunicação, pontos e formas de adaptações, mecanismos de *feedback* e de mediação pedagógica.

1.3 JUSTIFICATIVA

O desenvolvimento de novas tecnologias contribuem para ampliar as possibilidades e aplicações das TICs na Educação. Com o surgimento de novas necessidades, da melhoria e adequação das aplicações já disponíveis, se faz necessário o uso de recursos e soluções mais elaboradas. É importante, nestas soluções, a adequação dos métodos, das

tecnologias e a incorporação de características como flexibilidade, reusabilidade, padronização, entre outras.

Inicialmente, a maior necessidade da educação *on-line* era de disponibilização de conteúdo na *Web*. Esta necessidade evoluiu trazendo questões mais complexas para serem resolvidos, como é o caso do acompanhamento da aprendizagem, assunto bastante abordado em pesquisas mais recentes.

O aumento da complexidade das aplicações destinadas ao suporte de práticas pedagógicas em ambientes *Web*, no que diz respeito, tanto às questões educacionais, quanto às necessidades tecnológicas, exige a utilização de recursos mais especializados na sua solução. Por isso, o uso de uma extensão do UML específica para dar suporte ao desenvolvimento de aplicações *Web*.

A *UML-based Web Engineering* (UWE) (KOCH et al, 2001) é uma metodologia orientada a objetos e iterativa, baseada em padrões de processos de desenvolvimento unificado de software. A modelagem UWE especifica no formato *Web*, os Modelos de Casos de Uso e de Classes, resultam os modelos Conceitual, Navegacional e de Apresentação.

Segundo Koch (2001), a UWE se diferencia das metodologias tradicionais aplicadas aos sistemas hipermídia pela especificidade das atividades a serem executadas em cada fase do projeto, como também, os artefatos resultantes e os especialistas responsáveis por estas atividades. A UWE faz uma separação clara entre modelagem do usuário, conteúdo, estrutura, navegação, apresentação e mecanismo de adaptação.

Apesar de haver muitas ferramentas disponíveis para elaboração de exercícios *on-line*, as mesmas apresentam deficiências, citadas anteriormente, que muitas vezes inviabilizam o seu uso. Porém, desenvolver completamente uma ferramenta de elaboração de exercícios de verificação *on-line* para acompanhamento da aprendizagem é tarefa complexa envolvendo inúmeras funcionalidades.

Neste sentido, surge a proposta da modelagem de um *framework* visando agregar, organizar e disponibilizar diferentes funcionalidades necessárias na elaboração de EVAs *on-line* contemplando dois grandes domínios: o pedagógico e o tecnológico.

A especificação deste *framework* reuniu as funcionalidades básicas para geração de EVAs *on-line*, a partir do reuso de recursos já disponíveis em outras ferramentas de código aberto ou disponibilizadas em componentes de software; apresentou novas características e recursos necessários para o acompanhamento do aprendizado; incorporou uma metodologia e uma teoria pedagógica na elaboração dos exercícios; ampliou formatos das questões geradas; adequou o *feedback* apresentado para melhor detecção, correção e atuação sobre o erro; inseriu suporte para mediação pedagógica para auxiliar a compreensão e resolução dos exercícios.

1.4 ESTRUTURA DO TRABALHO

A presente pesquisa foi estruturada em sete capítulos onde no Capitulo 1 faz uma breve contextualização do problema, objetivo geral, específicos e motivação para o desenvolvimento dessa pesquisa.

O Capitulo 2 apresenta conceitos, tipos e estruturas de *framework*s que é de suma importância para o desenvolvimento desse trabalho tendo em vista que a proposta é a modelagem de um *framework*.

O Capitulo 3 visa aprofundar os conhecimentos em exercício de verificação de aprendizado detalhando os instrumentos, estratégias e metodologias existentes.

A notação de modelagem UWE, empregada no desenvolvimento dessa pesquisa, é descrita no Capitulo 4, bem como seus atributos, diagramas e etapas.

O Capitulo 5 apresenta trabalhos correlatos que serviram como base para a modelagem do *framework* proposto.

O processo de desenvolvimento do *framework* T2A, que é o resultado dessa pesquisa, é descrito no Capitulo 6.

Finalmente, no Capitulo 7 são relatadas as conclusões dessa pesquisa e propostos alguns trabalhos futuros.

2. *FRAMEWORKS*

Com o aumento da demanda e da complexidade de produtos de software, novas perspectivas e esforços vêm sendo aplicados para a melhoria das práticas de seu desenvolvimento, no sentido de proporcionar e ampliar usabilidade, robustez, confiabilidade, flexibilidade, adaptabilidade, mais facilidade e simplicidade na instalação e implantação (SOMMERVILLE, 2007). Outras exigências como produtividade, qualidade, redução de custos e esforços estão cada vez mais presentes e desafiam os desenvolvedores de software corporativos e comerciais. Dentro deste contexto, a reutilização é ponto chave para dar suporte às necessidades e exigências de desenvolvimento de software levantadas. Por meio da reutilização de recursos de software, agrega-se uma maior qualidade e reduz-se o esforço de desenvolvimento em um projeto (GIMENES; HUZITA, 2005).

Na Engenharia de Software, busca-se aplicar a reutilização desde o projeto, desenvolvimento até sua implementação. A granularidade do reuso, e a forma de aplicação, depende do paradigma de programação e do recurso utilizado. Na abordagem orientada a objetos, por exemplo, as entidades reusadas podem ser projeto, classes, objetos, código, entre outras (MATTSSON, 2000). Já na abordagem de desenvolvimento baseado em componentes de software, a granularidade é ampliada ao nível de uma funcionalidade (GIMENES; HUZITA, 2005). Neste caso, o componente, definido como “um conjunto independente de funcionalidades reutilizáveis que permite ter acesso a seus serviços através de interfaces” (BROWN, 1997), é um provedor de serviços (SOMMERVILLE, 2007).

A difusão do reuso no desenvolvimento de software e os avanços no sentido de padronização, mais especificamente com o uso de padrões de Projeto (Design Patterns) (GAMMA ET AL, 1995), outros recursos foram criados para ampliar e facilitar o reuso. Um desses recursos é o *Framework*.

Um *Framework* é uma infra-estrutura genérica, formada por um conjunto de entidades reutilizáveis a partir de um modelo, que captura funcionalidades comuns em várias aplicações de um determinado domínio, que pode ser adaptada para solucionar problemas específicos desse domínio, servindo como um modelo para a construção de aplicações utilizando implicitamente os conceitos de adaptabilidade, flexibilidade, manutenibilidade, reusabilidade, entre outros (SCHMIDT; FAYAD, 1997; PREE, 1995).

De acordo com Johnson (1997), um *framework* reusa: a) análise, pois descreve os tipos de objetos e como um problema maior pode ser dividido em problemas menores; b) projeto, porque contém algoritmos abstratos e descreve a interface que deve ser implementada e as restrições de implementação; e c) código por tornar mais fácil desenvolver uma biblioteca de componentes compatíveis e porque o desenvolvedor poderá herdar e especializar as classes disponibilizadas pelo *framework*.

A utilização de *frameworks* proporciona que uma gama de produtos seja desenvolvida, a partir de uma estrutura genérica que oferece conceitos mais gerais das aplicações (PINTO, 2000).

Nesse capítulo serão expostos os principais conceitos sobre *frameworks* encontrados na literatura. Inicialmente são descritas algumas definições sobre *frameworks*. Em seguida, descreve-se sobre a classificação dos mesmos em duas categorias: *frameworks* de aplicação orientado a objetos e *frameworks* de componentes. Criaremos também os papéis envolvidos no desenvolvimento e uso do mesmo, e os benefícios e desafios resultantes da aplicação de *frameworks* em um projeto.

2.1. DEFINIÇÕES DE *FRAMEWORKS*

Fayad et al (1999b) e Johnson; Foote (1988) definem um *framework* como um agregado de classes que formam um projeto abstrato para a solução de problemas com características comuns.

Mattsson (1996, 2000) refere-se a *framework* como uma arquitetura desenvolvida objetivando a máxima reutilização, fundamentada em uma gama de classes abstratas e concretas, com alto grau de especialização.

Já Johnson (1991) e Gamma et al (1995) estabelecem que um *framework* é um conjunto de objetos que visam atender às responsabilidades encontradas em uma aplicação específica ou um domínio de aplicação.

Segundo Buschmann et al (1996), Pree (1995) e Pinto (2000) um *framework* é projetado de forma parcialmente completa a fim de ser especializado mais tarde conforme as necessidades do projeto.

Szyperski (1997) define um *framework* de componentes como uma entidade de software, que fornece suporte a componentes que respeitam determinado modelo e dispõem uma forma de integrar as instâncias desses componentes ao *framework* de componentes. Ele determina as condições obrigatórias para a execução desse componente e regula a interação entre as instâncias dos mesmos.

Esse tipo de *framework* pode ser implantado em uma aplicação de forma única, onde ele estabelece uma ilha de componentes ao seu redor, ou ainda, mutuamente a outros componentes ou *frameworks* de componentes. Pode-se citar como exemplo o OpenDoc (APPLE, 2005) e o BlackBox (OBERON, 2005).

Encontra-se na literatura também denominações diferenciadas como: *framework* orientado a objetos, *framework* de componentes, *framework* conceitual, *framework* de software, etc.

Por mais que pareçam diferentes, as definições encontradas na literatura acabam por se complementarem, já que não são contraditórias. Porém, neste trabalho adotaremos o termo *framework* para referenciar uma arquitetura de software, que fornece um conjunto de funcionalidades usadas para o desenvolvimento de aplicações por meio de modelos de associação e interação bem definidos.

2.2. COMPOSIÇÃO DOS *FRAMEWORKS*

Um *framework* é composto, basicamente, por um núcleo e por pontos de flexibilização (PREE, 1995). Os aspectos de um *framework* que não são projetados para adaptação (*frozen spots*) são representados por meio de métodos *template* e constituem o núcleo normalmente referenciado como kernel. O kernel é o responsável pelo motor de execução do *framework* e compreende a parte genérica de uma aplicação para todos os problemas de um determinado domínio. As partes adaptáveis do *framework* são os pontos de flexibilização denominados de *hot spots*. Os *hot spot* são pontos de extensão onde serão aplicadas as particularidades de cada sistema, local onde ocorre a customização do software para uma determinada aplicação. Um *hot spot* é uma construção abstrata, que não possui implementação, e deve ser especializada para necessidades específicas da aplicação a ser gerada.

Considerando a orientação a objetos, os *hot spots* são estabelecidos em classes abstratas ou interfaces, que são implementadas ou estendidas pelas instâncias das aplicações. A especialização pode ser realizada tanto por herança como por delegação, dependendo da forma como os *hot spots* foram planejados (CRESPO, 2000).

2.3. CLASSIFICAÇÃO DOS *FRAMEWORKS*

Na literatura são encontradas duas possíveis classificações de *frameworks*: uma que considera seu escopo ou domínio de aplicação (TALIGENT, 1994; FAYAD et al, 1999a, 1999b; FAYAD; JOHNSON, 2000) e outra enfoca a técnica de implementação de aplicações com o *framework* (FAYAD et al, 1999b).

2.3.1. *Frameworks* Quanto ao Escopo de Aplicação

De acordo com os domínios de problemas que abrangem, os *frameworks* podem ainda ser classificados em (FAYAD et al, 1999b):

a) *frameworks* de infra-estrutura de sistemas: consistem em camadas de infra-estrutura de software, sendo utilizados como base para construção de outras aplicações e considerado de uso restrito. Como exemplo pode-se citar os *frameworks* para suporte à comunicação de dados e apoio à utilização de banco de dados, os sistemas operacionais, *frameworks* de interfaces gráficas e ferramentas de processamento de linguagens, Hibernate, Struts, Java Communication and Threads APIs;

b) *frameworks* de integração de *middleware* (*framework* horizontal): são empregados na integração de aplicações e componentes distribuídos freqüentemente usados em aplicações distribuídas, de forma a obter um ambiente de execução integrado. Estes *frameworks* encapsulam conhecimento aplicável a uma vasta gama de aplicações e resolvem apenas uma fatia do problema da aplicação. Eles possibilitam que os desenvolvedores trabalhem em um ambiente distribuído similarmente a um ambiente não distribuído, pois o baixo nível da comunicação entre componentes distribuídos já está implementado pelo *framework*. Como

exemplo destes pode-se citar: *frameworks* de integração de *middleware* Java RMI (RMI-IIOP, 2005) e *frameworks* ORB (Object Request Broker), Corba, EJB (Enterprise Java Beans);

c) *frameworks* de aplicações corporativas (*framework* vertical): são focados para domínio de aplicação específico encapsulando conhecimento utilizável em aplicações pertencendo a um domínio particular de problema e resolvem boa parte da aplicação. Como exemplo tem-se os domínios da aviação, telecomunicações, financeiro, manufatura, controle de produção e multimídia.

A grande vantagem de empregar os *frameworks* de infra-estrutura e de integração de *middleware* é que eles fornecerem mecanismos que visam integrar componentes distribuídos e tratam aspectos de infra-estrutura, ou seja, os detalhes de baixo nível são tratados por esses *frameworks* possibilitando assim que o desenvolvedor da aplicação foque seus esforços no objetivo principal da aplicação. Estes *frameworks* englobam boa parte dos requisitos de diversos domínios de aplicação. Sendo assim, há uma facilidade de encontrar uma solução disponível no mercado e na sua maioria é preferível utilizar uma dessas soluções a desenvolver uma própria.

Os *frameworks* de aplicação tornam-se mais caros de serem desenvolvidos ou comprados que os citados acima por requerer especialistas do domínio de aplicação para construí-lo. Contudo, seu custo-benefício se justifica por proporcionar um retorno considerável tendo em vista, que essa modalidade de *framework* encapsula o conhecimento sobre o domínio onde a instituição atua (FAYAD et al, 1999b). Para este trabalho, utilizaremos a definição de *framework* de aplicação, pois o objetivo é criar um *framework* partindo de um domínio particular, visando à criação de um modelo para elaboração de avaliação da aprendizagem, e testes de avaliação de aprendizagem

2.3.2. *Frameworks* Quanto a Técnica de Implementação de Aplicações

Outra forma de classificação de *framework*, segundo Fayad et al (1999b), é pela forma utilizada para estendê-los que são: caixa branca, caixa preta ou caixa cinza.

Os *frameworks* caixa branca, ou *white box*, são instanciados usando herança, por meio da implementação de *Template Methods* (GAMMA et al, 1995), dessa forma os métodos abstratos são implementados nas classes filhas, ou na redefinição de métodos ganchos (*hook methods*) (PREE, 1995).

Os *frameworks* caixa preta, ou *black box*, são instanciados por meio de configurações e composições definindo-se as classes que implementam determinada interface. Os *hot spots* desses *frameworks* são estabelecidos seguindo o padrão de projetos *Strategy*[1] (GAMMA et al, 1995).

De acordo com o descrito anteriormente, os *frameworks* caixa branca necessitam de um bom conhecimento sobre as estruturas internas por parte do desenvolvedor responsável pela instanciação do *framework*, além de serem mais difíceis de usar. Entretanto, o caixa preta não possui essa exigência, mas acaba por ser mais rígido. Nesse contexto, surgem os *frameworks* caixa cinza, chamados também de caixa de vidro, respectivamente: *gray box* e *glass box*, que unem as características dos *frameworks* caixa branca e preta a fim de suavizar as desvantagens encontrada dos dois.

2.4. DESENVOLVIMENTO E USO DE UM *FRAMEWORK*: CARACTERÍSTICAS, ARQUITETURA, PAPÉIS, PROJETO E METODOLOGIAS

O foco, na construção de um *framework*, é capturar o comportamento geral de um domínio de aplicação e desenvolver uma infra-estrutura de controle capaz de representar este

modelo. Segundo Carneiro (2003), um *framework* deve ser simples de ser aprendido pelo desenvolvedor de aplicações, prover características suficientes que possam ser usadas e métodos flexíveis para as características que podem ser mudadas.

As características básicas que devem estar presentes em um *framework*, os elementos que compõem sua arquitetura, os papéis envolvimentos no seu desenvolvimento e uso, os desafios e metodologias de projeto do *framework* são discutidos a seguir.

2.4.1 Características

As características de um *framework* para atingir seus objetivos são (FAYAD; SCHMIDT, 1997; PINTO, 2000):

a) Modularidade: encapsulam detalhes temporários da implementação em interfaces estáveis promovendo assim, a qualidade de software localizando mais facilmente o impacto de mudanças no projeto e na implementação e reduzindo o esforço necessário para compreensão e manutenção de software;

b) Reuso: as interfaces estáveis propiciam reuso, pois definem componentes genéricos que podem ser reaplicados na criação de novas aplicações. A reusabilidade influencia no conhecimento do domínio e no esforço prévio de desenvolvedores experientes a fim de evitar recriação e revalidação de soluções comuns. O reuso ainda pode aumentar a produtividade, produzir melhorarias na qualidade, no desempenho, na confiabilidade e interoperabilidade da aplicação;

c) Extensibilidade: o *framework* fornece métodos de adaptação explícitos possibilitando às aplicações estenderem suas interfaces (variações requeridas pelas

---

1 Esse padrão define uma família de algoritmos tornando cada um deles intercambiáveis ao encapsulá-los. A cada requisição é utilizado um algoritmo escolhido aleatoriamente independentemente dos clientes que o utilizam. (FERNANDES , 2003)

instanciações de uma aplicação). A extensibilidade é essencial para assegurar a especialização de novos serviços e novas características da aplicação;

d) Inversão de controle - IoC (*Inversion of Controll*): o *framework* (ao invés da aplicação) é responsável por definir o curso da aplicação coordenando e sequenciando as atividades. Os métodos definidos pelo usuário para especialização dos algoritmos genéricos são chamados de dentro do próprio *framework*, ao invés de serem chamados do código de aplicação do usuário (JOHNSON; FOOTE, 1988). O *framework* determina que o conjunto de métodos específicos da aplicação invoque em resposta a eventos externos. Essa inversão de controle dá força ao *framework* para servir de esqueletos extensíveis.

Para Fayad et al (1999b) *frameworks* são vistos como uma categoria de abstrações arquiteturais que suportam o projeto e a construção da estrutura lógica, pois atendem aos quatro aspectos da abstração arquitetural:

a) Estrutura: um *framework* é uma composição de classes, cujas colaborações e responsabilidades são especificadas;

b) Funcionalidade: um *framework* embute a funcionalidade comun de uma aplicação, no intuito de prover as características requeridas do domínio podendo também estender ou adaptar as funcionalidades;

c) Abstração: um *framework* representa uma abstração de estruturas e funcionalidades no domínio de aplicação;

d) Reutilização: um *framework* apresenta as características comuns de um contexto de aplicações reutilizáveis no desenvolvimento de aplicações desse domínio.

As principais características de orientação a objetos que tornam o desenvolvimento de *framework* viável são (CARNEIRO, 2003): herança, associação dinâmica, polimorfismo e classes abstratas.

2.4.2 Arquitetura

Um *framework* é composto de elementos em dois níveis (CARNEIRO, 2003):

a) Elementos do domínio: identificando as particularidades relevantes do domínio úteis nas aplicações;

b) Elementos estruturais: definido os componentes que facilitam a adaptação e a evolução do *framework*. As características estruturais incluem um arranjo de lógica (projeto) e estrutura física (classes) do *framework*.

Os elementos do *framework* são, na sua maioria, descritos com classes abstratas. A composição da arquitetura do *framework* inclui a interface do componente e geralmente um núcleo para implementação. A separação entre interfaces e a implementação constitui uma boa prática, resultando em um *framework* com camadas estruturadas. A estruturação em camadas visa atender ao padrão arquitetural *Layers* – Camadas (BUSCHMANN et al, 1996) As camadas mais abstratas são independentes das camadas mais concretas. As classes abstratas irão conter o esqueleto e os métodos abstratos. Um *framework* pode especificar as implementações padrões para estes métodos ou não implementá-los. Uma aplicação gerada pelo *framework* pode usar as implementações padrões ou especificar novas classes concretas que herdam as classes abstratas e implementam seus métodos abstratos (CARNEIRO, 2003).

O modelo abaixo (Figura 1) apresenta a divisão em camadas do *framework* de Viljamaa (2001) – camada de interface, de implementação núcleo e a padrão.

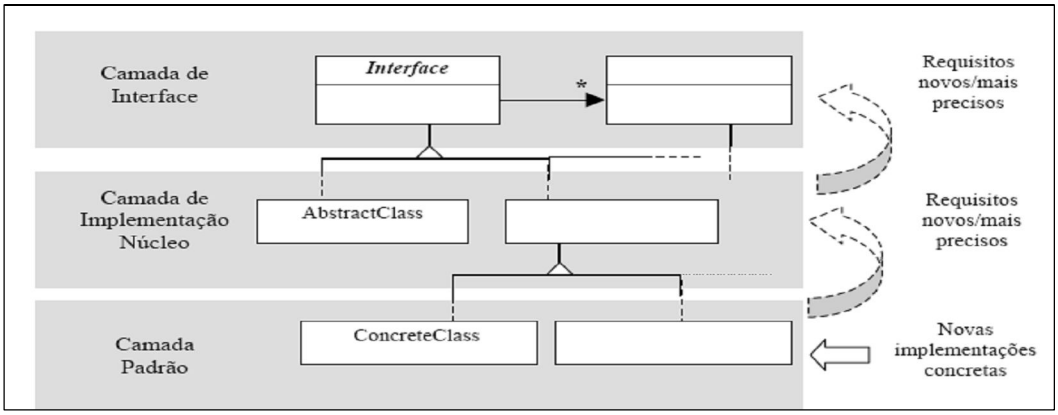

Figura 1. Divisão do *framework* em camadas
Fonte: VILJAMAA (2001)

A camada de interface é composta de interfaces abstratas que definem, por meio de assinaturas dos métodos, os componentes básicos (serviços e relacionamentos) do *framework*.

A camada de implementação núcleo é responsável pelo comportamento padrão do *framework* implementando parcialmente, com as classes abstratas, a camada de interface.

A camada padrão contém implementações completas (classes concretas) para funcionalidades comuns do domínio da aplicação (VILJAMAA, 2001).

2.4.3 Papéis

Dentre os profissionais que atuam na criação, manutenção e uso de um *framework*, os autores Froehlich et al (1997a, 1997b) e Pinto (2000) destacam os seguintes: projetista do *framework*, mantenedor do *framework* e o desenvolvedor de aplicações. Vale lembrar que esses papéis não são necessariamente desempenhados por pessoas distintas.

O projetista do *framework* responde pelo desenvolvimento da estrutura básica do *framework*. É ele quem determina os *hot spots* do *framework* e faz o levantamento de requisitos. O mantenedor é o responsável por ajustar e adicionar novas funcionalidades ao projeto do *framework*, especializando o mesmo para que novas características sejam

adicionadas a ele, por parte dos desenvolvedores de aplicação. O desenvolvedor por sua vez implementa os *hot spots* a fim de caracterizar a aplicação que é uma instância do *framework*.

2.4.4 Projeto e Metodologias

O desenvolvimento de um *framework* consiste na produção de uma estrutura de classes, que se adaptem a um conjunto de aplicações diferentes apresentando funcionalidades do domínio tratado.

Para Carneiro (2003), projetar *frameworks* flexíveis e reutilizáveis é difícil e trabalhoso, pois há vários problemas durante o seu desenvolvimento e utilização, entre os quais cita:

a) dificuldade na detecção de *hot spots* que determina quais propriedades devem ser flexíveis;
b) dúvidas na escolha da maneira mais adequada de implementar os *hot spots* e verificar o comportamento do *framework*;
c) complexidade no gerenciamento, entendimento e evolução da sua estrutura (interfaces, hierarquia de classes e outros);
d) inadequação quanto à generalidade de sua aplicação;
e) complexidade e curva acentuada de aprendizagem.

Mattsson (2000) identificou que o processo de desenvolver e utilizar um *framework* orientado a objetos, de forma geral, compreende três atividades macros: análise de domínio, projeto de *framework* e instanciação do *framework* para geração de aplicações.

Na análise de domínio o conhecimento é identificado e organizado sobre alguma classe de problemas para dar suporte à descrição e solução destes (ARANGO; PRIETO-DÍAZ, 1991).

O projeto de *framework* tem como objetivo alcançar a maior flexibilidade possível a partir das abstrações corretas e identificar as partes estáveis e variáveis do *framework.*

A instanciação do *framework* vai depender do tipo de técnica utilizada para sua implementação: no tipo caixa branca o usuário constrói classes a partir das classes disponíveis, e no tipo caixa preta o usuário deve escolher uma das classes disponíveis.

A literatura apresenta diferentes metodologias para desenvolvimento de um *framework*, a partir destas atividades macro, citadas a seguir.

O Projeto Dirigido por Exemplos (JOHNSON, 1993) apresenta um processo, composto de etapas de análise do domínio do problema, projeto de abstrações e teste do *framework.*

Taligent (1994) propõe o desenvolvimento de *frameworks* de estruturas menores e mais simples em uma seqüência de quatro passos: 1) análise do domínio do problema, 2) definição da arquitetura e projeto, 3) implementação do *framework*, e 4) disponibilização do *framework.*

A proposta de Pree (1999) é o Projeto Dirigido por *Hot Spot*, cuja base está na identificação dos *hot spots* na estrutura de classes de um domínio, constando das seguintes etapas: identificação de classes, identificados os *hot spots*, projeto do *framework* e refinamento da estrutura do *framework*.

A abordagem de Carneiro (2003) é um processo de desenvolvimento iterativo e incremental composto pelos passos: 1) análise de domínio; 2) construção do *framework*, passando por ciclos de desenvolvimento onde a cada ciclo é realizado: coleta dos requisitos, análise e detecção dos *hot spots*, projeto do *framework* com o detalhamento do modelo estático de objetos criado na análise e inclusão dos *hot spots* para geração do diagrama de classe de projeto, implementação e instanciação exemplo com a geração do código do

framework, e elaboração do roteiro constituindo em um manual de uso do *framework*; e 3) instanciação da aplicação.

A seguir explicaremos cada passo que compõem essa metodologia e ao final de cada passo apresentaremos uma síntese que indica os dados de entrada, o processo que é desenvolvido naquele passo e o produto final do respectivo passo.

2.5 ETAPAS PARA O DESENVOLVIMENTO INTERATIVO E INCREMENTAL

O processo de desenvolvimento iterativo e incremental é assim nominado devido à proposta de em cada iteração ser concluída com um aplicativo funcionando, ou seja, a primeira iteração resulta em um aplicativo simples e a cada nova iteração esse aplicativo é aperfeiçoado e se torna mais complexo até atingir o objetivo

As etapas que compõem esse processo são:

a) análise de domínio;
b) construção;
c) diversos ciclos de desenvolvimento onde são adicionados os requisitos;
d) avaliação dos *hot spots* e *frozen spots*;
e) adaptação do projeto para abranger os novos requisitos.

A figura 2 ilustra o processo de desenvolvimento:

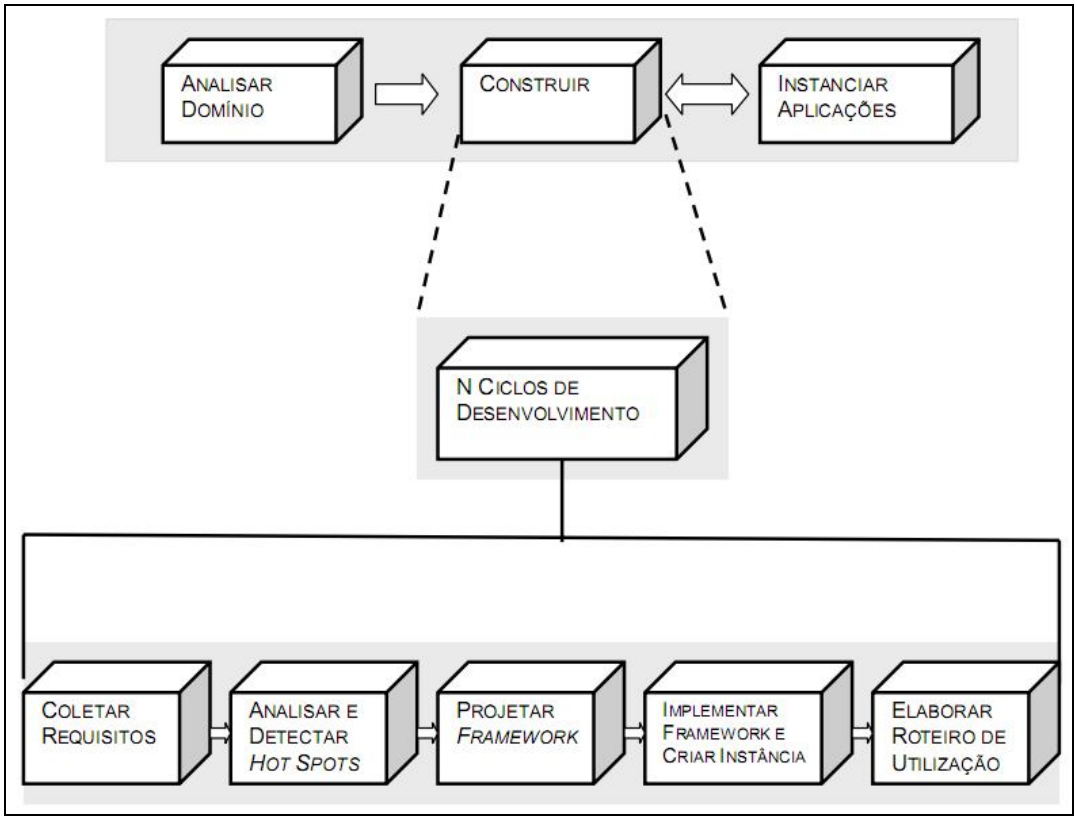

Figura 2. Processo de desenvolvimento iterativo e incremental
Fonte: Carneiro (2003)

2.5.1 Passo 1 - Analisar Domínio

O primeiro passo é de grande importância, pois define a entrada para o desenvolvimento, ou seja, nessa etapa que são identificados os objetos, operações e relações que os especialistas afirmam ser relevantes para a especificação do domínio do problema. Adicionado a isso são estabelecidos, num primeiro momento, os requisitos iniciais por meio da observação, entrevistas e recursos materiais.

Como resultado dessa etapa se obtém dois documentos que são: o escopo do domínio e um modelo estático ou conceitual, contendo objetos importantes do domínio. O escopo será de suma importância na coleta de requisitos, pois ajudará na identificação de requisitos válidos ou não para o problema descrito.

O modelo estático por sua vez será formado pelas classes e objetos do domínio que fazem referência ao mundo real e são comumente encontradas em aplicações com o domínio similar. Esse modelo é desenvolvido em alto nível excluindo assim, detalhes do *framework* e processo de desenvolvimento.

Síntese:

Passo 1: Análise de Domínio

Entrada: Obter informações sobre o domínio do problema através da observação, entrevistas, padrões já existentes e através de pessoas com conhecimento no domínio.

Procedimentos:

- Descrever suscintamente o objetivo do framework.
- Identificar seus limites.
- Descrever seus benefícios.
- Representar os objetos do domínio no modelo estático.

Saídas:

- Modelo do Domínio
  - Definição do escopo do domínio.
  - Modelo estático preliminar.

Figura 3. Síntese da análise de domínio.
Fonte: Carneiro (2003).

2.5.2 Passo 2 – Construir

O segundo passo consiste no desenvolvimento do aplicativo. Por conter diversas etapas o autor dividiu-o em passos menos complexos que definem os ciclos de desenvolvimento.

2.5.2.1 Passo 2.1 – Coletar Requisitos

Os requisitos refletem o que se espera do aplicativo final. Sendo assim, a coleta de requisitos consiste em identificar e documentar as necessidades em uma linguagem clara que garanta o entendimento tanto da equipe de desenvolvimento, quanto de qualquer pessoa que ler esse documento.

Outra abordagem para a especificação dos requisitos são os casos de uso que podem ser definidos como uma descrição narrativa de processos do domínio. Os casos de uso podem ser classificados como concretos e abstratos onde o primeiro é formado por um ator que produz um resultado, e o segundo é composto por uma seqüência de ações são utilizadas em outros casos de uso.

É muito importante identificar as inconsistências e ambiguidades nessa fase, pois se não forem solucionadas acarretarão em problemas futuros que resultarão em retrabalho. Como conclusão dessa etapa obtém-se a Especificação dos Requisitos que detalha uma lista com todos os requisitos funcionais e não funcionais.

Síntese:

Passo 2.1 : Coleta de Requisitos

Entrada: Modelo do domínio, aplicações existentes, usuários e pessoas com conhecimento no domínio

Procedimentos:

- Obter informações sobre os requisitos a partir de diversas fontes, tais como grupos de usuários e pessoas com conhecimento no domínio através de entrevistas e reuniões.
- Verificar, nas aplicações existentes, os requisitos funcionais e não funcionais comuns e gerar a lista de requisitos do *framework*.
- Verificar características relevantes do domínio e abstraí-las para a lista de requisitos do *framework*.
- Criar casos de uso abstratos, através da revisão dos documentos da especificação de requisitos.

Saída:

- Modelo de Requisitos
  - Especificação de Requisitos.
  - Casos de Uso abstratos.

Figura 4. Síntese da coleta de requisitos
Fonte: Carneiro (2003)

2.5.2.2 Passo 2.2 – Análise e Detecção de *Hot Spots*

A base para a identificação dos *hot spots* são o modelo de domínio, modelo de requisitos ou aplicações já existentes. Os *hot sposts* são os pontos de flexíbilidade para a especialização das aplicações que serão desenvolvidas a partir do *framework*.

A possibilidade de habilitar uma característica disponível no *framework* permite que a adição ou remoção seja feita independente da implementação padrão ou *template*. Essas adições são feitas basicamente por meio de extensão das classes existentes ou implementação de novas classes.

Já o processo de desabilitar é simplesmente desativar uma característica pré-existente no *framework*. Contudo a substituição de alguma característica consiste em

desabilitar uma existente e adicionar uma nova, e não fazer a edição daquela oferecida pelo *framework*.

Síntese:

Passo 2.2: Analisar e Detectar *Hot Spots*

Entrada:

- Modelo de domínio, requisitos e aplicações existentes (se disponíveis).

Procedimentos:

- Capturar os conceitos de importância que estão dentro do domínio;
- Identificar os *hot spots* verificando que aspectos fixos podem ser flexibilizados;
- Catalogar os *hot spots* e incluir para cada *hot spot* catalogado o tipo de adaptação que será requerida e o grau de apoio fornecido.

Saída:

- Diagrama de Classes;
- Catálogo de *Hot Spots*.

Figura 5. Síntese da analise e detecção de *hot spots*.
Fonte: Carneiro (2003)

2.5.2.3 Passo 2.3 – Projetar *Framework*

Para a etapa onde o *framework* é projetado toma-se como ponto de partida o modelo estático de objetos desenvolvido na análise. Para dar origem ao diagrama de classes esse modelo será alterado com a finalidade de abranger os *hot spots*. Dessa forma o diagrama de classes do projeto define onde serão encontrados os pontos de flexibilidade (*hot spots*) e as partes fixas (*frozen spots*).

O grande diferencial entre o modelo conceitual e o diagrama de classes é que o último define as entidades que irão compor o *framework* e não os conceitos do mundo real. Para realizar essa transição do primeiro modelo para o diagrama de classes deve-se proceder

da forma mais linear possível mantendo os nomes de objetos identificados em etapas anteriores.

Durante as etapas anteriores muitos dos conceitos comuns às aplicações foram destacados, dessa forma o trabalho realizado nessa etapa é de abstrair esses conceitos para um nível mais baixo sempre considerando a reutilização desses objetos mais tarde.

Síntese:

Passo 2.3: Projetar *Framework*

Entrada:

- Modelo estático de objetos da análise, catálogo de *hot spots*, catálogo de padrões de projeto.

Procedimentos:

- Detalhar o modelo estático da análise.
- Aplicar padrões de projeto, incluindo os *hot spots* e as partes fixas.
- Estruturar as classes em camadas.

Saída:

- Diagrama de Classes de Projeto.

Figura 6. Síntese do projeto do *framework.*
Fonte: Carneiro (2003)

2.5.2.4 Passo 2.4 – Implementação e Instanciação Exemplo

Com o diagrama de classes finalizado há detalhes suficientes para a geração do código do *framework*, pois a maioria das decisões e trabalho foram realizados nos passos anteriores. A etapa de programação não é apenas para cumprir o ciclo, é nessa etapa que são identificados problemas que não foram lembrados na análise de requisitos nem nos diagramas, sendo assim muitos problemas são resolvidos nessa etapa.

O beneficio encontrado nessa modelagem de desenvolvimento iterativo e incremental é justamente o fato de que os resultados de um ciclo serão a entrada de dados do ciclo seguinte. Dessa forma há um refinamento constante dos requisitos, análise e do projeto.

A instanciação de exemplo dá-se por meio da geração de uma ou mais aplicações com a finalidade de verificar se o *framework* está atendendo aos requisitos sem perder a flexibilidade desejada.

Síntese:

Passo 2.4: Implementação e Instanciação exemplo

Entrada:

- Diagrama de Classes de Projeto.

Procedimentos:

- Escrever código fonte para as classes e métodos.
- Criar uma aplicação a partir do framework codificado.

Saída:

- Código fonte do framework.
- Aplicação exemplo.

Figura 7. Síntese da implementação e instanciação exemplo
Fonte: Carneiro (2003)

2.5.2.5 Passo 2.5 – Elaborar Roteiro

O roteiro tem a finalidade de funcionar como um manual, onde irá explicar ao desenvolvedor de aplicações como usufruir do *framework*. Baseado nisso o roteiro necessita conter informações completas que permitam o entendimento do *framework*, como o mesmo trabalha e se comporta a determinadas ações.

É de suma importância que o roteiro exponha uma visão geral do *framework* e destaque as classes concretas que são oferecidas para uso imediato e quais podem ser estendidas para uma futura implementação e especialização da aplicação. Além dessas

informações deve conter outras a cerca do que pode ser desenvolvido, e quais as classes abstratas disponíveis, entre diversos aspectos importantes para o futuro usuário.

Síntese:

Passo 2.5: Elaborar Roteiro

Entrada:

- Documentação do *framework* e aplicação exemplo.

Procedimentos:

- Fazer uma descrição do objetivo do *framework*.
- Listar o que é necessário para a aplicação utilizar o *framework*.
- Explicar cada elemento básico do *framework* que a aplicação poderá utilizar em termos de classes concretas e/ou abstratas.
- Mostrar exemplos que demostrem a utilização dos elementos do *framework*.
- No caso de *framework* "caixa branca" explicar como e onde o desenvolvedor poderá estender as funcionalidades do *framework* e inserir seu próprio código.
- No caso de *framework* "caixa preta" explicar como os elementos do *framework* são acoplados a aplicação.

Saída:

- Roteiro.

Figura 8. Síntese da elaboração do roteiro
Fonte: Carneiro (2003)

2.5.3 Passo 3 - Instanciar Aplicações

O desenvolvimento de uma aplicação baseada em *framework* não é simples, pois existe a escolha de um mais indicado entre uma família de *frameworks* para a aplicação que se deseja criar. A escolha deve visar os requisitos levantados para a aplicação que se deseja implementar, além da documentação e do roteiro do framework, pois são esses fatores que indicarão qual dos *frameworks* oferecidos é mais indicado.

A maneira mais simples de se empregar um *framework* é desenvolver uma aplicação baseada nos componentes oferecidos pelo mesmo, pois dessa forma a comunicação entre esses componentes é algo concreto e pronto. Para o desenvolvedor de aplicação basta saber apenas quais objetos se conectam entre si e não como eles fazem isso.

Contudo isso não é regra para se trabalhar com *framework*s, sendo assim, há determinados *frameworks* que a cada uso requerem novas subclasses. Essa forma exige um maior conhecimento por parte do desenvolvedor de aplicações, pois o mesmo deve estender as classes abstratas disponíveis no núcleo do *framework* a fim de gerar a aplicação desejada. Essa maneira necessita de um esforço maior, mas resulta em uma aplicação mais poderosa.

Um *framework* sem documentação tornará o trabalho do desenvolvedor de aplicação algo extremamente difícil. Isso acarreta em uma perda de tempo desenvolvendo funcionalidades que já existem e estão disponíveis para uso imediato, mas que o desenvolvedor não possuía conhecimento. Sendo assim, faz com que o uso de *frameworks* seja inútil porque a característica chave é a reutilização visando à economia de tempo e esforços.

Síntese:

Passo 3: Instanciar Aplicação

Entrada:

- Requisitos da aplicação, documentação do *framework* e roteiro.

Procedimentos:

- Escolher o *framework* adequado;
- Verificar se o *framework* é do tipo caixa branca ou caixa preta;
- Checar o roteiro para verificar:
  - Quais classes podem ser criadas ou reutilizadas;
  - Quais métodos estão disponíveis.
- Desenvolver aplicação.

Saída:

- Aplicação.

Figura 9. Síntese da instância da aplicação
Fonte: Carneiro (2003)

3 AVALIAÇÃO DA APRENDIZAGEM E TESTES DE AVALIAÇÃO DE APRENDIZAGEM

Luckesi (2003) afirma que a avaliação deve ser traduzida como um instrumento de identificação do estagio de aprendizagem do estudante visando sempre diagnosticar problemas e tomar decisões que o façam avançar no processo de aprendizagem.

Durante o passar dos tempos, o conceito de avaliação foi mudando de “mensurar” o rendimento escolar para “descrever” padrões e critérios para indicação do sucesso. Após isso, passou pela fase em que “julgar” o conjunto de todas as dimensões do objeto, era o mais importante evoluindo para a “negociação” onde estudante e professores entravam em um acordo sobre como prosseguir, até atingir o estagio atual onde o objetivo é “emancipar” visando fortalecer a competência do professor e do aluno.

Isso faz da avaliação uma ferramenta de orientação, cooperação, interação e integração do processo ensino e aprendizagem deixando para traz o objetivo de ser um marco final do processo onde o estudante apenas era aprovado ou não.

Para atingir esse objetivo a avaliação pode ser classificada de diversas maneiras, por exemplo, quanto a sua função, ou seja, se ela é diagnóstica, formativa ou somativa. Quanto aos seus objetivos sendo esses definidos pelo professor para estabelecer se o estudante alcançou-os ou quanto à sua referência podendo esta ser normativa ou criterial. Há também a avaliação da aprendizagem quanto à mediação onde esta proporciona um momento de reflexão e de desafio ao professor, que visa favorecer a troca de idéia entre e com seus estudantes.

Além do conceito de avaliação, das etapas na evolução do foco e da classificação da mesma, trataremos nesse capítulo as técnicas e instrumentos de avaliação, bem como os testes e as ferramentas de desenvolvimento dos mesmos.

3.1 AVALIAÇÃO DA APRENDIZAGEM: CONCEITOS, EVOLUÇÃO E DENOMINAÇÕES

A avaliação da aprendizagem é um tema recorrente na discussão de autores em suas pesquisas, e entre os citados na literatura encontram-se: Luckesi, 2003; Vasconcelos, 1996; Villas Boas, 2001; Demo, 1999; Melchior, 1999; Esteban, 2001; Anastasiou; Alves, 2003; Perrenoud, 1999; Hadji, 2001; Salinas, 2004; Sacristán, 1998; Hoffmann, 1998; Haydt, 1998; Bloom, 1972. Este tema também se encontra na formação de professores, nas políticas voltadas à educação, entre outros nichos nos quais o processo educacional está inserido.

Segundo Barbosa (2008), a avaliação “é uma tarefa didática necessária e permanente do trabalho docente, que deve acompanhar passo a passo o processo de ensino e aprendizagem”.

Luckesi (1995) define avaliação da aprendizagem, como sendo um juízo de qualidade sobre dados relevantes para uma tomada de decisão e afirma que “a avaliação deverá ser assumida como um instrumento de compreensão do estágio de aprendizagem em que se encontra o estudante, tendo em vista tomar decisões suficientes e satisfatórias para que possa avançar no seu processo de aprendizagem” (LUCKESI, 2003, p. 81). Para Hoffmann (1998),

> O processo avaliativo é um método investigativo e que prescinde da correção tradicional, impositiva e coerciva. Pressupõe isso sim, que o professor esteja cada vez mais alerta e se debruce compreensivamente sobre todas as manifestações do educando. O erro lido em sua lógica, as hipóteses pré-dinamizadas construídas pelo estudante (o “ainda não, mas pode ser”) são elementos dinamizadores da ação avaliativa [...]

Bloom (1972) define avaliação como uma coleta sistemática de evidências por meio das quais se determinam mudanças que ocorrem nos estudantes e como ocorrem.

O conceito de avaliação e suas práticas vêem evoluindo ao longo do tempo e a partir do início do século XX, assumindo, em cada momento, uma atribuição (GUBA; LINCOLN apud FIRME, 1994; ANDRADE, 2008):

a) mensurar o rendimento escolar a partir de instrumentos ou testes para verificação;
b) descrever padrões e critérios para indicação do sucesso ou fracasso em relação aos objetivos da avaliação estabelecidos;
c) julgar o conjunto de todas as dimensões do objeto, considerando as qualidades intrínsecas do objeto (mérito) e as características externas do resultado (relevância), inclusive sobre os próprios objetivos;
d) negociar entre os pares, de forma interativa, as preocupações, proposições ou controvérsias em relação ao objetivo da avaliação fornecendo informações que permitam aos agentes escolares decidir sobre as intervenções e

redirecionamentos que se fizerem necessários objetivando a garantia da aprendizagem do estudante;

e) emancipar buscando fortalecer a competência dos participantes do processo avaliativo, impulsionando a autodeterminação e a busca de auto-aperfeiçoamento.

De acordo com Haydt (1998, p.14)

> [...] a educação: não mudou apenas os métodos de ensino, que se tornaram ativos, mas incluir também a concepção de avaliação. Antes, ela tinha um caráter seletivo, uma vez que era vista apenas como uma forma de classificar e promover o estudante de uma série pra outra ou de um grau para outro. Atualmente, a avaliação assume novas funções, pois é um meio de diagnosticar e de verificar em que medida os objetivos propostos para o processo ensino-aprendizagem estão sendo atingidos.

Dentro dessas perspectivas, a avaliação deixa de ser momento terminal do processo ensino e aprendizagem e assume uma dimensão orientadora, cooperativa, interativa e integradora desse processo. Seus resultados, no decorrer do trabalho conjunto do professor e dos educandos, são comparados com os objetivos propostos a fim de verificar progressos, dificuldades, como também reorientar a prática docente (BARBOSA, 2008).

Sob o enfoque da construção do conhecimento, a avaliação re-significa o erro e a dúvida, anteriormente vistos como fracasso e falta de conhecimento, abordando-os como um elemento significativo para o desenvolvimento da ação educacional ao possibilitar a observação e investigação de como o aprendiz constrói seu conhecimento (TEIXEIRA, 2005).

De acordo com Barbosa (2008), o conceito e a prática da avaliação da aprendizagem pressupõem um referencial teórico, onde estão implícitos os conceitos de pessoa, de sociedade, de educação e avaliação, mesmo que não tenhamos consciência deles. A prática da avaliação, na visão de diferentes pesquisadores (BLOOM, 1972; BARTOLOMEIS, 1977; SAUL, 1988; HOFFMANN, 1998; PERRENOUD, 1999; MEZZAROBA; ALVARENGA, 1999; HAYDT, 2000; MAZZETO, 2003; CAMPOS 2003, ROMANOWSKI E WACHOWICZ, 2003), pode ser classificada pela função que esta exerce.

São encontradas diversas denominações de avaliação citadas em Rosado (1997) e Rabelo (1999) nas quais os tipos de avaliação podem ser classificados quanto: aos objetivos (funcional, programas, políticas educacionais, aprendizado ou a própria avaliação); ao nível de explicitação (explicita ou implícita); a referência ou comparação (norma ou critério); a função (formativa, somativa ou diagnóstica); entre outros (mediadora, emancipatória, classificatória, qualitativa, auto-avaliação, etc.).

Para atingirmos alguns dos objetivos específicos deste trabalho, entre eles o de compreender as formas de avaliação da aprendizagem, e para dar suporte conceitual e metodológico na modelagem dos testes de avaliação de aprendizagem, serão abordadas as classificações de avaliação quanto a sua função (formativa, somativa ou diagnóstica), aos objetivos (de aprendizagem), a referência (critério) e a mediação (mediadora) sob o ponto de vista de Haydt (2000), Bloom (1956, 1983), Rosado (1997) e Rabelo (1999).

3.1.1 Avaliação da Aprendizagem Quanto a Função

Bloom (1956) descreve três categorias funcionais da avaliação de aprendizagem: diagnóstica, formativa e somativa. Estas categorias apresentam funções distintas, porém estão diretamente relacionadas e se complementam no processo de avaliação.

A **avaliação diagnóstica** tem como função determinar a presença ou ausência de habilidades e pré-requisitos e o nível de domínio prévio de conhecimento e habilidades necessárias para iniciar os estudos; ou detectar problemas e dificuldades repetidas de aprendizagem e identificar suas causas. Pode ser aplicada no início de um curso, uma unidade, semestre, ano letivo; ou durante o processo de ensino e aprendizagem, quando o aprendiz evidencia uma incapacidade constante de aproveitamento integral do ensino. A sua ênfase são os comportamentos cognitivos, afetivos e psicomotores, bem como os fatores físicos, psicológicos e ambientais. A atribuição de pontos nesse modelo é baseada em normas

e critérios. O método de relato da avaliação diagnóstica se dá pelo perfil individual de subabilidades.

A **avaliação formativa** tem como propósito o controle de qualidade de cada ciclo do processo ensino e aprendizagem fornecendo *feedback* ao professor e ao aprendiz seu progresso de modo a verificar se o mesmo está atingindo as metas. É por meio desse modelo de avaliação que se assegura dar prosseguimento ao conteúdo com o estudante apto a isso, caso contrário pode-se acarretar problemas para o estudante num futuro próximo como aumento de dificuldade em aprender um novo assunto por não ter uma base de conteúdo sólida.

Esse modelo avaliativo possui a função orientadora uma vez que é por meio dela que o estudante toma conhecimento dos seus erros e acertos. A avaliação formativa objetiva, também oferece a localização de erros em termos de estrutura de um conteúdo, de modo a possibilitar a indicação de técnicas alternativas de recuperação e estimular o aprendiz a buscar mais conhecimento no assunto que ele possui dificuldade.

A aplicação de instrumentos deste tipo de avaliação ocorre durante todo o processo de ensino e aprendizagem e visa os comportamentos cognitivos do aprendiz. Os pontos são atribuídos a partir de critérios pré-definidos e seu método de relatos enfoca o padrão individual de escores relativos a acertos e erros em cada tarefa da hierarquia.

A **avaliação somativa** é utilizada quando se visa classificar, graduar ou atribuir nota ao aprendiz no final de uma unidade, semestre ou curso conforme seu nível de aprendizagem. Nesse modelo são enfatizados na avaliação os comportamentos cognitivos ou afetivos. A atribuição de pontos se dá em função de normas e o método de relato que ocorrem por escore total ou sub-escores obtidos para cada objetivo.

3.1.2 Avaliação da Aprendizagem Quanto aos Objetivos

Observa Haydt, (2000), que quando se avalia o processo de ensino-aprendizagem afirma-se que a avaliação é funcional, ou seja, é empregada em função de objetivos. É de suma importância que os objetivos sejam descritos de uma maneira clara e precisa, pois tanto o professor quanto o aprendiz devem estar ciente dos mesmos. Os objetivos podem ser definidos sob o ponto de vista das instituições, programas, políticas educacionais, aprendizado ou a própria avaliação. O objetivo abordado visa atender ao aprendizado e à avaliação.

Para Zaina (2002, p.17) os objetivos educacionais "são metas definidas com o intuito de identificar, de maneira mais eficaz, onde se deseja chegar através de um conceito ensinado". Estes objetivos educacionais devem informar o conteúdo a ser ministrado, as habilidades que se espera desenvolver e as definições de modificações comportamentais que o aprendiz deverá apresentar no processo de ensino e aprendizagem.

Dentro deste contexto, a avaliação deve ser estabelecida de acordo com os objetivos propostos, do contrário não seria possível saber se o processo de ensino e aprendizagem está evoluindo ou não, sem ter um parâmetro para fazer tal diagnostico.

A partir de problemas de clareza, de nomenclaturas, falhas na comunicação e diante da necessidade de organizar as definições do estudo dos objetivos educacionais, Bloom (1956) e Krathwohl; Bloom; Masia (1964) publicaram dois documentos que tratavam da taxonomia de objetivos educacionais. A primeira publicação *Taxonomia dos objetivos educacionais – domínio do cognitivo* abordou as tarefas intelectuais do estudante. A segunda *Taxonomia dos objetivos educacionais – domínio do afetivo* apresentou os objetivos referentes a sentimentos, emoções, aceitação ou rejeição de uma determinada situação. Nesta pesquisa trataremos apenas o domínio do cognitivo.

A Taxonomia dos objetivos educacionais – domínio do cognitivo de Bloom (1956) é composta por seis categorias de comportamentos observáveis, sendo que os de nível

mais alto agregam comportamento dos níveis inferiores. A figura 10 ilustra essa organização que define a hierarquia da taxonomia.

Figura 10. Hierarquia da Taxonomia dos Objetivos Educacionais de Bloom

Fonte: Adaptado de Zaina L. A. M. (2002)

As categorias de comportamento são descritos de forma resumida na tabela 1.

Tabela 1. Síntese das categorias da Taxonomia de Bloom

Comportamento	Características	Exemplo de verbos utilizados
Conhecimento	Observação e lembrança de informações. Lembrança de conceitos	Definir Descrever Identificar Listar
Compreensão	Entendimento de um conceito. Capacidade de transmitir um conceito com suas próprias palavras.	Descrever Interpretar Distinguir Diferenciar
Aplicação	Resolução de problemas utilizando conceitos compreendidos.	Demonstre Aplique Calcule Ilustre Modifique
Análise	Expressão das idéias dos estudantes a partir de um problema dado.	Analise Compare Explique
Síntese	Uso de idéias já conhecidas para criação de novas. Relacionado ao conhecimento de varias áreas. Grande importância a criatividade.	Invente Desenvolva Modifique
Avaliação	Comparação entre idéias. Realizar escolhas baseadas em argumentos pré-determinados.	Avaliar Decidir Selecionar

Fonte: Zaina L. A. M. (2002)

Os verbos citados para indicar os comportamentos são apenas exemplos para auxiliar na construção dos objetivos educacionais podendo ser utilizados outros desde que o contexto não seja distinto da categoria que se deseja atingir.

Com os objetivos educacionais e as mudanças de comportamento bem delineadas o professor está apto a estabelecer a melhor estratégia no processo de ensino e aprendizagem. Dessa forma, o professor terá como referência os objetivos pré-determinados para elaborar a avaliação e, com seus resultados, poderá visualizar o avanço gradativo dos aprendizes perante aos objetivos e, caso contrário, poderá saber que decisão tomar para garantir que todos alcancem as metas.

3.1.3 Avaliação da Aprendizagem Quanto à Referência

Uma forma de caracterizar a avaliação está relacionada aos elementos de referência: norma ou critério. De maneira geral, a avaliação normativa referencia um conjunto de regras comuns que servem de padrão para medir, classificar e comparar o desempenho dos indivíduos num grupo, na realização da mesma tarefa. Este método é utilizado em avaliações seletivas (testes de seleção para uma vaga de trabalho, vestibular) nas quais se deseja classificar o grupo independente dos erros ou acertos obtidos e sem especificar o que realmente foi aprendido ou seu domínio de conhecimento sobre o conteúdo (ZAINA, 2002).

A avaliação por critérios ou criterial, tem como padrão de referência, ou de comparação, um conjunto de critérios. Neste tipo de avaliação, o conhecimento do sujeito é o resultado da medida da comparação entre sua resposta com os objetivos educacionais ou critérios pré-estabelecidos, de modo a permitir a verificação dos acertos e erros situando-o em relação à sua aprendizagem. Os critérios devem indicar o que será avaliado, tanto com relação aos conhecimentos, quanto às atitudes esperadas (ROSADO, 1997).

Segundo Zaina (2002), algumas das dificuldades no estabelecimento de critérios são a burocratização do processo avaliativo e a necessidade de desmembrar os aspectos a serem observados. E como benefício do uso de critérios na avaliação, a autora cita a redução da subjetividade, visto que, são estabelecidas as expectativas para as situações de aprendizagem. Bonniol (1981) acrescenta ainda que tanto os professores quanto os aprendizes atribuem notas melhores quando os critérios são explicitados.

3.1.4 Avaliação da Aprendizagem Quanto à Mediação

Para Barbosa (2008), a forma de encarar e realizar a avaliação reflete a atitude do professor em sua interação com os estudantes/classe, bem como suas relações com o estudante.

Neste sentido, a avaliação é vista como um processo orientador e interativo, servindo como diagnóstico dos avanços e dificuldades dos estudantes e como indicador para o re-planejamento do conteúdo, de práticas e comportamentos.

A avaliação mediadora, definida por Hoffmann (1994), é vista como um momento de reflexão e de desafio do professor que busca contribuir, esclarecer e favorecer a troca de idéia, entre e com seus aprendizes. De acordo com a autora, na ação avaliativa é tida uma das mediações pela qual se encoraja, por meio da comunicação e da reflexão, e se acompanha a reorganização do saber por meio de uma ação, movimento, provocação, na tentativa de reciprocidade intelectual entre os participantes. O acompanhamento se dá no sentido de favorecer o desenvolvimento do aprendiz, procurando desenvolver atividades que busquem sua evolução. Nessa perspectiva, o professor deixa de ter uma postura baseada na transmissão e verificação dos conhecimentos e assume a responsabilidade na produção e aprimoramento conjunto do saber (HOFFMANN, 2005).

A mediação no processo de construção de conhecimento implica em orientação no desenvolvimento de tarefas, oferecimento de novas leituras ou explicações, oportunizar sugestões e investigações, proporcionar vivências enriquecedoras e favorecedoras à sua ampliação do saber, possibilitar abertura a novas condutas de elaboração de esquemas de argumentação, contra-argumentação, para o enfrentamento de novas tarefas (HOFFMANN, 1994).

3.2 TÉCNICAS E INSTRUMENTOS DE AVALIAÇÃO DA APRENDIZAGEM

Por técnicas de avaliação, entende-se o método como se obtém as informações desejadas, e os instrumentos de avaliação são registros ou recursos usados para obter estas informações (GONÇALVES, 2006). As técnicas de avaliação da aprendizagem visam avaliar, por meio de instrumentos, o desenvolvimento do aprendiz quanto ao conhecimento e a compreensão dos conteúdos no processo de ensino e aprendizagem.

Existem diferentes técnicas e instrumentos para acompanhar a evolução da aprendizagem e podem ser classificadas em grupos de acordo com a forma como os dados são coletados para a avaliação. Declara Mediano (apud HAYDT, 2000), que quatro técnicas são utilizadas para coleta de informações: observação, inquirição, técnica sociométrica e testagem.

A observação consiste em olhar, ouvir, prestar atenção para obter, registrar e classificar informações sobre atitudes, comportamentos e habilidades cognitivas, afetivas e psicomotoras. Na inquirição os sujeitos envolvidos, podem ser entrevistados ou questionados na busca de informações desejadas. As técnicas sociométricas, definidas por Moreno (apud BASTIN, 1980), se caracterizam como um conjunto de procedimentos para analisar as relações informações intragrupais que se expressam por uma série de índices e esquemas gráficos, escalas de distância social (em que o sujeito se situa face aos seus colegas), listas de

participação (instrumentos para observar, analisar e caracterizar as intervenções de cada participante durante uma sessão grupal).

A testagem utiliza instrumentos (testes) para avaliar a exteriorização de um determinado comportamento ou habilidade, seja ele cognitivo, afetivo ou psicomotor (MELCHIOR, 1994). Gonçalves (2006) assegura que a testagem é a técnica com resultados mais eficientes e o teste é considerado o instrumento de avaliação com maior precisão nos dados coletados. Na literatura são identificados diferentes tipos de teste agrupados por sua finalidade: de aptidão, de atitudes, de maturação, de personalidade, de aproveitamento, de desempenho, entre outros. Os testes direcionados à avaliação da aprendizagem são chamados de testes de aproveitamento, de escolaridade, de conhecimento, de rendimento escolar.

É essencial compreender e diferenciar os termos testar, medir e avaliar. Para Haydt (1997, p.289), testar é “verificar um desempenho através de situações previamente organizadas, chamadas testes”; medir é “descrever um fenômeno do ponto de vista quantitativo”; e avaliar é “interpretar dados quantitativos e qualitativos para obter um parecer ou julgamento de valor, tendo por base padrões ou critérios”.

Dentro do processo de ensino e aprendizagem, as técnicas de observação, de inquirição e as sócio métricas são mais adequadas para avaliar o ajustamento do aprendiz em situações que envolvam relações sociais, bem como para detectar hábitos e aptidões operacionais. Já o uso de testes de avaliação de aprendizagem é especialmente recomendado para determinar se os objetivos cognitivos estabelecidos para o processo de ensino e aprendizagem estão sendo atingidos. Por ser parte fundamental do domínio de conhecimento do *framework* desta pesquisa, o instrumento de teste de avaliação de aprendizagem será abordado mais detalhadamente na seção a seguir.

3.3 TESTE DE AVALIAÇÃO DE APRENDIZAGEM (T2A)

O teste de avaliação de aprendizagem é um instrumento para avaliar a aquisição de informações e/ou o domínio de capacidades e habilidades resultantes do processo de ensino e aprendizagem (HAYDT, 2003). Destaca a autora, que os testes apresentam aspectos positivos (+ vantagens) e negativos (- desvantagens):

a) + avaliação de vários objetivos ao mesmo tempo, fornecendo uma ampla amostra do conhecimento;
b) + possibilidade de julgamento objetivo e rápido com correção direta;
c) + diminuição do aspecto subjetivo da correção;
d) + os resultados podem ser submetidos a tratamento estatístico;
e) - elaboração mais difícil e demorada;
f) - não avaliam as habilidades de expressão;
g) - restringem as respostas dos estudantes;
h) - facilitam a "cola".

Como características desses testes Adkins (apud COFFMAN, 1964) cita: 1) presença de uma grande variedade de estímulos aos quais o aprendiz deve responder; 2) aumento do número de respostas novas dadas na presença de estímulos já familiares ao aprendiz; e 3) aumento do número de novas respostas a serem dadas em situações que envolvam novos estímulos.

Ribeiro (1999) aponta três grandes finalidades para os testes de avaliação de aprendizagem:

a) Colaborar para o enriquecimento da aprendizagem com aumento da quantidade de aptidões adquiridas e do grau de proficiência do aprendiz;
b) Prover indicadores da situação de aprendizagem para propiciar maior qualidade e eficácia ao ensino;

c) Fornecer informações que, em conjunto com outras, possam dar subsídio para a avaliação da aprendizagem e para a tomada de decisões relativas à melhoria do processo de ensino e aprendizagem.

Munn; Gebhardt (1985) afirmam que os testes ensinam, pois os aprendizes não somente reestudam o material, como também avaliam e sintetizam em novos caminhos o que aprenderam. Sendo assim, o teste não é apenas uma forma de avaliar, mas pode também ser utilizado como estratégia de ensino.

Os testes de aprendizagem podem ser classificados, de acordo com o tipo de questões que o compõe, em: testes objetivos e testes subjetivos. Afirma Vianna (1981), que os testes objetivos são aqueles planejados e organizados com itens para os quais as respostas podem ser antecipadamente estabelecidas, ou seja, admitem uma única resposta como correta, e cuja pontuação não é afetada pela opinião ou julgamento dos examinadores. Este tipo de teste pode ser facilmente aplicado, corrigido e há possibilidade de *feedback* imediato. Outra possibilidade, em função de seu formato, é automatização do instrumento e do *feedback*. O teste dissertativo é um instrumento no qual o estudante elabora sua própria resposta. De acordo com Silva (2004), um teste dissertativo pode requerer do aprendiz desde o reconhecimento de informações específicas previamente aprendidas até a análise de um sistema de relações complexas, não estabelecidas previamente.

A utilização de testes como instrumento de avaliação da aprendizagem pressupõe atividades de desenvolvimento e aplicação do instrumento, análise e *feedback* dos resultados, que serão descritas a seguir.

3.3.1 Desenvolvimento e aplicação de Testes

O desenvolvimento do instrumento do tipo teste de aprendizagem deve levar em consideração o conteúdo e os objetivos de aprendizagem. O professor precisa conhecer

significativamente o conteúdo a ser avaliado, ter objetivos claros e definidos, conhecer as metodologias e técnicas de construção de testes, expressar as idéias de forma objetiva e usar linguagem adequada ao nível do aprendiz (HAYDT, 2000).

A elaboração do teste pode ser dividida em três etapas principais (ANASTASI; URBINA, 2000): 1) planejamento do teste, 2) redação dos itens e 3) análise dos itens. Para Ribeiro (1999) e Queiroz (2005), o planejamento é uma etapa de descrição da avaliação onde são identificados: os objetivos do teste, os critérios e fatores de avaliação, as habilidade trabalhadas, os resultados esperados e as prioridades da avaliação. A redação e análise dos itens consistem na tarefa de formulação e validação das questões. As autoras afirmam que é fundamental a seleção adequada do tipo de questão (detalhadas na seção seguinte), utilização de recursos apropriados, a observação do tempo para execução e correção, clareza e suficiência das instruções.

Quanto à aplicação dos instrumentos, pode ser considerado seu suporte, momento de aplicação ou acompanhamento. Quanto ao suporte, pode ocorrer no formato tradicional em papel ou disponibilizado de forma digital, seja *on-line* (*Web*) ou em outra mídia de armazenamento. O momento de aplicação do teste vincula-se ao modelo de avaliação a ser utilizado: no início do processo como avaliação diagnóstica, durante a formação como avaliação formativa ou ao final como avaliação somativa. Quanto ao acompanhamento a avaliação pode ser realizada de forma solitária pelo aprendiz ou mediada seja pelo professor ou por outro procedimento.

É importante o detalhamento dos tipos de questões que podem compor os testes classificados em questões objetivas ou dissertativas cujas características e particularidades de aplicação são abordadas abaixo.

3.3.2 Questões Objetivas

As questões objetivas se caracterizam por requererem respostas diretas, ou seja, a opinião do estudante não é expressa nessa resposta. Outra marca desse tipo de questões é quanto à análise, por serem respostas diretas, o professor não pode julgá-las como parcial. De acordo com Sant'Anna (1995) apud Medeiros uma avaliação é dita objetiva “quando a opinião do examinador e sua interpretação dos fatos não influem no seu julgamento”.

Pela sua essência objetiva, esse tipo de questão acaba se tornando algo difícil de ser elaborada, pois a preocupação maior do professor é quanto à clareza da pergunta e garantir que não haja respostas ambíguas.

A seguir serão apresentados os principais tipos de questões objetivas:

a) Pergunta-resposta ou resposta curta: onde há apenas uma resposta curta correta. Nesse tipo o estudante não é capaz de analisar criticamente, pois a resposta é direta. É empregada geralmente quando se deseja obter uma resposta que é basicamente a memorização de um fato ou conceito. Exemplo: (P) Qual a massa atômica do hidrogênio? (R): 1,00784 u

b) Múltipla-escolha: esse tipo é composto por uma pergunta seguida de possíveis respostas. Na sua maioria, pedem como resposta a melhor opção apresentada. As questões de múltipla-escolha auxiliam o professor a identificar dificuldades individuais por meio de questões erradas indicada pelo estudante. Sua elaboração é complexa, pois necessita que o professor crie respostas erradas plausíveis a fim de evitar que o estudante responda a questão por dedução. Exemplo: (P) Quem são os *stakeholders* de um projeto?

    a. (PR) a) São os usuários finais dos produtos do projeto.
    b. (PR) b) São os acionistas da organização hospedeira do projeto.
    c. (PR - R) **c)** São todas as pessoas que participam, influem e/ou são afetadas pelo projeto.

d. (PR) d) São os participantes do escritório de projetos que controlam os projetos de uma organização.

c) Associação, acasalamento, correlação ou combinação: são apresentados normalmente em duas colunas onde uma possui a palavra-chave e a outra coluna o contexto. Sua formulação é simples, porém seu uso é limitado uma vez que aborda situações que haja ideias em comum. A possibilidade de o estudante acertar esse tipo de questão ao acaso é remota, pois as combinações possíveis são inúmeras. Exemplo: Relacione as obras literárias com seus respectivos autores:

(a) Érico Veríssimo	a) ( ) O Dom Supremo
(b) Gabriel Garcia Marques	b) ( ) Dom Casmurro
(c) Paulo Coelho	c) ( ) O Tempo e o Vento
(d) Luís Fernando Veríssimo	d) ( ) Cem Anos de Solidão
(e) Machado de Assis	e) ( ) O Analista de Bagé

d) Verdadeiro-Falso, certo-errado ou resposta alternada: é de difícil elaboração, pois as afirmações devem ser absolutas a fim de evitar dupla interpretação. Exemplo: É verdadeiro afirmar que o ouro é um bom condutor de eletricidade.

( ) Verdadeiro ( ) Falso

e) Complete ou lacunas: esse tipo de questão é formada por frases que possuem espaços em branco no lugar de determinadas palavras. Sua elaboração é simples tendo em vista que visa comprovar conhecimentos absolutos. Assim como a pergunta-resposta, a possibilidade de acerto ao acaso é praticamente nula, pois está ligada diretamente a memorização. Exemplo: A velocidade de um corpo é calculada pela _____ dividida pelo _____.

A tabela a seguir sintetiza os tipos de questões objetivas quanto a sua utilização, preparação, memorização, acerto por sorte e limitações:

Tabela 2. Características dos tipos de questões objetivas.

Características	Tipo de questão objetiva			
	Múltipla-Escolha	**Associação**	**Lacunas e Respostas Curtas**	**Verdadeiro-Falso**
Pode ser utilizada	Nas mais diversas situações. Extremamente adaptável	Em temas em que as mesmas opções são respostas plausíveis para uma série de elementos	Em domínios de conhecimento muito específicos	Em situações onde é importante a verificação da veracidade de afirmações e conceitos
Sua preparação	Difícil, pois exige que as opções incorretas sejam plausíveis;	Simples	Trabalhosa, pois não deve haver a possibilidade de varias interpretações;	Difícil, pois exigem afirmativas ou negativas absolutas;
Memorização	Desencorajada devido ao número de alternativas	*pouquíssimo encorajada	Muito encorajada já que exige respostas absolutas	Dependendo da forma de construção das questões pode ser encorajada
Acerto por sorte	Reduzido pela diversidade de opções apresentadas	Reduzido, pois há muitas combinações possíveis	Desconsiderada	Grande, o que é balanceado aumentando o numero de questões
Limitações	Pedem tempo e habilidade na elaboração	Restringem-se a áreas onde há relação comum entre vários elementos	Restringem-se a noções específicas, podendo estimular a memorização de dados isolados	Restringem-se aos tópicos de muita certeza (escassez que se contorna fazendo-se questões bem específicas)

Fonte: Adaptado de Zaina L. A. M. (2002)

3.3.3 Questões Dissertativas

Sua principal característica é induzir o estudante a descrever e/ou opinar sobre determinado assunto pertinente ao conteúdo cobrado no teste. Por esse motivo a ocorrência de respostas distintas para a mesma questão, porém com o mesmo significado, é algo extremamente comum tendo em vista que o estudante possui a liberdade de organizar suas idéias da forma que melhor lhe convém.

Vianna (1987) afirma que a essência da questão discursiva é “verificar comportamentos mais complexos, como, por exemplo, a capacidade de explicar, descrever, interpretar, comparar, constatar, entre outras”. Com esse objetivo, esse tipo de questão é empregado em situações que o professor deseja ter conhecimento se o estudante tem

desenvolvido as habilidades como análise, síntese e avaliação, onde o julgamento crítico do estudante é primordial.

A elaboração da questão não é complexa como as questões objetivas que demandam maior cautela, contudo deve se preocupar com a possibilidade de dupla interpretação em relação ao que está sendo perguntado. A objetividade na pergunta é fundamental para que a correção seja simplificada.

A correção da questão discursiva é o problema maior na sua utilização, pois a correção leva em conta o julgamento do professor, seu conhecimento prévio e valores. Dessa forma, a mesma questão corrigida por um professor pode não ter o mesmo valor se corrigida por outro, já que a interpretação dos dois pode ser diferente.

3.4 INTERPRETAÇÕES DOS RESULTADOS DA AVALIAÇÃO

O próximo passo após a aplicação e correção de uma avaliação é a interpretação dos resultados pelo professor de modo a obter um retorno acerca do aprendizado do conteúdo ministrado. Para isso, há dois métodos de análise:

a) Por definição de critérios: basicamente o estudante é comparado com os critérios pré-estabelecidos nos objetivos educacionais.
b) Por normalização: refere-se à comparação de um estudante com outro.

As avaliações normativas apontam o grau de aprendizado do estudante sem detalhar o que realmente foi aprendido. É empregada geralmente em processos seletivos, pois seu principal objetivo é diferenciar a alta da baixa performance.

Quando o propósito é identificar os pontos em que o estudante possui dificuldade é recomendado que seja empregada a avaliações por critério. Esta é capaz de apontar as dificuldades que o estudante requer, um reforço antes de dar continuidade ao conteúdo. Popham (1978) enfatiza que a avaliação por critério indica informações de aprendizagem de

acordo com objetivos específicos, com isso o professor possui subsídios para analisar o desempenho do estudante além da comparação com os demais.

A nota atribuída à avaliação por critério, apenas posiciona o estudante em relação ao total de pontos propostos, e assim habilita-o a seguir no conteúdo, ou não, nos casos que usam a nota como pré-requisito. É com o auxilio dessa interpretação, que tanto o estudante quanto o professor, podem rever suas estratégias de ensino-aprendizagem para garantir a construção do conhecimento de forma efetiva.

3.5. FERRAMENTAS DE DESENVOLVIMENTO DE TESTES *ON-LINE*

Com a popularização do ensino a distância o número de ferramentas de desenvolvimento de testes *on-line* vem crescendo. Essas ferramentas visam auxiliar o professor na criação de testes, tanto para avaliações inteiramente *on-line*, quanto para exercícios de fixação e revisões. Outro beneficio encontrado nessas ferramentas é a praticidade que o professor encontra na hora de organizar os testes, utilizando muitas vezes avaliações de anos anteriores como exercícios para os estudantes.

Dessa forma, um dos objetivos ao se empregar uma ferramenta de desenvolvimento de testes *on-line*, é eliminar as dificuldades e desconforto que os professores sentem quando desejam fazer uma análise estatística dos testes. Outro seria o gerenciamento de questões e distribuição dos testes pela *Web*, de acordo com a requisição dos professores.

Dentre as ferramentas pesquisadas as principais funcionalidades encontradas foram a disponibilidade de tipos de questões na própria ferramenta, como por exemplo, questões de múltipla escolha, objetiva, associativa; módulo de autoavaliação que informa o retorno de forma imediata ao estudante; coleta de diversas informações adicionais que auxiliam o professor na análise do desempenho e dificuldades do estudante como, por

exemplo, o tempo empregado na resolução de uma ou um grupo de questões e geração dos testes de forma automática por meio de questões pré-cadastradas.

Um fator interessante encontrado nessa pesquisa foi a possibilidade do professor indicar o nível de dificuldade de cada questão, isso favorece na organização das questões pelo índice de dificuldade. Além disso, algumas ferramentas fornecem a opção de ordem randômica, tanto para questões quanto para a ordem das opções de respostas, resultando em uma prova praticamente exclusiva para cada estudante, há também a configuração dos testes que engloba pergunta, resposta e *feedback*.

Após a análise das ferramentas pode-se identificar uma estrutura comum onde se encontra o ambiente do professor e o ambiente do estudante. No primeiro, o professor realiza o cadastro das questões, cria os testes e os configura para as turmas correspondentes, podendo também consultar os resultados dos testes aplicados e acessar o rendimento de cada estudante gerando indicadores que facilitarão sua análise do desempenho.

No ambiente do estudante por sua vez fornece os testes cadastrados previamente pelo professor, onde o estudante soluciona a prova em um ambiente *Web*, podendo realizar essa tarefa independente do lugar. O estudante pode consultar provas anteriores, visualizar comentários do professor e receber *feedback* e nota assim que finalizar o teste. Tanto o ambiente do estudante quanto o do professor são restritos por usuário e senha que auxiliam na segurança dos dados, como nota e no sigilo da prova antes de ser aplicada.

A seguir apresentaremos de forma resumida, as ferramentas consultadas para a realização desse trabalho. Após isso apresentaremos um quadro comparativo entre as funcionalidades das mesmas.

3.5.1 AvalWeb

A ferramenta AvalWeb foi desenvolvida com base nas avaliações do Ensino a Distância por Cardoso (2001). Dessa forma a ferramenta oferece a opção para o professor criar questões e gerar avaliações automaticamente. Quanto aos acessos nos sistemas, por parte dos professores e estudantes, ocorre por meio da *Web*, e as informações ficam contidas em uma base de dados.

Essa ferramenta organiza as questões cadastradas por assunto, garantindo assim uma prova coerente com o conteúdo estudado pelo estudante. Há a opção para o professor informar o nível de dificuldade das questões fornecendo informações para que o sistema calcule o índice de dificuldade da avaliação. O sistema aceita a criação de vários tipos de questões onde múltipla escolha, verdadeiro/falso, relacionamento e completar lacunas estão inseridos.

Para o professor o sistema disponibiliza relatórios completos sobre o aproveitamento dos estudantes de acordo com as disciplinas avaliadas de forma estatísticas onde é possível extrair informações gerais sobre a avaliação em questão.

No entanto para o estudante o sistema oferece juntamente com a nota final da avaliação uma explicação dos professores sobre como resolver as questões da avaliação. Essa informação é de grande validade para o aprendizado efetivo do estudante.

3.5.2 QuestComp

A ferramenta QuestComp (ABRÃO, 2003) foi desenvolvida objetivando auxiliar a fixação do conteúdo por meio de testes e exercícios via *Web*. O funcionamento da ferramenta se dá principalmente no módulo *Engine QuestComp*, onde todas as validações (usuários) e tratamento de requisições são realizados. Após a validação do *login* o estudante

tem acesso às atividades disponibilizadas pelo professor, visualização do seu ranking e gabarito de questões anteriores.

O QuestComp é dividido em módulo do professor e do estudante. A seguir exibiremos as opções oferecidas em cada módulo.

Módulo Professor:

a) Elaboração de provas, exercícios e questionários;
b) inserção de perguntas (perguntas de múltipla escolha ou dissertativas.). O sistema é responsável pela ordem de apresentação das mesmas;
c) relatórios de desempenho.

Módulo Aluno:

a) responder questões;
b) consultar seu desempenho nas provas;
c) consultar os gabaritos.

3.5.3 Hot Potatoes

A ferramenta *Hot Potatoes* foi desenvolvida pelo Grupo de Pesquisa e Desenvolvimento do Centro de Computação e Multimídia da Universidade de Victoria no Canadá (versão 6.2, 2009). Ela oferece a possibilidade de criação de seis tipos de exercícios interativos para a *Web* sendo alguns desses em formato clicar-arrastar-soltar.

O fato de ser uma ferramenta desenvolvida na língua inglesa não restringe a mesma de aceitar caracteres com acentuação. Isso torna a ferramenta portável para qualquer idioma.

A ferramenta *Hot Potatoes* não exige um conhecimento em programação, pois o ambiente foi desenvolvido para que o usuário apenas preencha as informações como textos,

questões, respostas, e configure o aspecto das páginas como desejar. O programa fica responsável pela criação da página *Web* correspondente.

As informações que o usuário deve preencher para criar os exercícios são basicamente: título, mensagem de ajuda, mensagens de *feedback* após o estudante submeter uma questão, configurações de aparência da página, configuração da visualização das questões, temporizador para leitura do texto, entre outros.

A criação do exercício se da em três etapas. Primeiramente o professor insere as questões, respostas e *feedback*, após isso ele faz a configuração das instruções, botões de navegação entre outras, e por fim é feita a exportação dos exercícios para uma página *Web* que ficará disponível para o estudante acessar.

Uma funcionalidade oferecida pelo *Hot Potatoes*, que é muito interessante, é o envio das informações como: identificação do estudante, título do exercício, resultado, e o tempo de início e fim para o e-mail do estudante.

O software *Hot Potatoes* não se restringe apenas a criação de questões, ele possibilita a elaboração de um teste completo. Para isso ele oferece cinco aplicativos básicos que são:

a) JCross: palavras cruzadas;
b) JCloze: complete a lacuna;
c) JQuiz: respostas curtas e múltipla escolha;
d) JMix: ordenação de frases;
e) JMatch: associação de palavras.

Além desses há o Masher que permite agrupar os exercícios em matérias de uma unidade. Outra funcionalidade encontrada no *Hot Potatoes* é a inserção de figuras e *links* nos exercícios, tornando os mesmos mais interativos.

O *Hot Potatoes* exige o registro do programa para liberar o acesso ao sistema, entretanto é um software gratuito para uso em Instituições Educacionais sem fins lucrativos.

3.5.4 WebTest

O WebTest foi desenvolvido por Vilcachagua. et al (2001) e possui uma estrutura organizada de forma hierárquica divididas em cursos, módulos, assuntos e questões. Isso facilita na geração do testes, pois os mesmos são compostos por questões randômicas.

Os ambientes do professor e do estudante são totalmente separados. O professor possui o programa local chamado TestBuilder onde se faz o cadastramento de cursos, módulos, assuntos, matérias, estudantes e questões. Há também a versão *Web* (TestBuildirRemoto) desse ambiente que possibilita ao professor realizar as mesmas atividades independente de onde está.

No entanto, o estudante realiza os testes por meio do software TesteTaker. Esse ambiente permite que o estudante escolha entre receber a resposta correta, após submeter à resposta da questão, ou apenas quando o teste estiver finalizado.

Como citado anteriormente, os testes são gerados automaticamente de forma aleatória, pois consulta as questões em uma base de dados que corresponde ao conteúdo estudado. Isso dificulta a cola, já que, as questões aparecem em ordens diferentes a cada acesso.

Como a maioria dos sistemas *Web*, o WebTest exige um *login* e senha para realizar o acesso ao sistema. O WebTest também oferece ao professor o recurso de consultar o índice de acerto de cada estudante por meio de relatórios disponíveis no sistema.

3.5.5 Sisa-*Web*

A ferramenta Sisa-*Web* foi desenvolvida por Machado (2002) e possui duas interfaces diferenciadas: a primeira chamada Interface Administrativa onde os professores possuem acesso e a segunda Interface do Usuário que dà acesso aos estudantes.

A interface do Administrador é composta por cinco menus que facilitam a interação do professor remotamente; são eles:

a) Menu do usuário: onde ocorre o cadastro, alteração e remoção dos dados pessoais dos usuários como *login* e senha, que permitem o acesso ao sistema;
b) Menu de cursos: onde ocorre o cadastro, consulta, alteração e remoção dos cursos;
c) Menu de assunto: onde ocorre o cadastro, alteração e remoção dos assuntos dos cursos da instituição;
d) Menu de questões: onde ocorre o cadastro, alteração e remoção das questões que farão parte das avaliações. É nesse menu que são cadastradas imagens explicativas que compõe as questões;
e) Menu de provas: esse menu oferece os recursos necessários para a elaboração da avaliação onde esta é dividida em curso e assunto. Além disso, na opção relatórios, é possível visualizar o desempenho de cada estudante nas avaliações aplicadas.

Já a interface do Usuário é formada por apenas dois menus; são eles:

a) Menu de seleção de curso: neste, o estudante informa o curso que gostaria de participar e ser avaliado.
b) Menu de provas: onde o estudante realiza as avaliações e pode consultar gabarito, tanto das provas atuais, quanto das provas antigas.

3.5.6 Quiz

A ferramenta Quiz é um recurso do ambiente Learnloop (LearnLoop, 2007). que possibilita aos usuários e administradores do sistema criar provas/pesquisas de opinião. Para a montagem da mesma, devem-se realizar os seguintes passos:

a) Preencher um formulário para a criação da prova/pesquisa de opinião;

b) Criar as questões preenchendo o formulário correspondente e adicionar a prova/pesquisa de opinião. O formulário das questões exige o preenchimento dos campos para a pergunta e oferece até seis opções de respostas, onde se pode definir uma ou mais como correta. Outro fator interessante na elaboração da questão, é que o professor pode definir uma palavra chave que será associada a essa questão no campo categoria. Posteriormente, essa questão pode ser localizada pela informação na opção 'Procurar Questão'. Além disso, é possível associar dicas e sugestões a uma questão, no entanto, quando o estudante errar e solicitar esse recurso o peso da questão reduz em 50% do original;

c) Configurar o status da prova para que ela fique visível apenas quando desejado, definindo o dia ou período que a prova estará ativa, configurando a cor da tela onde as questões serão apresentadas no momento em que o estudante estiver respondendo, indicando também se as questões serão apresentadas em formato seqüencial ou randômico, e finalmente definir mensagens que o professor deseja mostrar para os estudantes que responderem a prova.

As configurações da prova/pesquisa de opinião podem ser acessadas sempre que desejado fazer alguma alteração como reativar, editar, remover, apagar ou adicionar questões. Assim como outras ferramentas o Quiz possibilita a consulta dos resultados por usuário, resultados por questão, percentual de execução e a cópia da prova para outra disciplina.

Como forma de segurança o ambiente de aprendizagem exige que o usuário faça o *login* impedindo assim, o acesso de pessoas não autorizadas. É por meio do *login* que o usuário possui seu perfil identificado e isso resulta na liberação ou restrição a determinadas funcionalidades.

Ao realizar a prova o estudante visualiza as questões de forma crescente ao clicar no botão responder. No caso de uma resposta errada, o estudante é informado na hora e pode optar em consultar a sugestão (previamente cadastrada pelo professor), se o estudante fizer essa escolha estará reduzindo em 50% a pontuação daquela questão. Quando a prova estiver finalizada, o estudante será informado da nota, media dos participantes e total de participantes. Ele pode optar em visualizar todas as questões que errou durante a prova, basta clicar na opção ver erros/todas.

3.6. COMPARATIVO DAS FERRAMENTAS DE DESENVOLVIMENTO DE TESTES *ON-LINE*

Após a análise das ferramentas descritas no item anterior, foi elaborada uma tabela elencando as principais características encontradas para facilitar a visualização das vantagens e desvantagens de cada um.

Os critérios apontados foram:

a) acesso *on-line*;
b) acesso *off-line*;
c) gerar teste automático;
d) agrupar as questões por assunto;
e) índice de dificuldade;
f) tipos fixos de questões;
g) relatórios de desempenho do estudante;

h) *feedback* a cada questão;

i) *feedback* final com a nota do estudante;

j) necessário *login*;

k) consulta de gabarito e provas anteriores;

l) aspecto da página configurável;

m) mediação na questão (ajuda);

n) mediação na prova (tempo de execução);

o) envio de resultados por e-mail;

p) associar mídias e *links* às questões;

q) questões randômicas;

r) perfil de usuários limitando o acesso.

Tabela 3. Comparação entre Sistemas Existentes.

	Sisa-*Web*	AvalWeb	WebTest	HotPotatoes	QuestComp	Quiz
Acesso *On-line*	X	X	X	X	X	X
Acesso *Off-line*			X			
Gerar teste automático		X	X			X
Agrupar as questões por assunto		X	X	X		X
Índice de dificuldade	X	X		X		
Tipos fixos de questões	X	X	X	X	X	X
Relatórios de desempenho do estudante	X	X	X		X	X
***Feedback* a cada questão**			X			X
***Feedback* final com a nota do estudante**	X	X	X	X	X	X
Necessário *login*	X	X	X	X	X	X
Consulta de gabarito e provas anteriores			X	X	X	X
Aspecto da pagina configurável				X		X
Mediação na questão (ajuda)				X		X
Mediação na prova (tempo de execução)				X		
Envio de resultados por e-mail				X		
Associar mídias e *links* às questões	X			X		X
Questões randômicas			X			X
Perfil de usuários limitando o acesso	X	X	X	X	X	X

Fonte: Adaptado de Justulin et al

4. MODELAGEM DE APLICAÇÕES *WEB* E A UWE

Dentro da Engenharia de Software, a atividade de Modelagem de Aplicações ou de Software consiste na construção de modelos gráficos, textuais ou matemáticos que expliquem as características ou o comportamento de um software. A abstração das aplicações computacionais, através de modelos que o descrevem, é um bom instrumento para o entendimento e comunicação do produto final, que será desenvolvido. Desta forma, os modelos representam uma simplificação da realidade podendo ser menos ou mais detalhados, de acordo com seu grau de abstração.

Cada sistema pode ser descrito sob diferentes aspectos, com a utilização de modelos distintos, sendo cada um deles uma abstração semanticamente específica da aplicação. O paradigma de engenharia que dá suporte ao desenvolvimento de aplicações *Web*[2] é denominado Engenharia de *Web* sites ou Engenharia *Web* (EW). A EW define um processo para construir aplicações hipermídia na Internet com qualidade e com base em critérios tais como: necessidades dos usuários, conteúdo de alta qualidade, atualizações constantes (manutenção), tempo de *download* mínimo, usabilidade e cultura corporativa centralizada na rede (ZAFIRIS, 2001).

A Engenharia *Web* se diferencia da Engenharia de Software por seu produto (*Web site*) e por apresentar requisitos próprios, como a necessidade de gerenciar um grande volume de informações, combinar a navegação controlada pelo leitor com a própria natureza das informações multimídia, atualizações constantes, tempo de *download* mínimo, usabilidade, cultura corporativa centralizada na rede, entre outros fatores. Sendo assim métodos, técnicas e ferramentas tradicionais de desenvolvimento de software, na maioria das vezes, são inadequados para o projeto aplicações *Web*.

Na construção do modelo, em específico para aplicações *Web*, são utilizados artefatos que simbolizam e identificam, de maneira abstrata: as características e funcionalidades que o software deverá prover (análise de requisitos); dão suporte ao planejamento de sua construção definindo a organização das informações e seus relacionamentos (modelo conceitual); apresentam a estruturação e fluxo de dados que indicam como as informações serão acessadas na aplicação (modelo de navegação) e definem como as informações e o acesso a essas informações serão apresentados ao usuário da aplicação (modelo de apresentação).

Diversas técnicas e formalismos foram criados para a realização da atividade de modelagem. Surgiram as Linguagens de Modelagem cujo "vocabulário e regras têm seu foco voltado para a representação conceitual e física de um sistema" (BOOCH; RUMBAUGH; JACOBSON, 1999). No final dos anos 90, as melhores técnicas foram reunidas num único formalismo que se tornou um padrão na modelagem de software, sendo criada a Linguagem de Modelagem Unificada (UML - *Unified Modelling Language*).

Porém, segundo Booch (2002), utilizar a UML em projetos *Web* não significa explorar todos os significados de classes, objetos, relacionamentos, fluxos, mensagens e outras entidades comuns da orientação a objetos, mas fazer uso dos recursos pertinentes a UML em busca de melhores projetos de *Web site.* As melhorias nos projetos podem ser vistas no uso de diagramas como diagramas de seqüência, diagramas de caso de uso, diagramas de classes, arquitetura da informação a partir da organização, navegação, rotulação e busca de conteúdo. O objetivo de utilização da UML em modelagem de aplicações *Web* é aplicar seu recurso de unificação, de

---

2 Aplicações Web são sistemas computacionais projetados para serem utilizados a partir de um navegador, na internet ou intranet.

linguagem comum entre desenvolvedores e administradores de *Web sites* no projeto, implementação e gerência de aplicações (BOOCH, 2000).

O problema ainda de representar aplicações *Web* com o uso de UML pode ser visto na insuficiência da expressão, na página *Web*, de *scripts* a serem executados no servidor e parte no cliente. Surge assim, a necessidade de modificar a UML de forma a torná-la flexível a partir da definição de mecanismos de extensão (CONALLEN, 1999):

a) Estereótipo: é um mecanismo de extensão da UML que permite a classificação de seus elementos originais de outra forma a partir da definição de restrições a esses elementos, ou seja, estendem o vocabulário da UML permitindo a criação de novos tipos de blocos de construção, derivados dos já existentes, mas específicos ao seu problema. São representados como uma seqüência de caracteres entre os símbolos << >>;
b) *Tag* valorado: estende as propriedades de um bloco de construção da UML, permitindo a criação de novas informações na especificação do elemento. Um valor com *tag* é representado em um diagrama como uma seqüência de caracteres delimitada por colchetes [];
c) Restrições: estendem a semântica de um bloco em construção da UML, permitindo adicionar novas regras ou modificar as existentes.

Existem diversos formalismos disponíveis para a modelagem de aplicações *Web* como: OOHDM – *Object Oriented Hipermídia Design Method* (SCHWABE; ROSSI, 1998; LIMA; SCHWABE, 2003; MOURA; SCHWABE, 2004), OOWS – *Object Oriented Web Solution* (FONS et al, 2001, 2003; PASTOR et al, 2001; PASTOR; INSFRAN, 1999; PELECHANO et al, 2003), WebML – *Web* Modeling Language (CERI et al, 2000, 2004), WAE – *Web Application Extension* (CONALLEN, 2003) e UWE – *UML Web based Engeenering* (KOCH,2001; KNAPP et al, 2004).

Apresentaremos a seguir dois formalismos estendidos de UML: WAE – *Web Application Extension* e UWE – *UML Web based Engeenering*. Estes formalismos utilizam a especificação padrão UML e estendem estereótipos para mapear estruturas típicas da arquitetura *Web*, como páginas, *applets*, componentes *servlets*, etc.

O estereótipo é um mecanismo de extensão da UML que permite a classificação de seus elementos originais de outra forma a partir da definição de restrições a esses elementos. A extensão, na forma de estereótipos, busca solucionar o problema da semântica de representação dos elementos presentes na modelagem de aplicações *Web*, através da modelagem conceitual (dados do sistema) e arquitetural (composição física). Estes estereótipos representam a composição física da aplicação a ser desenvolvida e o relacionamento dos seus componentes (modelo navegacional).

Os formalismos estudados para modelagem de aplicações *Web* (WAE e UWE) possuem etapas similares de modelagem conceitual, modelagem navegacional e modelagem de apresentação. Na modelagem conceitual são identificados os requisitos e conceitos aplicando, como artefatos, os casos de uso, diagramas de classes e de modelos comportamentais. Os modelos navegacionais inserem os elementos de ligação entre os contextos navegacionais e como entre os usuários que têm acesso a estes contextos expressados com estereótipos de páginas, de classes, de objetos navegacionais e relacionamento entre páginas. A modelagem de apresentação consiste na elaboração dos estilos gráficos e dos elementos que propiciam a interação da aplicação com o usuário podendo ser usado o artefato diagrama de composição da UML.

4.1 WAE - *WEB APPLICATION EXTENSION*

A WAE, proposta por Conallen (1999), estende a notação UML a partir de estereótipos com semântica e restrições adicionais que permitem a modelagem de elementos específicos da arquitetura envolvida em uma aplicação *Web*.

O objetivo principal desta extensão é modelar páginas *Web* expressando o relacionamento entre as mesmas, *links* e rotas de navegação, conteúdo dinâmico como *scripts* ao lado do cliente, e nas páginas ao lado do servidor, no nível apropriado de abstração e detalhe, permitindo o relacionamento entre os elementos específicos da *Web* e os demais elementos do sistema.

Exibiremos a seguir os mecanismos de extensão e as etapas modelagem WAE.

4.1.1 Mecanismos de Extensão WAE

Os mecanismos de extensão WAE são: estereótipos de páginas, relacionamentos entre páginas, *forms* e *framset*.

4.1.1.1 Estereótipo de Páginas

A página é um dos artefatos primários de uma aplicação *Web*. Na extensão WAE, uma página é representada no modelo através de duas classes distintas <<*server page*>> e <<*client page*>> de acordo com o local onde seus métodos são executados. Os métodos servidores e variáveis do escopo da página são contidos em na classe <<*server page*>> (figura 11) e representam os *scripts* do

servidor, sub-rotinas e funções. Não fazem parte da página servidor os *scripts* ao lado cliente e a formatação da interface gráfica. Uma página servidor pode ter relacionamentos com componentes existentes no servidor, como objetos de negócio, bancos de dados, sistemas externos e assim por diante. Os atributos da página cliente <<*client page*>> (figura 12) são variáveis do escopo da página e funções que executam no *browser* cliente. As páginas cliente são associadas com componentes como *Java Applets* e controles *ActiveX*.

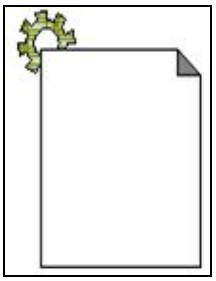

Figura 11. Server Page
Fonte: Conallen (1999)

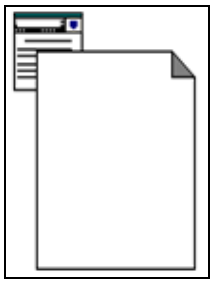

Figura 12. Client Page
Fonte: Conallen (1999)

4.1.1.2 Relacionamento entre páginas

Existe um relacionamento unidirecional intrínseco entre uma página cliente e uma página servidor indicando que uma página servidor geralmente é responsável pela construção de páginas cliente. Esse relacionamento é representado no modelo através de uma associação estereotipada por <<*build*>> (figura 13).

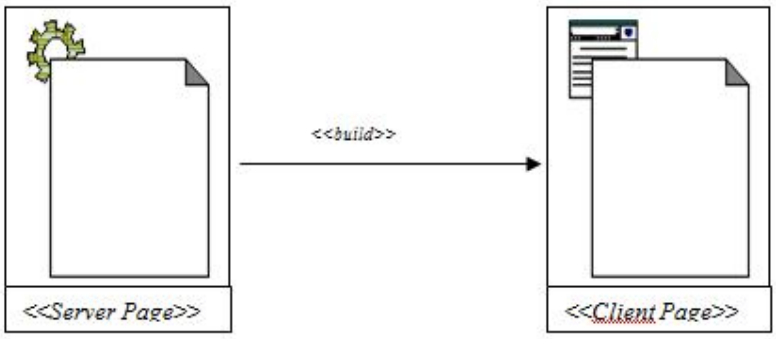

Figura 13. Construção de uma página cliente
Fonte: Conallen (1999)

Em algumas tecnologias de desenvolvimento *Web*, é possível uma página servidor redirecionar a requisição de um processamento para outra página servidor, esse relacionamento é representado como uma associação estereotipada por <<*redirect*>> (figura 14). O redirecionamento é uma característica muito útil, pois permite o reuso em aplicações *Web*.

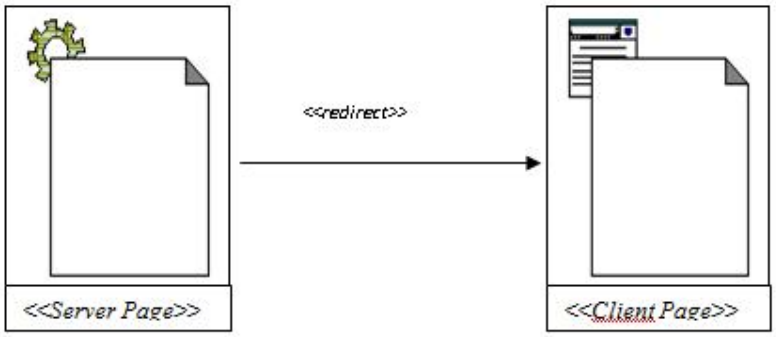

Figura 14. Redirecionamento de processamento entre páginas servidor
Fonte: Conallen (1999)

Um relacionamento adicional e importante é o *link*, estereotipado por <<*link*>>, utilizado para representar a associação entre páginas cliente e outras páginas *Web* – cliente. A figura 15 apresenta um exemplo de relacionamento entre páginas *Web*.

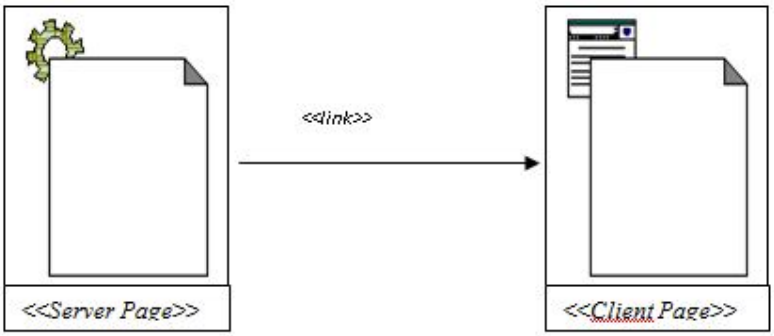

Figura 15. Associação <<link>> entre páginas cliente
Fonte: Conallen (1999)

4.1.1.3 *Forms* (Formulários)

Formulários (Figura 16) são expressos no modelo como uma agregação de páginas clientes. Um objeto «*form*» existe somente no contexto de páginas clientes. Este objeto é uma coleção de elementos de entrada padrão que aceita entradas do usuário e submete à página servidora para processamento. Em uma mesma página pode haver diversos *forms*, cada um possuindo uma ação diferente na página.

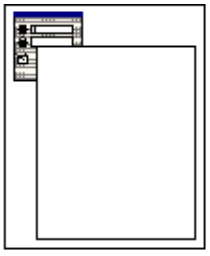

Figura 16. Formulário <<form>>
Fonte: Conallen (1999)

Uma classe <<*form*>> tem como atributos campos de entrada a serem preenchidos pelos usuários. Seus atributos representam: caixa de entrada, áreas de texto, botões de rádio, caixas de seleção e campos ocultos. Essa classe não possui operações, qualquer operação que interaja com o formulário seria uma propriedade da página que contenha o formulário. Métodos em uma página cliente têm acesso a

todos os atributos do *form* que estão associados página. Portanto, páginas clientes contêm *form*.

O relacionamento utilizado para representar qual página contém o formulário é a associação do tipo agregação entre as classes envolvidas. Outro relacionamento é necessário para identificar qual página *Web* processa os dados submetidos por um formulário, esse relacionamento é representado através de uma associação estereotipada <<*submit*>>. A Figura 17 abaixo mostra os relacionamentos citados.

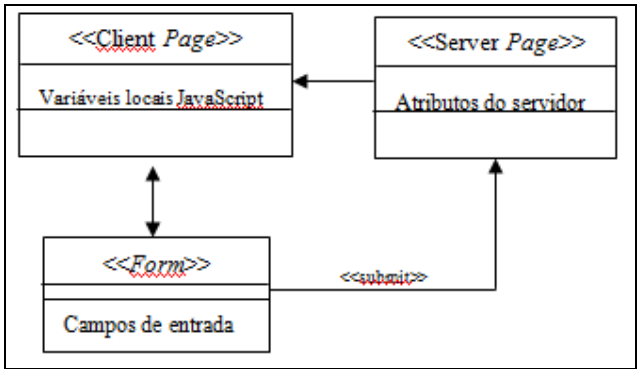

Figura 17. Uso de formulário em aplicações *Web*
Fonte: Conallen (1999)

4.1.1.4 Frameset

Frames ou quadros permitem que o projetista *Web* possa dividir a janela do navegador em sub-áreas retangulares (*panels*), cada uma com uma diferente página (Figura 18). Em aplicações *Web* os quadros geralmente são utilizados para dividir a janela em um painel de navegação e um painel de conteúdo. O painel de navegação exibe um índice de todas as páginas do *site*, e a cada vez que o usuário clica nesse painel, o painel de conteúdo é preenchido.

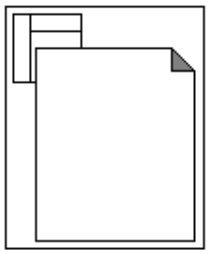

Figura 18. Quadro <<frameset>>
Fonte: Conallen (1999)

Coordenar atividades entre páginas em frames requer habilidade para referenciar páginas contendo *frames*. *Target* é o termo usado quando uma página cliente referencia outra página *Web* ou *frame*. Um *target* não tem propriedades ou atributos, ele é simplesmente uma referência possível para uma página cliente. Uma classe «*frameset*» pode conter um *target,* ou um *target* pode existir independentemente. Um estereótipo precisa ser definido para associação que indica que uma página cliente está requerendo um *link* para ser rodado em outra janela. Um estereótipo «*targeted link*» é aplicado para associação entre páginas clientes e *targets* com quem elas interagem. Parâmetros que são passados para o servidor com o *targeted link* podem ser identificados com um atributo *link* da UML.

Outros estereótipos de classes e associações podem ser vistos em Conallen (2003).

4.1.2 Etapas de Modelagem WAE

A modelagem WAE no desenvolvimento de uma aplicação *Web* é composta das seguintes etapas: elaboração do modelo conceitual representado pelos artefatos:

diagrama de casos de uso com os requisitos funcionais principais, diagrama de classes da UML identificando a estrutura dos dados e seus relacionamentos; representação do modelo navegacional utilizando os estereótipos de classes estendidas pela notação WAE através do relacionamento entre essas classes para representação navegacional e arquitetural da aplicação. Eventualmente, para a representação da iteratividade entre as classes do modelo navegacional pode ser utilizado o diagrama de sequência da UML.

4.2 UWE - *UML WEB BASED ENGEENERING*

A UWE é um projeto desenvolvido na Ludwig-Maximilians-Universität München[3] na Alemanha, cujo grupo de pesquisa é composto por seis membros, estando entre eles, Nora Koch[4]. De acordo com o site oficial, a UWE é definido como:

> UWE uses "pure" UML notation and UML diagram types whenever possible for the analysis and design of *Web* applications, i.e. without extensions of any type. For the *Web* specific features, such as nodes and *links* of the hypertext structure, the UWE profile includes stereotypes, tagged values and constraints defined for the modelling elements. The UWE extension covers navigation, presentation, business processes and adaptation aspects. The UWE notation is defined as a "lightweight" extension of the UML. (LMU, 2009)

Frente à definição apresentada, pode-se destacar que a UWE é basicamente uma extensão da UML que visa disponibilizar recursos para a modelagem de aplicações *Web* focando os conceitos de usabilidade e de arquitetura de informação. Seu desenvolvimento se baseou na metodologia orientada a objetos e iterativa, fundamentada em padrões de processos de desenvolvimento unificado de software.

---

[3] Ludwig-Maxililians-Universitat München: Universidade situada em Munique na Alemanha

[4] Nora Koch: pesquisadora do Departamento de programação e Engenharia de Software no Instituto de Ciência da computação na Universidade Ludwig-Maximilians em Munique. Responsável pelo inicio das publicações sobre UWE no ano de 2000.

De acordo com Koch (2000), o processo de Engenharia de *Web* baseado na linguagem UML dá suporte para o desenvolvimento de aplicações *Web* com foco na sistematização e personalização. É uma abordagem orientada a objetos baseado em dois princípios dimensionais do Processo Unificado (*Unified Process*): tempo e conteúdo.

Os princípios dessa abordagem são: o uso de uma notação padrão para todos os modelos (UML), a definição do método e a especificação de restrições.

O UWE recomenda o uso de restrições definidas pela OCL - linguagem de expressões para especificar restrições sobre modelos orientados a objetos ou outros artefatos da linguagem UML.

O formalismo UWE aborda as três dimensões de um projeto de aplicações *Web*: definição do conteúdo, estruturação da navegação e requisitos de apresentação, utilizando elementos padrões da UML juntamente com a notação UWE.

Baumeister (1999) e Hennicker (2000) propõem o emprego da UML ao longo de todo o processo de modulação da UWE. Versões mais recentes apresentadas por Koch (2002) e Hennicker (2001), disponibilizam especificações focadas em transição entre modelos e interligação entre os conceitos, conhecidos como estereótipos, incorporados nos modelos, ou meta-modelos como são chamados na UWE (figura 19).

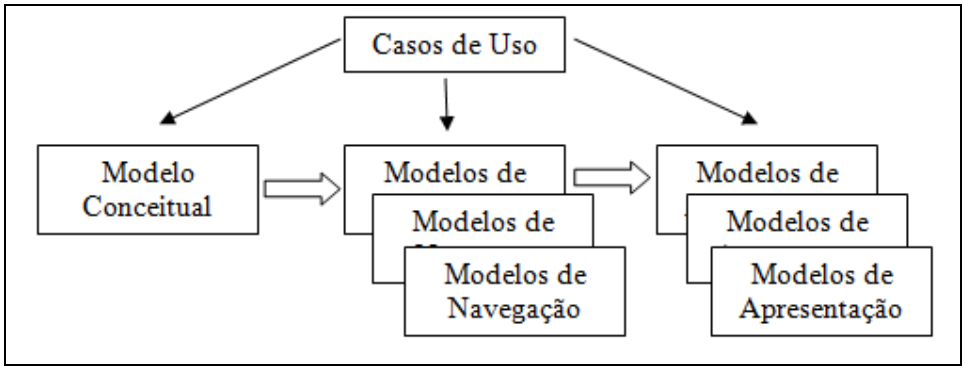

Figura 19. Modelos de concepção de um Sistema *Web*
Fonte: Adaptado de LOPES, R. et al (2003)

4.2.1 Mecanismos de Extensão UWE

A notação utilizada para o desenvolvimento de aplicações *Web* é a UML e uma pequena extensão, chamada perfil (*profile)* desenvolvida pelos próprios autores do formalismo. O perfil UML inclui estereótipos para a modelagem de características de navegação e apresentação de aplicações *Web*.

Os elementos que possuem extensão UWE estão presentes no modelo navegacional e no modelo de apresentação (KOCH et al, 2000)

4.2.1.1 Modelo Navegacional

a) Classe Navegacional**:** representam uma classe conceitual, cujas instâncias poderão ser visitadas por usuários durante a navegação. O ícone usado para o estereótipo «*navigational class*» é mostrado na Figura 20.

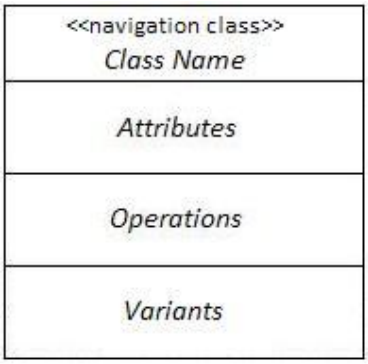

Figura 20. Classe de Navegação
Fonte: Koch et al (2000)

b) Navegabilidade Direcionada: associação no modelo navegacional (Figura 21) é interpretada como navegabilidade direcionada da classe navegacional de origem para a classe navegacional de destino. Possui semântica diferente de uma associação no modelo conceitual. Ela determina a direção de navegação entre as classes, e são exibidas por setas direcionadas ou bidirecionadas (navegação em ambos os sentidos).

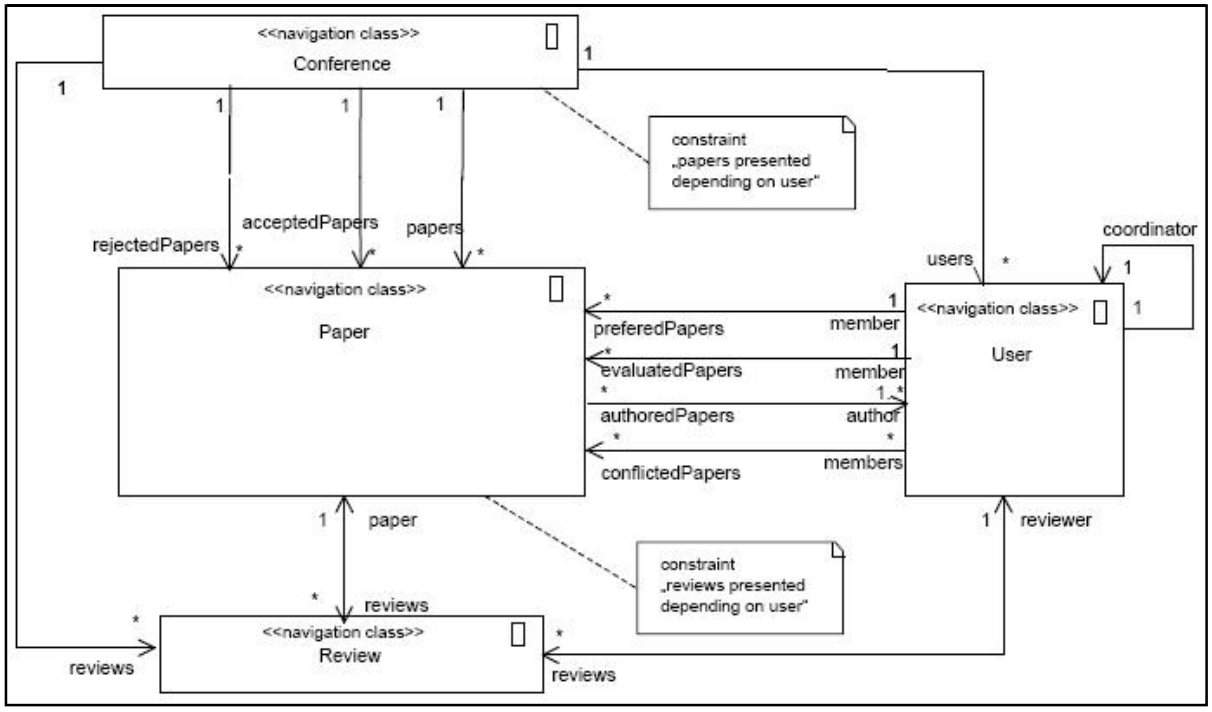

Figura 21. Modelo de Navegação

Fonte: Koch et al (2000)

c) *Index*: é modelado por um objeto composto que contém um número arbitrário de itens de *index*. Cada item de *index* possui um nome e um link para uma instância de uma classe navegacional. Qualquer *index* é um membro de alguma classe *index* que é estereotipada por «*index*» com um ícone correspondente como mostrado na Figura 22.

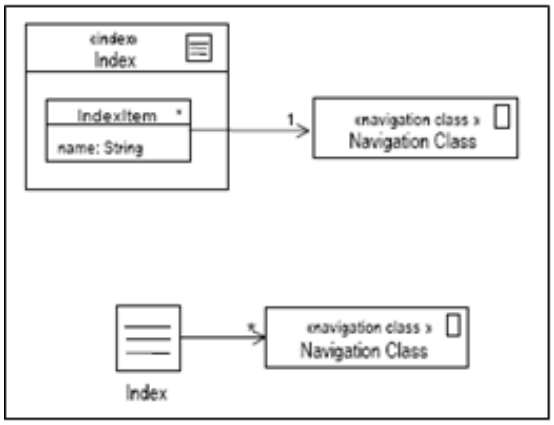

Figura 22. Índice

Fonte: Koch et al (2000)

d) *Guide Tour*: é um objeto que fornece acesso seqüencial para as instâncias de uma classe navegacional. O ícone correspondente para o estereótipo «*guide Tour*» é mostrado na Figura 23. Toda classe *guide tour* será conectada a uma classe navegacional por uma associação direcionada que tenha a propriedade {*ordered*}.

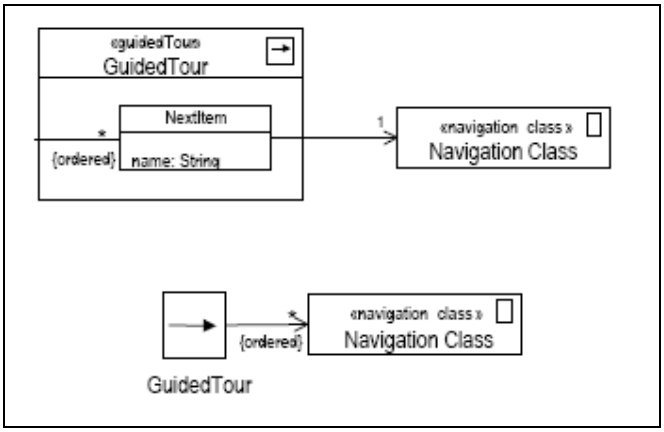

Figura 23. Classe *Guide Tour* e ícone correspondente
Fonte: Koch et. al (2000)

e) *Query*: é representado por um objeto que tenha uma string de consulta como atributo. O ícone para o estereótipo «*query*» é mostrado na Figura 24.

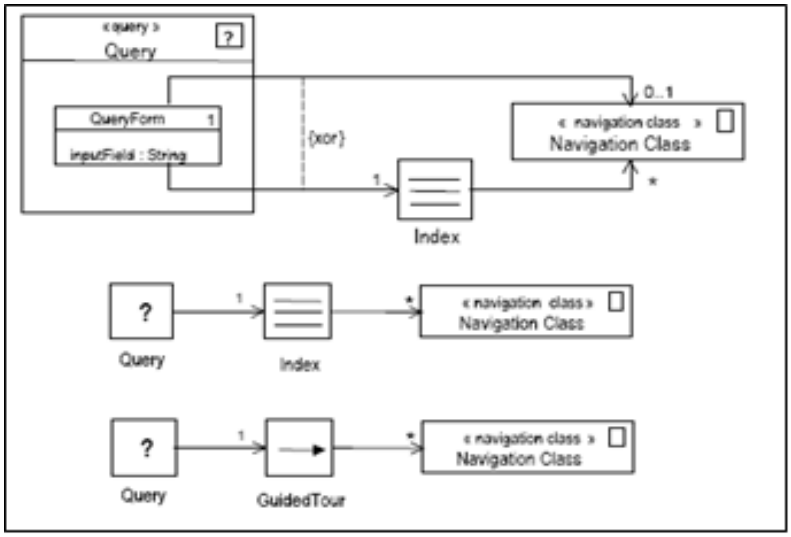

Figura 24. Classe *Query* e ícone correspondente
Fonte: Koch et al (2000)

f) Menu**:** é um objeto composto que contém um número fixo de itens. Cada item de menu tem um nome e um *link* para a instância de uma classe navegacional, *index*, *guide tour* ou *query*. Qualquer menu é uma instância de uma classe menu que é estereotipada, por «*menu*» com um ícone correspondente mostrado na Figura 25.

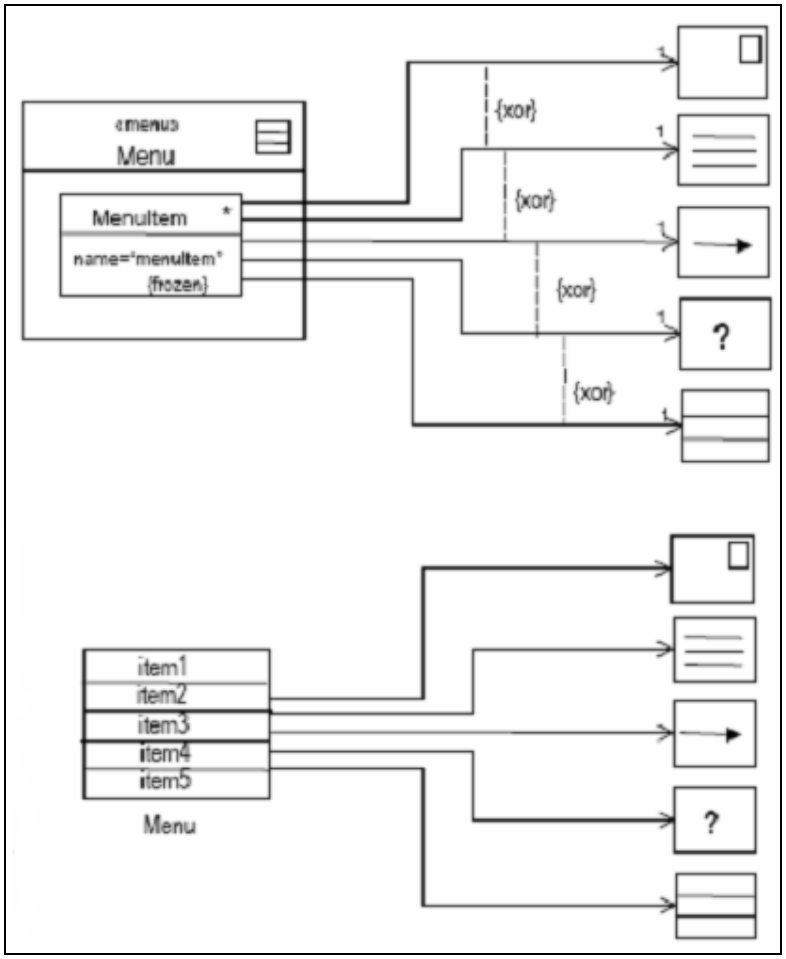

Figura 25. Classe Menu e ícone correspondente

Fonte: Koch et al (2000)

4.2.1.2 Modelo de Apresentação

a) Classe de apresentação: modela a apresentação de uma classe navegacional ou primitiva de acesso, como *index*, *guide tour*, *query* ou menu. Instâncias de classes de apresentação recebem elementos de modelagem como texto, imagens, vídeos, áudio, âncoras, coleções, coleções ancoradas, etc. A Figura 26 mostra o ícone correspondente.

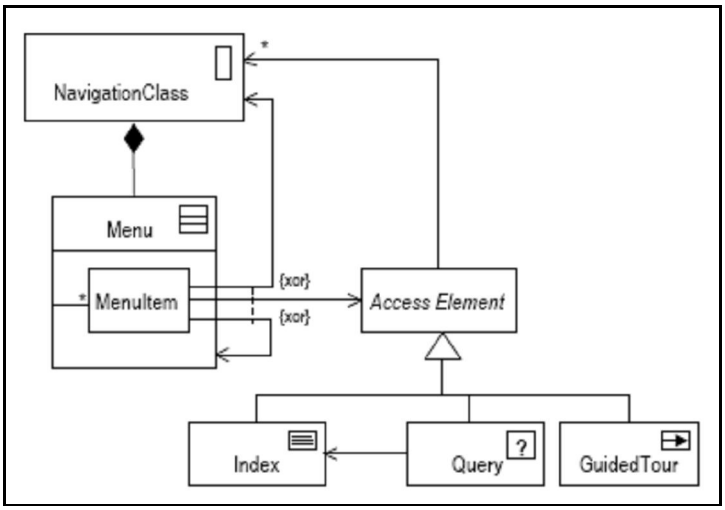

Figura 26. Padrão para estrutura de acesso
Fonte: Koch et al (2000)

b) Visão de Interface com o usuário: especificam que cada instância de uma classe contém todos os elementos abstratos apresentados. É representado por <<UI view>> e seu ícone correspondente (Figura 27).

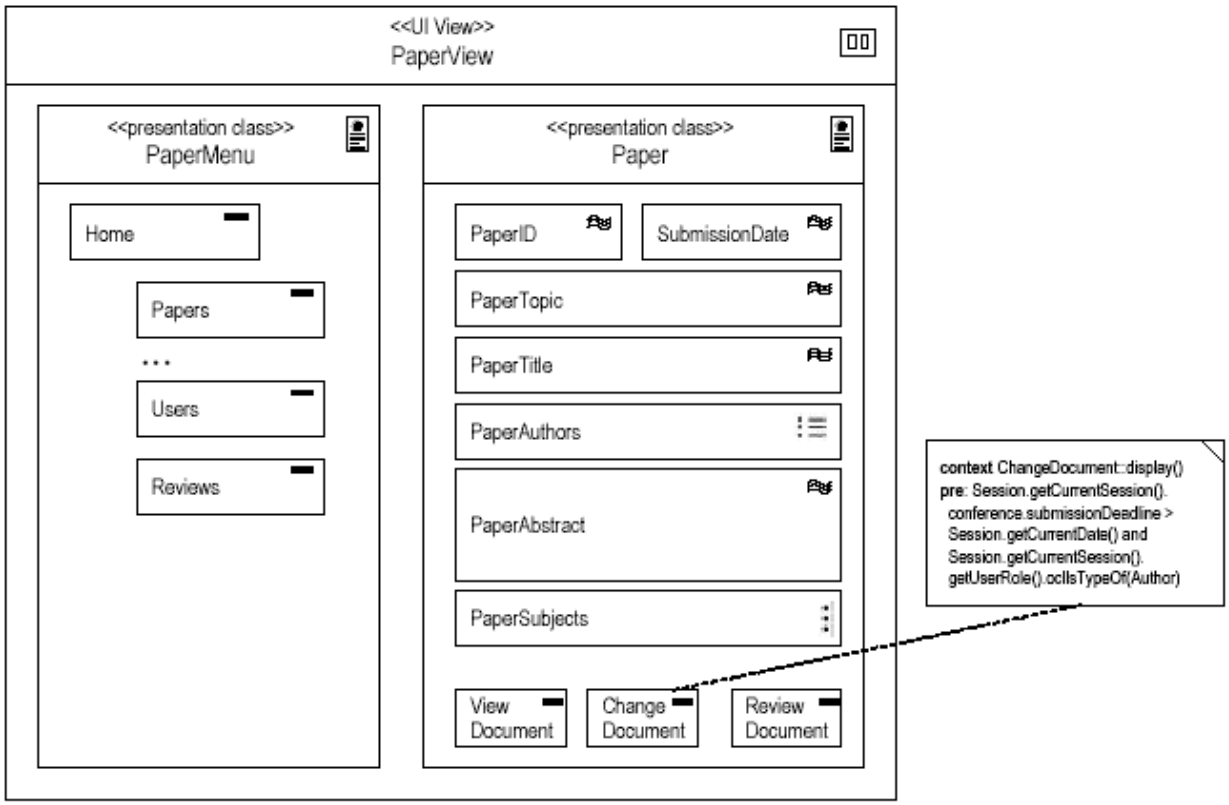

Figura 27. Visão de interface com usuário e classes de apresentação

Fonte: Koch et al (2000)

c) Elementos de interface com usuário: classe abstrata com especializações que descrevem elementos de interface particulares (Figura 28).

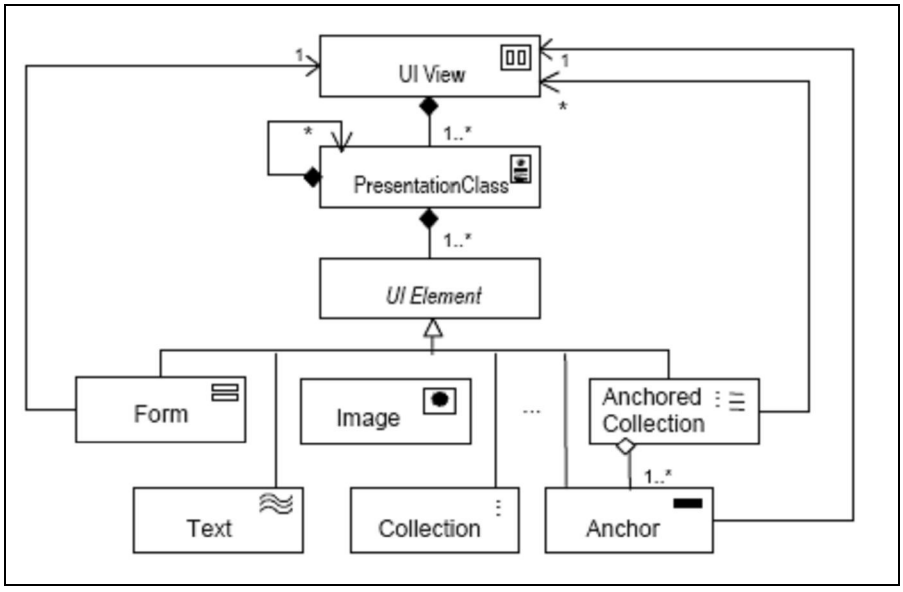

Figura 28. Exemplo de Elementos de interface com usuário

Fonte: Koch et. al (2000)

d) Frameset e Frame**:** é um elemento de alto nível que é modelado por uma composição de objetos de apresentação ou outros *framesets*. Uma área de um *frameset* é associada a cada elemento de baixo nível, chamado frame. Os ícones para os estereótipos «*frameset*» e «*frame*» são apresentados na Figura 29.

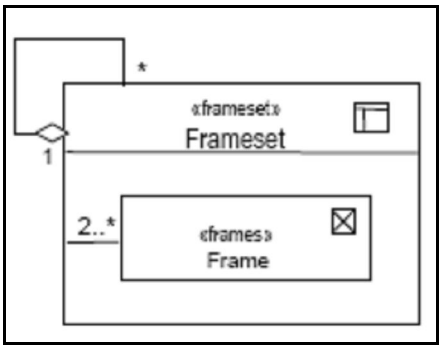

Figura 29. Padrão para estrutura de acesso

Fonte: Koch et. al (2000)

e) Janela - *Window*: é a área de interface de usuário onde *framesets* e objetos de apresentação são exibidos. Uma janela pode ser movida, maximizada ou minimizada a um ícone. Ela possui dois pequenos botões, um para transformar a janela em ícone (minimizar) e outro para fechar a janela. A Figura 30 apresenta o estereótipo de classe para janela.

Figura 30. Padrão para estrutura de acesso
Fonte: Koch et. al (2000)

f) Fluxo de Apresentação: modela as dinâmicas da apresentação mostrando onde os objetos de navegação e elementos de acesso vão ser apresentados para o usuário (em qual *frame* ou janela o conteúdo é mostrado e qual conteúdo será substituído quando um *link* é ativado).

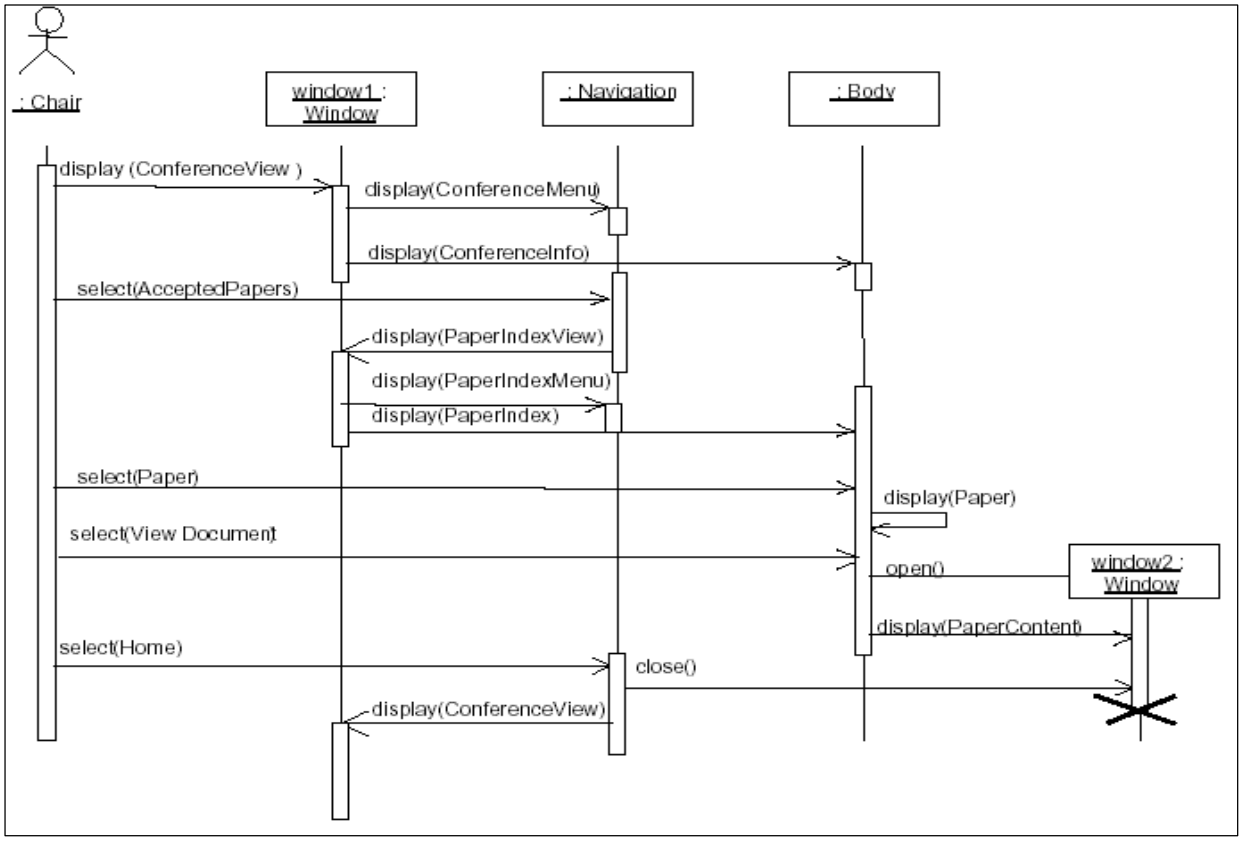

Figura 31. Padrão para estrutura de acesso

Fonte: Koch et. al (2000)

A Figura 32 mostra os ícones escolhidos para os estereótipos de elementos de modelagem apresentados abaixo. Eles são usados no projeto de apresentação de aplicações *Web* em adição aos objetos de apresentação, janelas, *frameset* e *frames*.

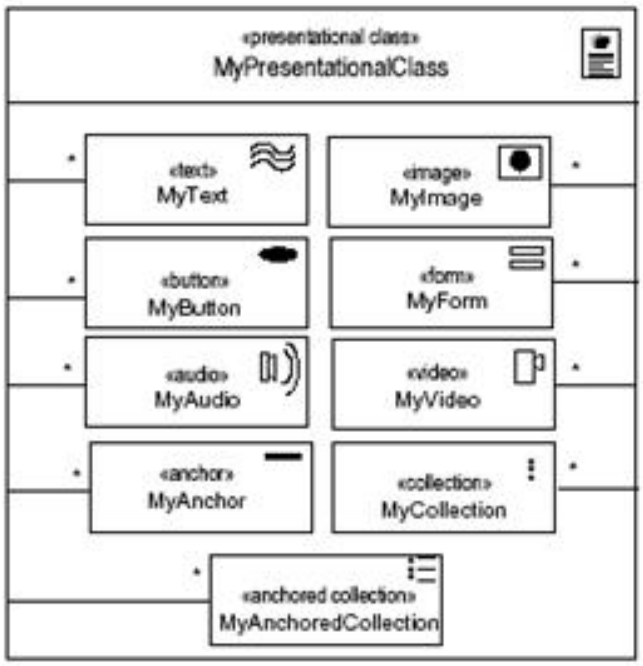

Figura 32. Estereótipos de apresentação

Fonte: Koch et. al (2000)

- **texto:** é uma seqüência de caracteres;

- **âncora:** é um texto que pode ser clicado, que é o ponto de partida para o relacionamento entre outros nós;

- **botão:** é uma área que pode ser clicada que tem uma ação associada a ela;

- **imagem, vídeo, áudio:** são objetos multimídia. uma imagem pode ser exibida. áudio e vídeo podem ser iniciados, parados, podemos voltá-los, ou adiantá-los;

- **form:** é usado para receber informações do usuário que passam informações em um ou mais campos de entrada ou opções de seleção do *browser* ou checkbox;

- **coleção:** é uma lista de elementos de texto que é introduzida como estereótipo que provem uma representação conveniente desta composição;

- **coleção ancorada:** é uma lista de âncoras que é introduzida como estereótipo que provem uma representação conveniente de uma coleção de âncoras.

4.2.2 Etapas de Modelagem UWE

A metodologia proposta pela UWE detalha e organiza as fases de análise e concepção de um projeto a fim de decompô-la nos seguintes modelos apresentados abaixo (KOCH, 2000 et al; ABREU, 2007):

a) Análise de requisitos: tem como objetivo encontrar os requisitos funcionais da aplicação *Web* e representar esses requisitos na forma de casos de uso;

b) Projeto conceitual: fundamentado no modelo de casos de uso, geralmente emprega a orientação a objetos que ilustram a interação do usuário com a aplicação. Resulta no Modelo conceitual representado pelo diagrama de classes UML;

c) Navegação: ilustra a navegação dentro da aplicação proposta, faz uso de índices, *guided tours*, *queries* e menus na sua construção. Seu resultado é o Modelo de Navegação Espacial e Estrutural representado também pelo diagrama de classes UML e entendido por estereótipos.

d) Apresentação: constituído por interfaces abstratas que indicam a interação do usuário com a aplicação *Web*, que representa o Modelo de apresentação expressado pelo diagrama de seqüência UML e entendido por estereótipos.

4.2.2.1 Etapa 1 UWE: Análise de requisitos

Na primeira etapa da modelagem de um sistema é elaborado uma análise formada por uma coleção de artefatos como: documentos, registros de banco de dados, modelos. Esse levantamento visa evitar ambiguidade no software que será implementado.

Conallen (p.176, 2003) registra que “um requisito é uma restrição que o sistema deve observar e normalmente é expresso como uma declaração que começa com uma frase semelhante a ‘O sistema deverá...”. Diante disso, a análise de requisito tem a finalidade de expressar um comportamento ou uma característica em uma linguagem clara e autoexplicativa. Conallen (p.183, 2003) aponta três pontos fundamentais na elaboração de uma análise, são eles:

a) Clareza e concisão na descrição de cada requisito;

b) Deve ser focada em apenas um ponto garantindo uma granularidade mais fina.

c) Deve ser verificável, ou seja, poderá ser testado após sua implementação.

Em um software podem-se classificar os requisitos funcionais e os não funcionais. O primeiro refere-se à ação que o sistema deve executar, geralmente trata-se da entrada e saída. Já o segundo é definido a partir da qualidade do sistema, estes podem ser sub-classificados em: usabilidade, desempenho, robustez/confiabilidade, segurança, hardware, implantação e outros (CONALLEN, 2003).

Para a elaboração da análise de requisitos é de suma importância que a coleta seja realizada por um grupo de pessoas, já que, uma terá uma visão diferenciada da outra e isso proporciona uma análise mais detalhada. Sendo assim, a equipe de requisitos é responsável por desenvolver os artefatos e a equipe de gerenciamento de projetos por estabelecer a prioridade de cada um.

O documento, que é o resultado dessa análise, não possui uma metodologia para ser desenvolvido, deve-se prezar por dispor cada requisito em um marcador com a finalidade de rastreamento.

O fragmento a seguir é um exemplo (CONALLEN, p.181, 2003):

> **3. Requisitos de desempenho**
>
> Essa seção descreve os requisitos de desempenho do sistema. Esses requisitos normalmente relacionam-se com a velocidade de execução e a capacidade de cada componente do sistema.
>
> **3.1 Desempenho do servidor *Web***
>
> A seção do desempenho do servidor *Web* descreve o desempenho esperado do servidor *Web* e da rede do sistema.
>
> 3.1.1 Cada servidor *Web* no sistema deverá ser capaz de manipular pelo menos 150 sessões de usuários simultâneas.
>
> 3.1.2 O sistema não deverá exigir mais de três segundos para recuperar e responder à solicitação de um cliente para uma página *Web* estática.
>
> 3.1.3 O sistema não deverá exigir mais de oito segundos para responder a uma página dinâmica.

Um modelo comumente utilizado na análise de requisitos é o modelo de casos de uso. É ele quem dará origem aos modelos: Conceitual, Navegacional e de Apresentação que é a característica marcante de aplicações *Web*.

Nos modelos de Casos de Uso são definidos os usuários da aplicação, nominados atores, e representação das funcionalidades que cada ator interage no sistema. Para a criação desse modelo faz-se uso de atores e casos de uso onde estes podem estar ligados por relacionamentos como inclusão, extensão e herança (ABREU; LARA; RAGAZZO, 2007). Os casos de uso são estruturados no padrão UML como o exemplo a seguir:

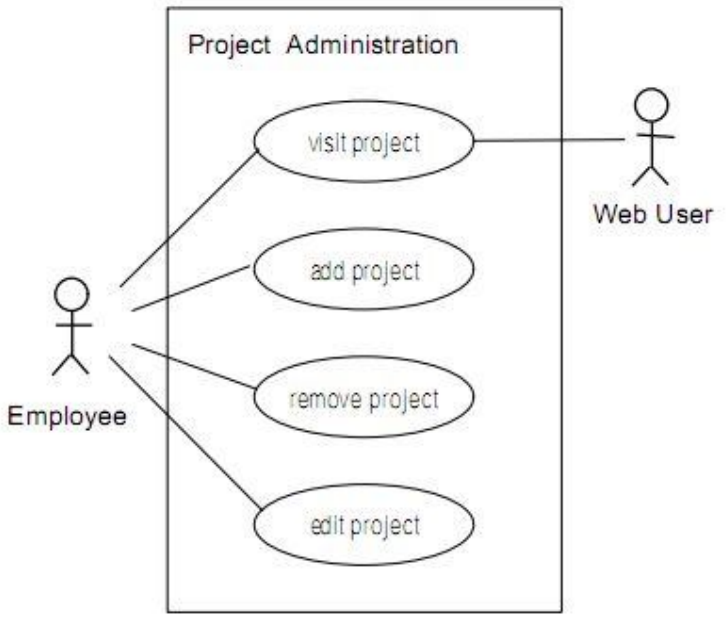

Figura 33. Exemplo de Modelo de Caso de Uso

Fonte: Koch, N.; Kraus, A. (2002)

Segundo Abreu, Lara e Ragazzo (2007), a elaboração dos casos de uso se dá mediante a execução dos seguintes passos:

> a) encontrar os atores. Para cada ator, procurar o texto com as atividades que cada um deles irá executar. b) agrupar atividades para os casos de uso. c) estabelecer as relações entre os atores e os casos de uso. d) simplificar o modelo de casos de uso acrescentando o relacionamento herança entre atores e/ou entre casos de uso.

Pode-se ainda associar uma descrição para cada caso de uso a fim de identificar o cenário que ele se encontra.

4.2.2.2. Etapa 2: Projeto Conceitual

A etapa anterior, ou seja, a análise de requisitos é quem fomenta o projeto conceitual, este é composto pelos objetos que participam na interação entre o usuário e a aplicação, que são descritos no caso de uso. É nessa etapa que o diagrama de classes é construído.

Para a construção do diagrama de classes empregam-se basicamente os elementos UML:

a) Classe: formadas por um nome, atributos e operações;

b) Associações: ligação entre as classes;

c) Pacotes: agrupamento de classes.

Um exemplo de diagrama de classes é a figura 34:

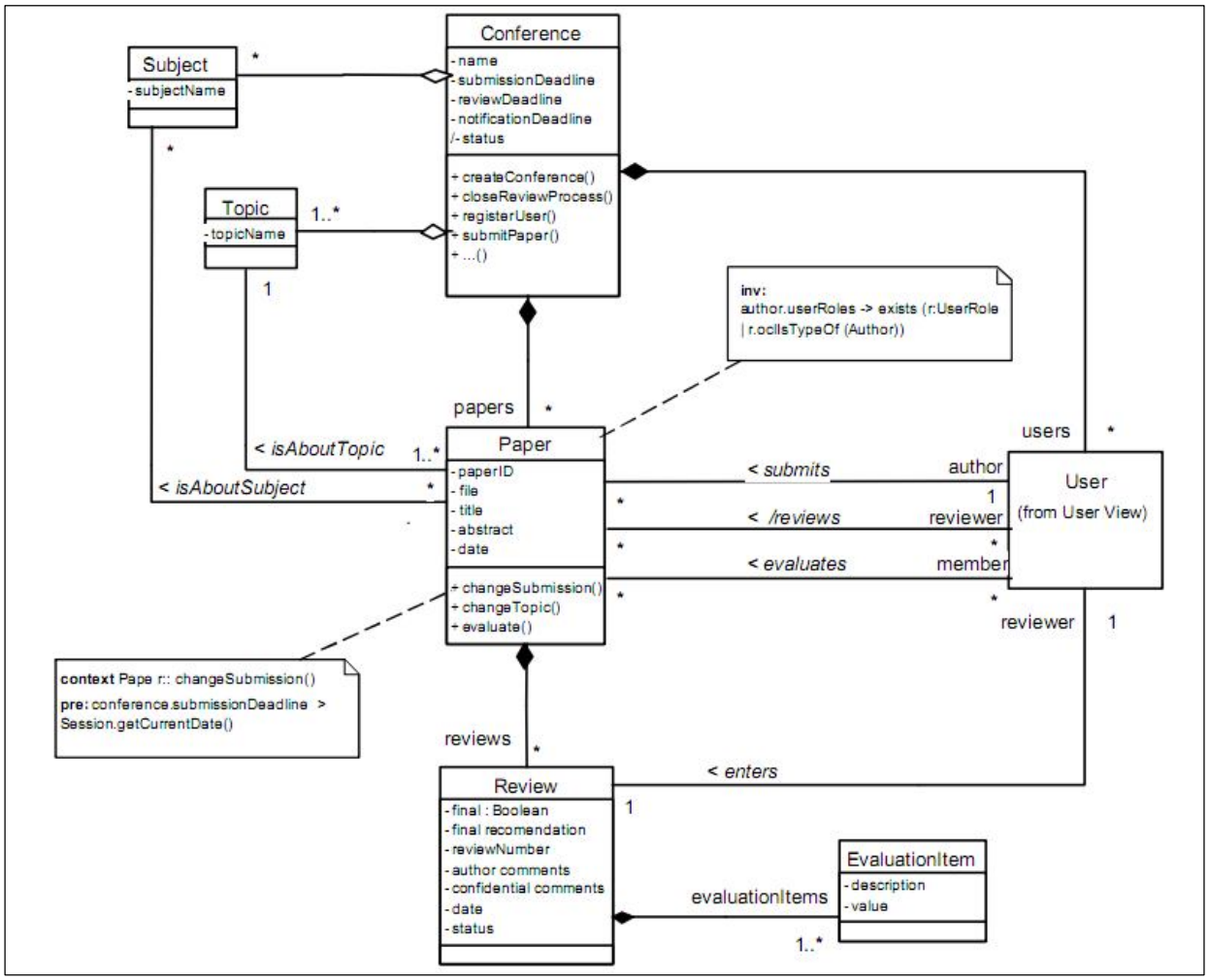

Figura 34. Diagrama de Classes de uma conferência.

Fonte: Koch, N.; Kraus, A.; Hennicker, R. (2001)

Abreu, Lara e Ragazzo (2007) definem os passos para a elaboração do modelo conceitual da seguinte forma:

> [...] encontrar as classes referentes à aplicação em questão e especificar os atributos e operações mais relevantes de cada classe. Determinar, então, associações entre classes. Agregar classes e identificar composições de classes. Definir heranças e hierarquia. Por fim, definir restrições do modelo conceitual.

O diagrama de classes desenvolvido no modelo conceitual será a base para a elaboração do modelo de navegação.

4.2.2.3 Etapa 3: NAVEGAÇÃO

O projeto de navegação é de grande utilidade para a documentação da estrutura da aplicação e para o aumento estrutural da navegabilidade, sendo assim, pode-se afirmar que é um processo complexo na modelagem de uma aplicação *Web*.

O modelo de navegação é constituído por dois sub-modelos que podem ser descritos como “views” de forma análoga a um banco de dados. Com isso, objetiva-se relacionar as classes de navegação às classes conceituais e seus respectivos atributos.

Neste sentido, a UWE distingue o modelo de navegação em espacial e estrutural a fim de detalhá-lo ao máximo, conforme apresenta Abreu, Lara e Ragazzo (2007):

a) Navegação espacial

O modelo de navegação espacial é fundamentando em certas decisões que o projetista deve ater sua atenção, Abreu, Lara e Ragazzo (2007) destacam as seguintes:

- as visões do modelo conceitual fundamentais para a aplicação;
- as tomadas de decisão ao longo da aplicação que garantam a funcionalidade da mesma.

Todas essas escolhas estabelecidas pelo projetista são frutos do modelo conceitual e dos requisitos da aplicação que estão ilustrados nos casos de uso. Dessa forma a modelagem do espaço de navegação deve possuir uma especificação detalhada das associações existentes, da sua multiplicidade incluindo nomes completos garantindo uma possível geração semiautomatica do modelo estrutural de navegação.

Para a elaboração desse modelo são empregados basicamente dois elementos que indicam navegação direta:

- navigation classes (Páginas): possui o mesmo nome da classe do modelo conceitual e é responsável pela modelagem de uma classe que terá sua instância visitada pelo usuário no decorrer da navegação;

- navigation associations (*Links*): são basicamente a direção a ser seguida durante a navegação em uma aplicação, sendo assim, são em sua maioria representadas por setas bi-direcionadas.

Como exemplo de espaço de navegação tem-se a Figura 35 que representa a navegação de um sistema de revisão de uma conferência:

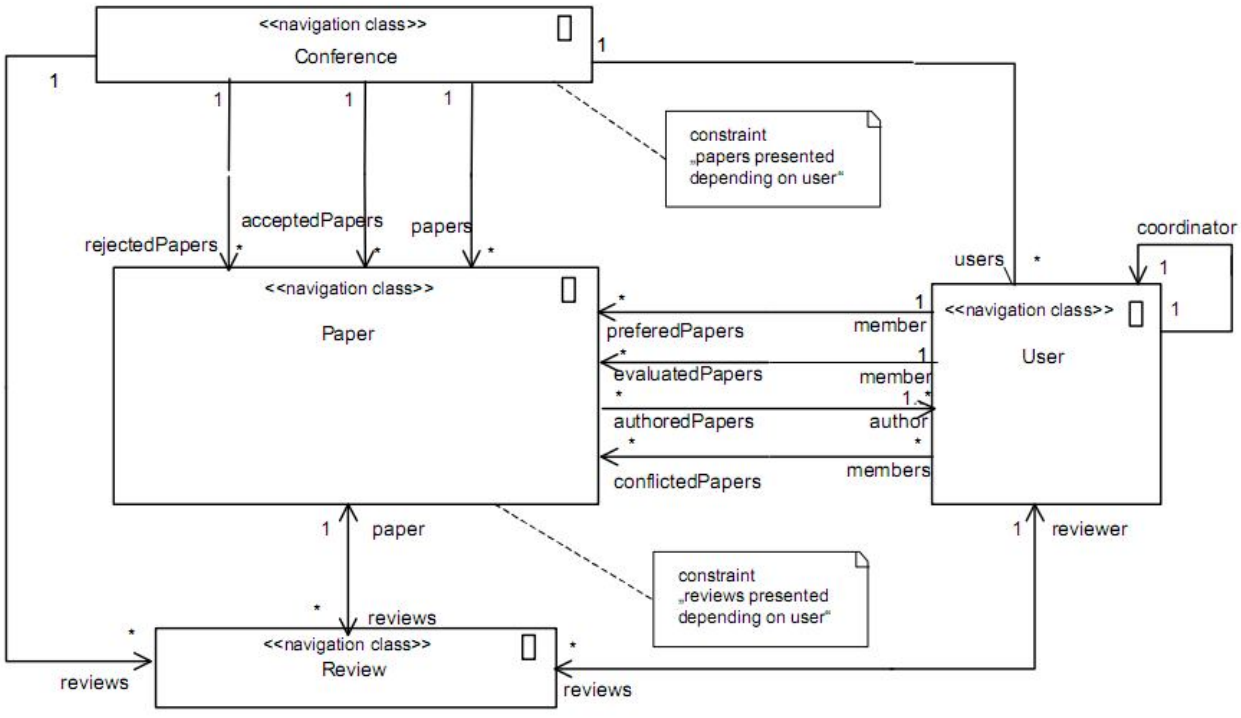

Figura 35. Modelo de espaço de navegação de um sistema de revisão de uma conferência.

Fonte: Koch, N.; Kraus, A.; Hennicker, R. (2001)

De acordo com Abreu, Lara e Ragazzo (2007), deve-se proceder da seguinte maneira na construção de um modelo espacial de navegação:

> a) incluir classes do modelo conceitual que são relevantes para a navegação (Navigation classes) no modelo espacial. Manter informações de classes omitidas (Classes conceituais, mas que não entraram no modelo de navegação espacial por não serem relevantes) por meio de atributos pertencentes a outras classes no modelo espacial de navegação. b) associações do modelo conceitual devem ser mantidas no modelo de navegação e outras associações podem ser adicionadas para melhorar a navegabilidade dentro da aplicação. c) Adicionar associações com base nas descrições de requisitos e situações descritas pelo modelo de casos de uso.

As limitações que definem restrições na navegação espacial devem ser definidas ao concluir os passos descritos.

b) Navegação estrutural

A diferença do modelo de navegação espacial para o modelo de navegação estrutural é que esta especifica como a navegação é sustentada por elementos de acesso nominados primitivas de acesso que são: índices, *guide tours*, *queries* e menus. Este modelo é construído empregando-se o diagrama de classes UML que utiliza como base o modelo espacial descrito anteriormente.

As Primitivas de acesso são nós de navegação incorporados ao modelo de navegação, com a finalidade de identificar o tipo de acesso feito aos objetos de navegação. Na figura 36 tem-se a representação gráfica dessas primitivas.

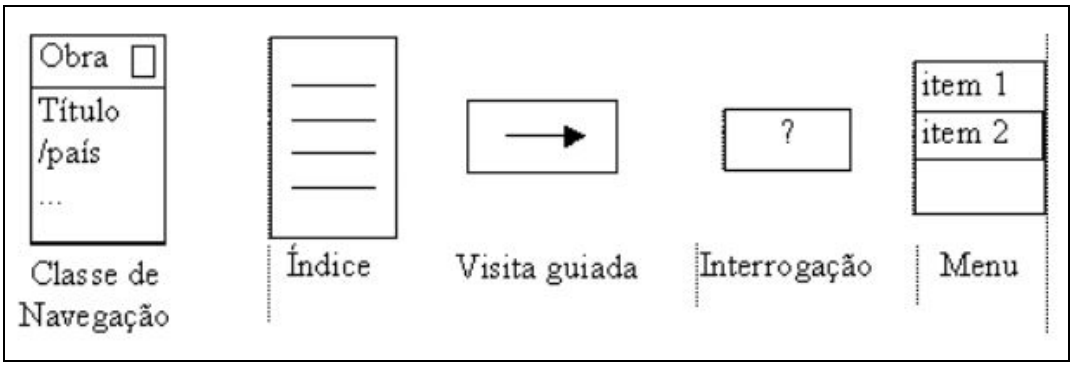

Figura 36. Representações visuais das primitivas de acesso

Fonte: LOPES, R. et al (2003)

Lopes et al (2003) descreve cada uma delas da seguinte maneira:

- índices: representam listas de ligações, ou âncoras, para instâncias da classe a qual estão ligadas à saída, ou seja, é uma associação dirigida do índice para a classe de destino. estas possuem base em parâmetros estabelecidos na classe de entrada, em outras palavras é uma associação dirigida da classe de origem para o índice;
- visitas guiadas: a informação oferecida pelas visitas guiadas é semelhante à anterior, porém, com o detalhe de ser uma navegação que se apresenta passo a passo, podendo retroceder ou avançar sem precisar saber qual ligação foi acessada;
- interrogações: são basicamente formas flexíveis de pesquisa, pois diferentemente dos menus, estes não são enumerados, finitos e pré determinados de ligações;

- menus: formados por uma lista finita e pré-determinada de ligações.

Abreu, Lara e Ragazzo (2007) descrevem os passos para a elaboração do modelo de navegação estrutural executando os seguintes passos:

> Primeiramente, substituir todas as associações bidirecionais que tenham multiplicidade maior que um nas duas extremidades da associação por duas associações unidirecionais correspondentes. Depois, substituir as que possuem multiplicidade maior que um em uma das extremidades por uma associação unidirecional com uma associação direcionada na extremidade da multiplicidade maior que um [...]. Posteriormente, considerar somente as associações do modelo espacial de navegação que possuem multiplicidade maior que um na extremidade direcionada da associação. Para cada associação desse tipo, escolher um ou mais elementos de acesso para realizar a navegação. Por fim, aumentar o modelo espacial de navegação correspondente.

Após a finalização dessas alterações, concluí-se o modelo de navegação estrutural movendo os nomes completos do modelo espacial para os elementos de acesso. O resultado dessa etapa é um diagrama de classes UML como a figura 37:

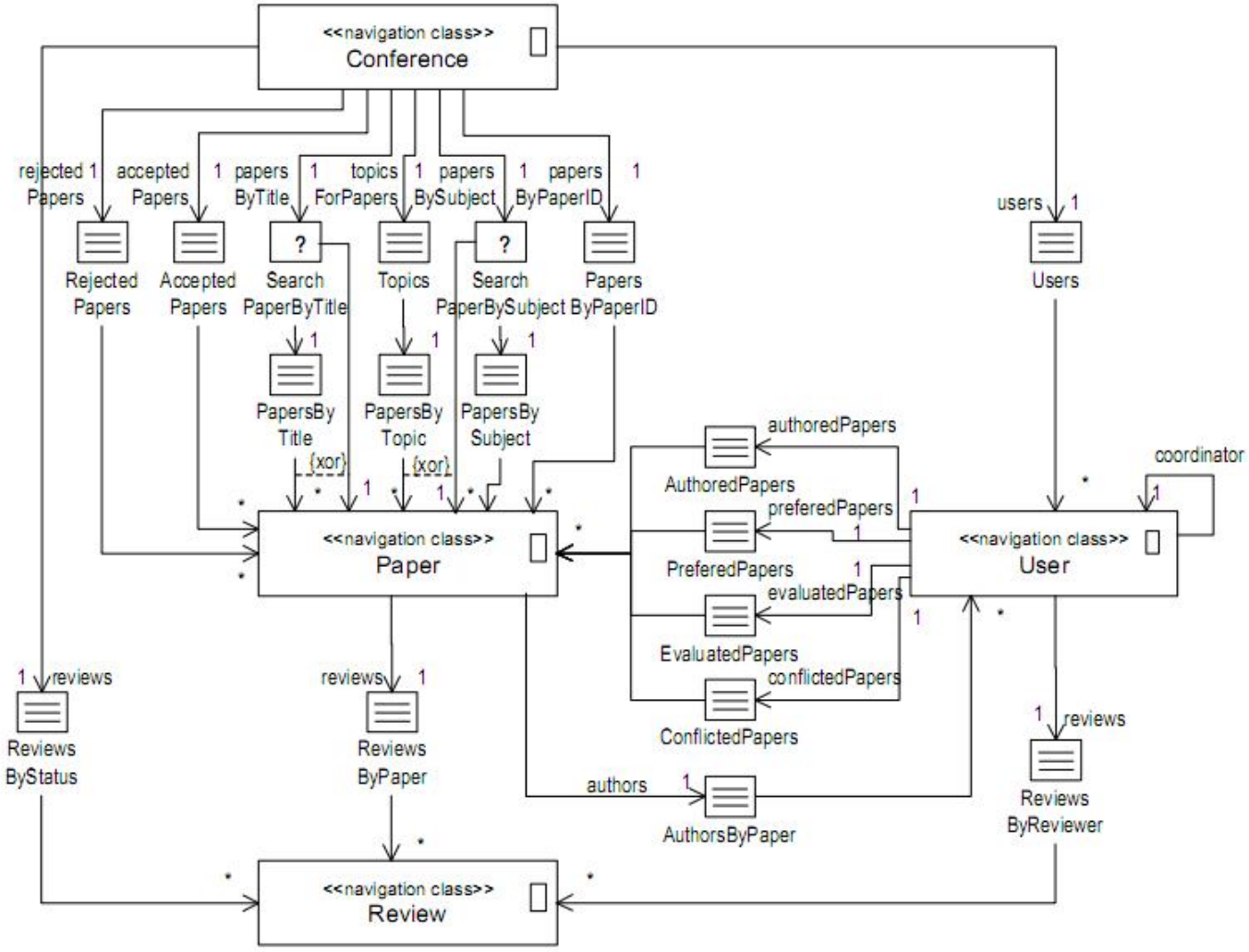

Figura 37. Modelo de navegação estrutural do sistemas de revisão de uma conferência.

Fonte: Koch, N.; Kraus, A.; Hennicker, R. (2001)

4.2.2.4 Etapa 4: APRESENTAÇÃO

O modelo de apresentação é criado a partir do modelo de estrutura de navegação e também de informações adquiridas no processo de análise de requisitos, segundo Abreu, Lara e Ragazzo (2007). Seu objetivo é definir um conjunto de *views* que mostram o conteúdo estruturado em nós simples além de mostrar as possibilidades que o usuário possui para interagir com a aplicação.

Hennicker (2001) expressa algo semelhante, porém, afirmou que o modelo de navegação é a representação esquemática da interface por meio de um conjunto de estereótipos UML. Essas representações são formadas pelos modelos de apresentação e estereótipos, podendo ser classificados em: enquadramento (*frame*), coleção, formulário, texto, imagem, entre outros.

De acordo com os autores, o modelo define onde as visões de interface são apresentadas ao usuário por meio de janelas e *frames*. Estes por sua vez indicam onde os conteúdos serão substituídos à medida que o usuário interage com essa interface, por exemplo, ao clicar em link.

Antes de estabelecer o modelo de apresentação o projetista deve definir a técnica empregada nessa aplicação, ou seja, quantas janelas ou frames irão compô-la. No caso de se empregar apenas uma janela, cada interação do usuário com a aplicação resultará na completa substituição do conteúdo da janela por um novo. A representação do modelo de apresentação é feita empregando-se o modelo de interação da UML, conhecido também por diagramas de sequência.

Os elementos que usados na modelagem dessa etapa são basicamente:

a) Janela: área que delimita a distribuição de objetos de apresentação. É basicamente uma instância de uma classe <<window>>.

b) *Frameset*: empregado quando se deseja estabelecer inúmeras áreas de visualização em uma mesma janela. Este elemento é uma instância de uma classe <<*frameset*>> e é sub-dividido em frames.

c) *Frame*: delimita uma das muitas áreas do *frameset* a que ele pertence, ou seja, estabelece o espaço para apresentar determinado conteúdo. Este elemento é representado pela instância de uma classe <<*frame*>>. Os *frames* podem conter também *framesets* aninhados.

A figura 38 ilustra os elementos descritos:

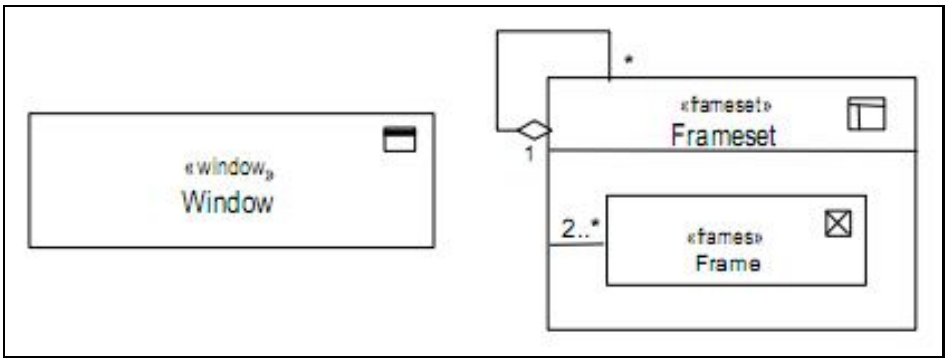

Figura 38. Elemento janela, frameset e frame.

Fonte: adaptado de Koch, N.; Kraus, A.; Hennicker, R. (2001)

O projetista deve definir, em um primeiro momento, algumas características para a elaboração do modelo de apresentação como a quantidade de janelas que será usada e o número de *frames* relacionados a cada *frameset*. Para facilitar esse processo Abreu, Lara e Ragazzo (2007) descrevem os seguintes passos a ser seguidos para a elaboração do modelo de apresentação:

> a) escolher entre a técnica de uma ou múltiplas janelas. Em caso de múltiplas janelas, planejar quantas janelas serão utilizadas. Escolher o estilo de frame, com ou sem frames. No primeiro caso, especificar o número de frames existentes em cada frameset. b) representar a estrutura de apresentação com um diagrama de classes da UML(opcional). c) ajustar o cenário para o modelo de interação, (Por exemplo, definir qual caminho da navegação no diagrama de estrutura de navegação será modelado). Um caminho de navegação é sempre relacionado com um caso de uso. c) Representar o usuário, a janela e/ou objetos de frames na dimensão horizontal. d) Especificar uma mensagem de display para cada objeto de apresentação que deva ser apresentado para o usuário (em uma janela ou frame). O parâmetro da mensagem de display é o objeto de apresentação correspondente. e) Incluir uma mensagem select para cada ação do usuário que selecione uma âncora ou um botão. A âncora ou o

> botão são os parâmetros da mensagem. f) Especificar uma mensagem fill e submit para cada ação do usuário, que consiste em fornecer dados em um formulário de consulta. Este formulário é o parâmetro da mensagem. Inclui uma mensagem para cada abertura e fechamento de uma janela. g) usar "balking" para especificar o período de tempo que uma janela ou frame está ativo.

Usa-se o diagrama de sequência para representar, de uma forma simples, o fluxo de controle da apresentação da aplicação. As figuras 39 e 40 ilustram como deve ser um modelo de apresentação.

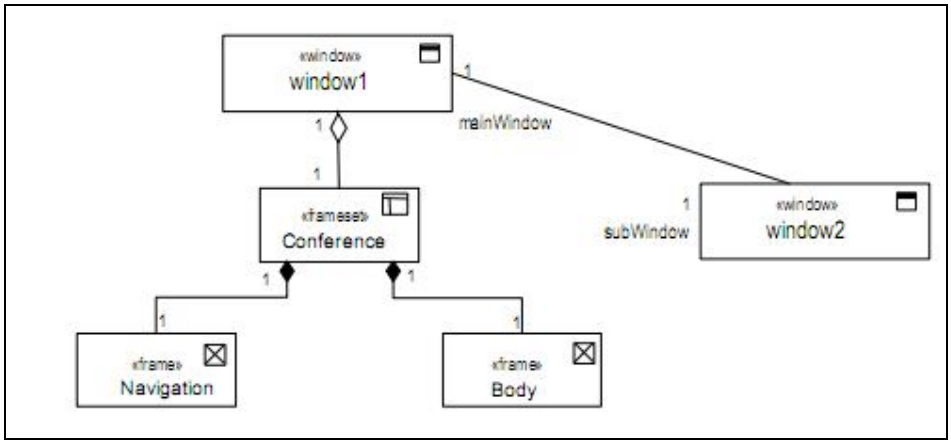

Figura 39. Localização dos elementos em uma interface do usuário

Fonte: Koch, N.; Kraus, A.; Hennicker, R. (2001)

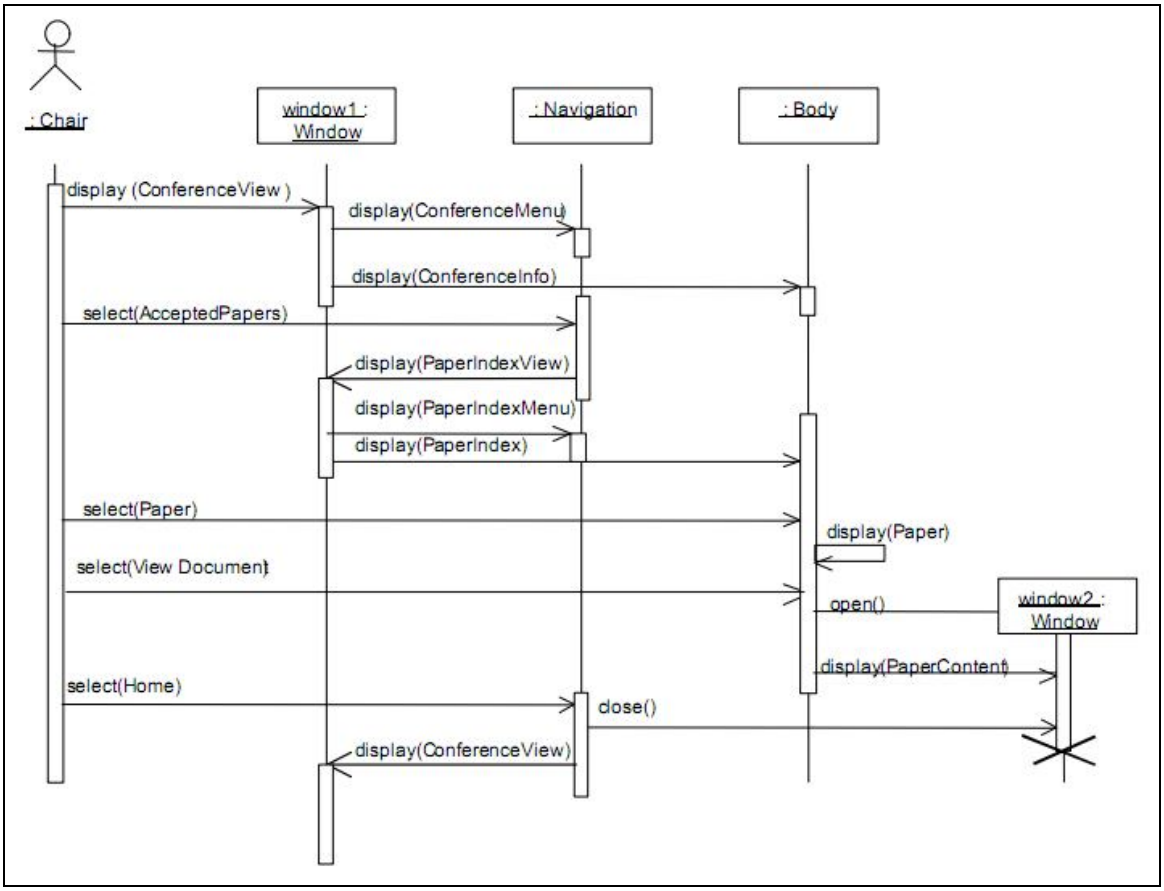

Figura 40. Visão de um fluxo de apresentação.

Fonte: Koch, N.; Kraus, A.; Hennicker, R. (2001)

A metodologia de modelagem UWE não disponibiliza os estereótipos de forma avulsa, dessa forma o modelo emprega mecanismos que visam interligar estes estereótipos com classes e estruturas de navegação definidas nos modelos anteriores.

4.3 FERRAMENTAS DE MODELAGEM DAS NOTAÇÕES

Como visto anteriormente, a modelagem de uma aplicação *Web* é composta por diversos modelos e para a elaboração desses se faz necessário o emprego de uma

ferramenta CASE para auxiliar e facilitar o processo de modelagem de cada uma das notações. Diante disso será feita uma breve descrição das ferramentas mais comuns para a UWE, WAE e WebML.

a) ArgoUWE: essa ferramenta CASE oferece diversos atributos para a modelagem de aplicações *Web* através da metodologia UWE . Ela é um pacote de expansão da ferramenta de modelagem ArgoUML por meio de bibliotecas adicionais e especificas para a notação UWE.

   A ferramenta ArgoUML pode ser encontrada no endereço http://argouml.tigris.org e é fornecida de forma gratuita. O pacote com as bibliotecas adicionais para a metodologia UWE é disponibilizado no endereço http://www.pst.informatik.uni-muenchen.de/projekte/argouwe/. Este endereço encontra-se dentro do site relacionado ao projeto UWE desenvolvido na Alemanha e oferece um tutorial para a instalação dessas bibliotecas no ArgoUML. Contudo, na versão disponibilizada atualmente não é suportado a modelagem do modelo de apresentação.

b) Estereótipos para o Rational Rose: O Rational Rose é uma ferramenta CASE que foi desenvolvida pela empresa IBM, sendo assim é um software proprietário que oferece subsídios para a modelagem de sistemas com a notação UML. Para habilita-lo para a notação WAE é requerido a instalação dos atributos específicos dessa notação. No endereço http://www.wae-uml.org encontra-se a arquivo de instalação dessa expansão para o Rational Rose.

c) Visual UML: Desenvolvida pela Visual Object Modelers, empresa membro da OMG (Object Management Group), o Visual UML é uma ferramenta

CASE proprietária. Ela oferece recursos para a modelagem de todos os diagramas da UML 1.3 e 1.4. Assim como as anteriores, a Visual UML possui extensões sendo eles: para modelagem de objeto de negócio, modelagem de aplicações *Web* (usando a notação WAE) e modelagem XML. Pode-se encontrar uma versão para testes no endereço http://www.visualuml.com que oferece 30 dias gratuitos.

d) WebRatio: é uma ferramenta CASE proprietária que foi desenvolvida pela empresa WebModels S.r.I. e que permite a modelagem e a geração de aplicações *Web*, baseados na notação WebML. A equipe de desenvolvimento dessa ferramenta contou com a ajuda dos criadores da WebML. Encontra-se uma versão de teste no endereço: http:///www.webratio.com.

5. TRABALHOS CORRELATOS

Na elaboração deste trabalho consultamos diversos outros trabalhos e pesquisas desenvolvidas a respeito dos assuntos de interesse para sua construção. A fim de expor trabalhos relacionados aos temas abordados, foram escolhidos os dois mais interessantes entre todos que foram consultados.

5.1 EXEMPLOS DE *FRAMEWORKS*

Esta sessão citará dois trabalhos desenvolvidos e baseados em *framework*. O primeiro que foca o desenvolvimento de avaliações empregando agentes inteligentes, que se comunicam com professores e estudantes e o segundo que aborda a construção cooperativa de ambientes virtuais.

5.1.1 Estratégias para o Desenvolvimento de um *Framework* de Avaliação da Aprendizagem a Distância

Prata (2003) propôs um *framework* que faz uso de agentes inteligentes para a elaboração de avaliações à distância.

Esse *framework* é dividido em dois principais ambientes: ambiente de aprendizagem do estudante e ambiente de aprendizagem do professor. Estes por fim são compostos por diversos módulos que auxiliam na elaboração (por parte do professor) e na resolução (por parte do estudante) conforme a figura 41:

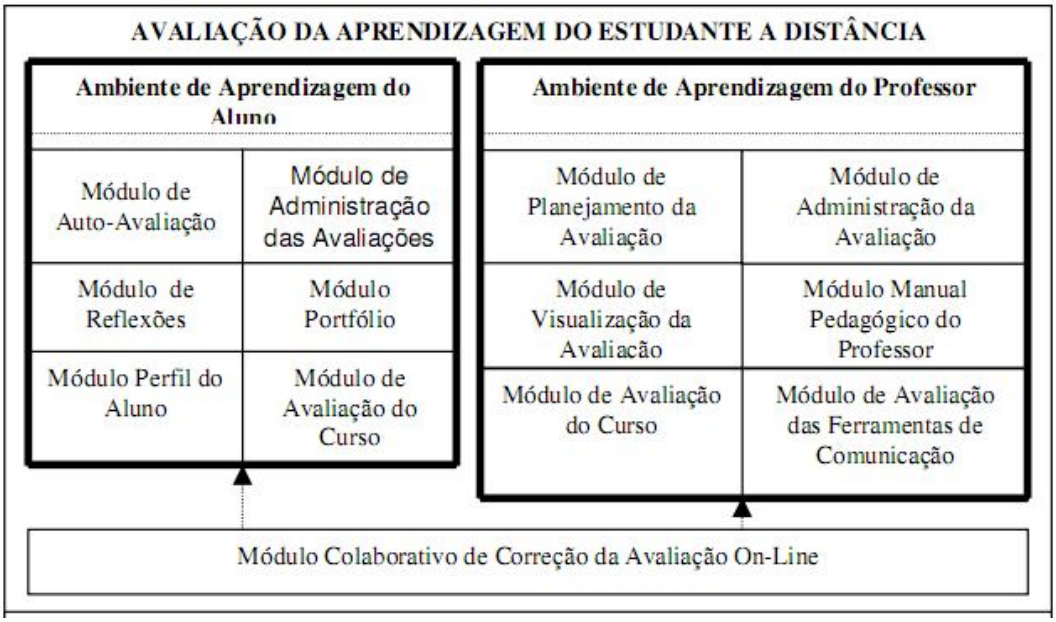

Figura 41. Módulos do ambiente de aprendizagem.

Fonte: Prata D. N. (2003)

O *framework* em questão faz uso de uma abordagem estratégica que se dá por meio de ontologias onde se estabelece, que para um estudante conhecer determinado assunto ele deve ter um conhecimento prévio do que já foi ensinado. A figura 42 ilustra essa abordagem estratégica:

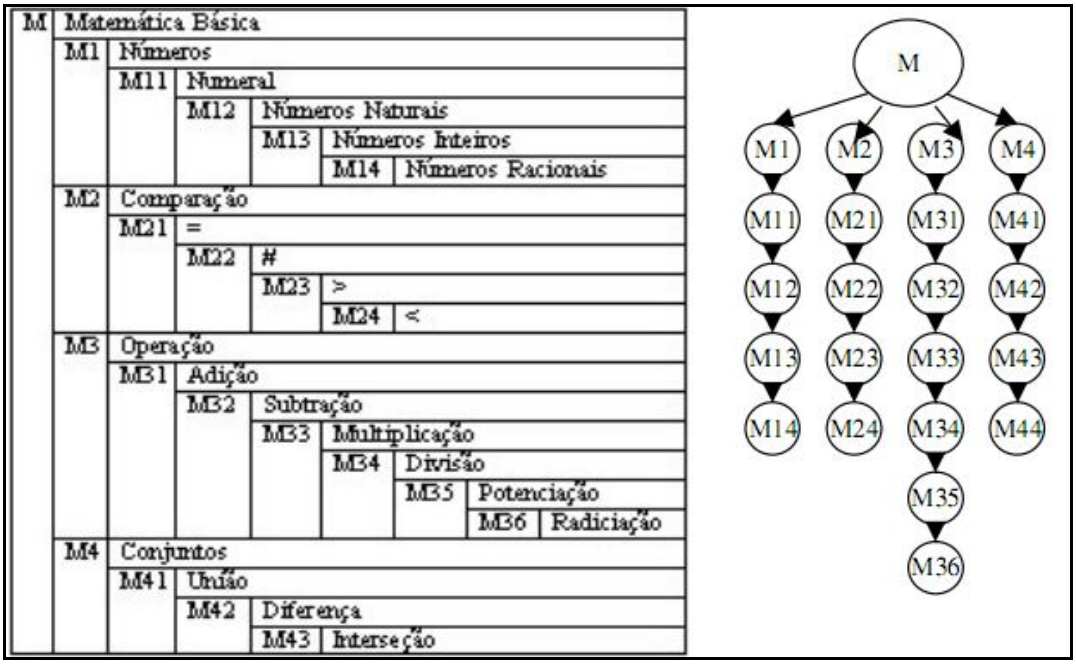

Figura 42. Ontologia de avaliação

Fonte: Prata D. N. (2003)

As ontologias citadas acima juntamente com novas sentenças que podem ser criadas formam a base de conhecimento para o agente inteligente (A1) que interage com o professor.

No entanto o agente inteligente (A2) que interage com o estudante trabalha percebendo o ambiente e estabelecendo conclusões a cerca da sua comunicação com A1, e da observação dos resultados obtidos nas avaliações pelo estudante. A2 atua como maximizador do processo de ensino, pois sugere ao estudante materiais de estudo, participação em grupos de discussão ou contato com o professor.

A Figura 43 ilustra a comunicação dos agentes com os usuários.

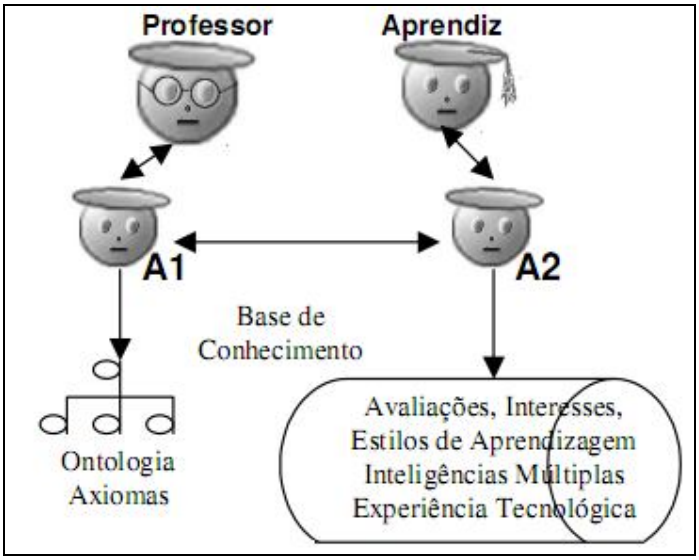

Figura 43. Agentes

Fonte: Prata D. N. (2003)

Prata (2003) justifica o emprego de agentes afirmando que "o uso de agentes e ontologia proporciona ao ambiente o diagnóstico e acompanhamento do aprendizado do estudante."

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

O trabalho desenvolvido por Pessoa e Menezes é uma proposta de modelo de desenvolvimento cooperativo dividido em duas camadas onde: a primeira concentra os esforços dos desenvolvedores e a segunda é desenvolvida a atividade de autoria na construção de ambientes virtuais que atende o modelo de aprendizagem, ou seja, a aplicação é moldada pelo usuário de acordo com sua necessidade e não pelo desenvolvedor. O *framework* proposto nesse trabalho é baseado em padrões da *Web* com enfoque especial no reuso de aplicações executáveis, e não de código fonte.

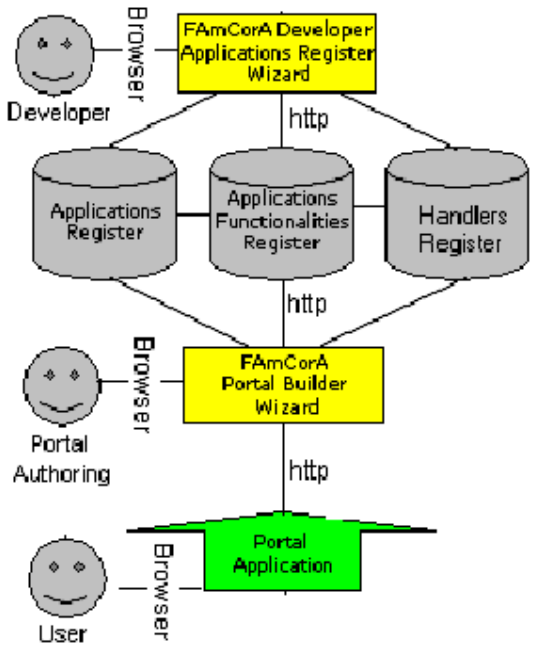

Figura 44. Arquitetura da Fábrica

Fonte: Pessoa; Menezes (2003)

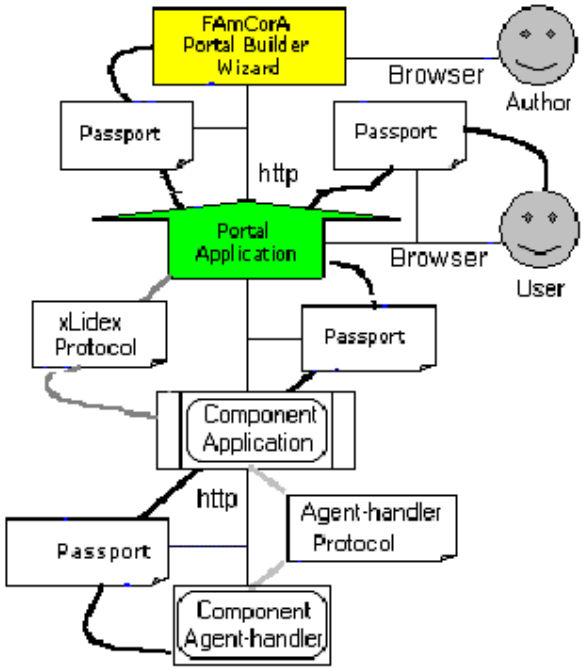

Figura 45. Arquitetura de Comunicação

Fonte: Pessoa; Menezes (2003)

A figura 45 ilustra a arquitetura de comunicação entre os módulos onde o módulo Portal *Builder Wizard* é o responsável pelo catálogo de portais e por outro lado o módulo Portal *Application* é o responsável pelo catálogo de grupos, usuários e papéis.

Esse *framework* também oferece um sistema de *log* a fim de gravar um histórico de interação dos usuários. O *log* das atividades é mediado pelo protocolo xLidex (*Extensible Learning Interactions Data Exchange*).

O Protocolo *Passport*, encontrado na figura 45, é o principal mediador da arquitetura desse *framework* quando se diz respeito a modelagem não funcional, ou seja, organização, coordenação, autenticação (portal, grupo, usuário), configuração e composição (reuso).

O *passport* do portal possui a sua configuração básica. No entanto, as configurações de cada usuário são mantidas pelo *passport* do indivíduo, dessa forma é criado um *passport* novo ao fazer "*login*".

Os componentes (*famcoralets*) são desenvolvidos por colaboradores especialistas em programação e armazenados em um repositório do *framework* constituindo assim os módulos básicos. O componente ilustrado na figura 46 possui permissão para publicar métodos, propriedades, eventos e receptáculos.

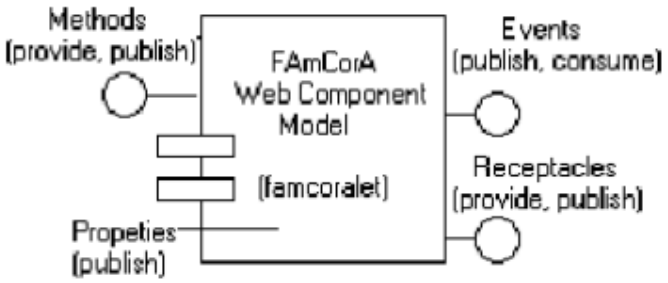

Figura 46. Modelo de Componente
Fonte: Pessoa; Menezes (ANO)

Em suma, um componente participa do processo de registro ao *framework* e se comunica com outras aplicações de acordo com os protocolos estabelecidos pelo mesmo.

Outro fator importante que merece destaque é o fato de todo usuário inscrito em um grupo possuir um papel, ou nível de acesso. De acordo com o papel do usuário, determinadas ferramentas ficam ou não disponíveis para o mesmo, além disso, cada ferramenta pode ser configurada.

O autor (quem cria as aplicações com os componentes disponíveis) possui permissão para criar novos grupos e papéis assim que surgir a necessidade, configurando e reconfigurando as ferramentas. O mais importante é que essa configuração não acarreta em impacto sobre o ambiente devido ao fato de que o trabalho de autoria é basicamente uma configuração de (re)uso.

5.2 EXEMPLOS DE PROJETOS MODELADOS COM UWE

As aplicações que serão descritas a seguir tiveram como fator decisivo na escolha da metodologia UWE o fato de serem aplicações para um ambiente *Web*. A primeira aplicação visa disponibilizar todos os dados que atualmente encontra-se em um documento físico, que é a carteira de trabalho, de forma *on-line*. Para isso o histórico profissional seria cadastrado em páginas dinâmicas onde as empresas teriam um *login*, que daria permissão de acesso a esses dados, tanto para cadastro manutenção, quanto para consulta de possíveis candidatos.

O segundo,diz respeito a modelagem de um museu interativo onde o emprego da UWE se justificou por ser a metodologia que melhor expressa os detalhes de uma aplicação *Web*. O trabalho desenvolvido por Lopes et al (2005) chegou a ser implementado, porém foi descrito apenas a parte que se refere a modelagem.

5.2.1 Modelagem do Software Carteira de Trabalho *On-line* Usando UWE:

Rocha e Costa (2004) modelaram o Software Carteira de Trabalho *On-line* valendo-se do recurso UWE. A proposta da modelagem visa auxiliar os empregadores na coleta de informações mais detalhadas acerca dos candidatos à vaga de emprego.

A motivação para a criação desta proposta esta nos problemas encontrados na carteira de trabalho atual, como por exemplo: espaço insuficiente para escrever devido à portabilidade do documento, perigo de perda ou extravio, onde ao se criar uma segunda via as informações antigas não podem ser recuperadas, entre outras questões.

Na proposta apresentada, a utilização da carteira de trabalho pode ser realizada por meio da Internet ou, particularmente, por uma empresa através de sua rede local.

O principal objetivo dessa proposta é viabilizar a visualização das informações trabalhistas de determinada pessoa, por meio de uma consulta direta na *Web* e a atividade de cadastro realizada no documento, seria migrada para a *Web*, onde usuários registrados teriam acesso à aplicação por páginas dinâmicas.

Essa proposta oferece agilidade e facilita as consultas e cadastros das informações trabalhistas, uma vez que na *Web* o limite para descrição de determinado conteúdo é maior que no documento em papel.

5.2.3 Modelagem do Produto de Software

A modelagem como perfil para a UWE, teve como elemento motivador, o fato de que a mesma oferece diversos recursos para a modelagem de aplicações *Web*, além de técnicas de modelagem visual e de desenvolvimento em processo iterativo.

Na prática, muitos modelos derivam desta proposta, como as que são listadas abaixo.

a) Espaço do Modelo de Navegação: modelo que abrange as classes do produto de software e os objetos que podem ser visitados durante a navegação. Como elementos principais, constam: tela principal, chamada de Classe de Navegação, e a direção da navegabilidade. As páginas adicionais serão descritas em outro modelo.

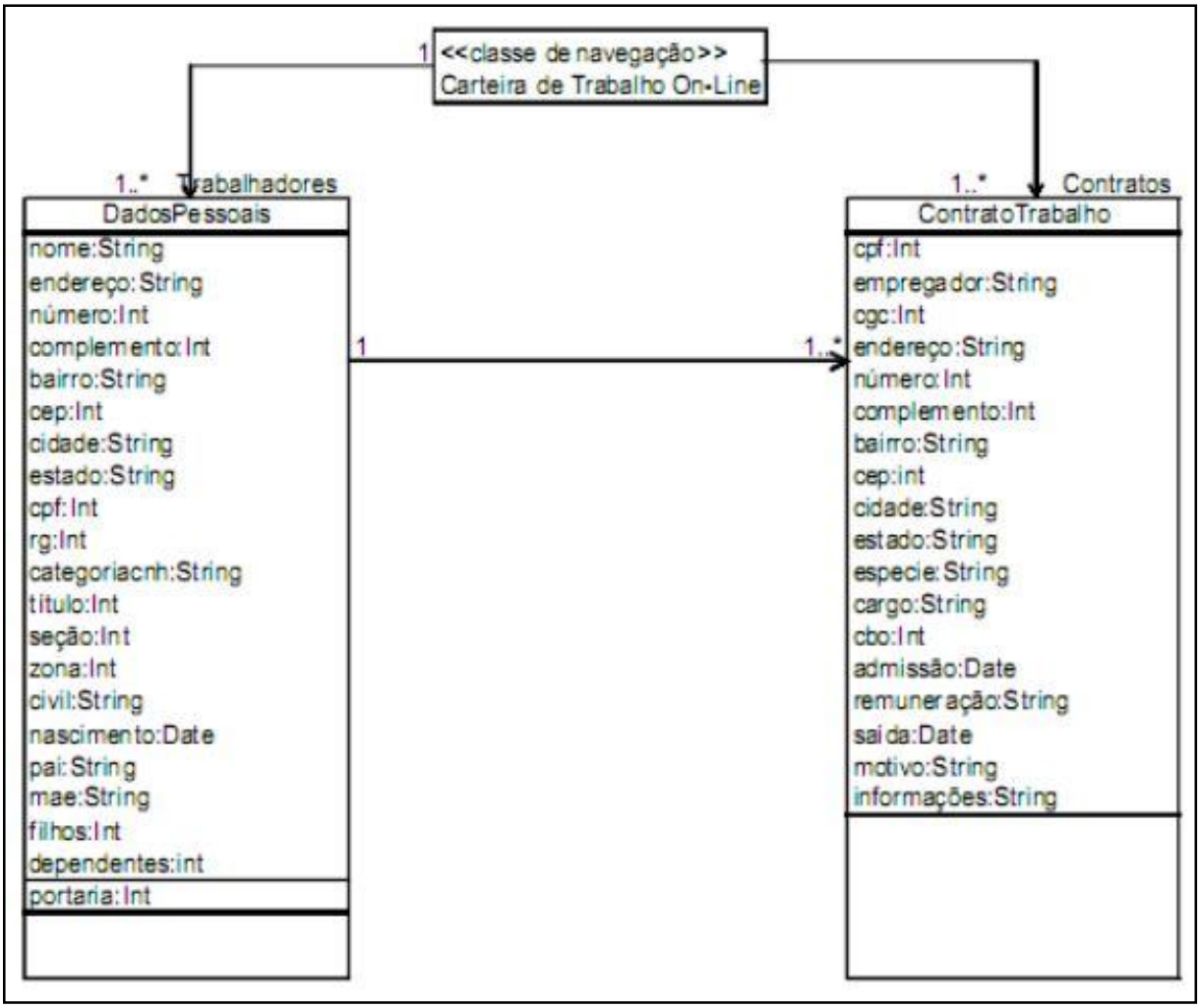

Figura 47. Espaço do Modelo de Navegação

Fonte: ROCHA, Paulo R.S.; COSTA, Heitor A.X (2004)

b) Estrutura do Modelo de Navegação: Estrutura que utiliza o Espaço do Modelo de Navegação como base. Neste modelo, as telas secundárias da aplicação, os acessos primitivos como: menus, índices, consultas são detalhados, sendo estes elementos os responsáveis por definir o tipo de acesso e as funções das telas.

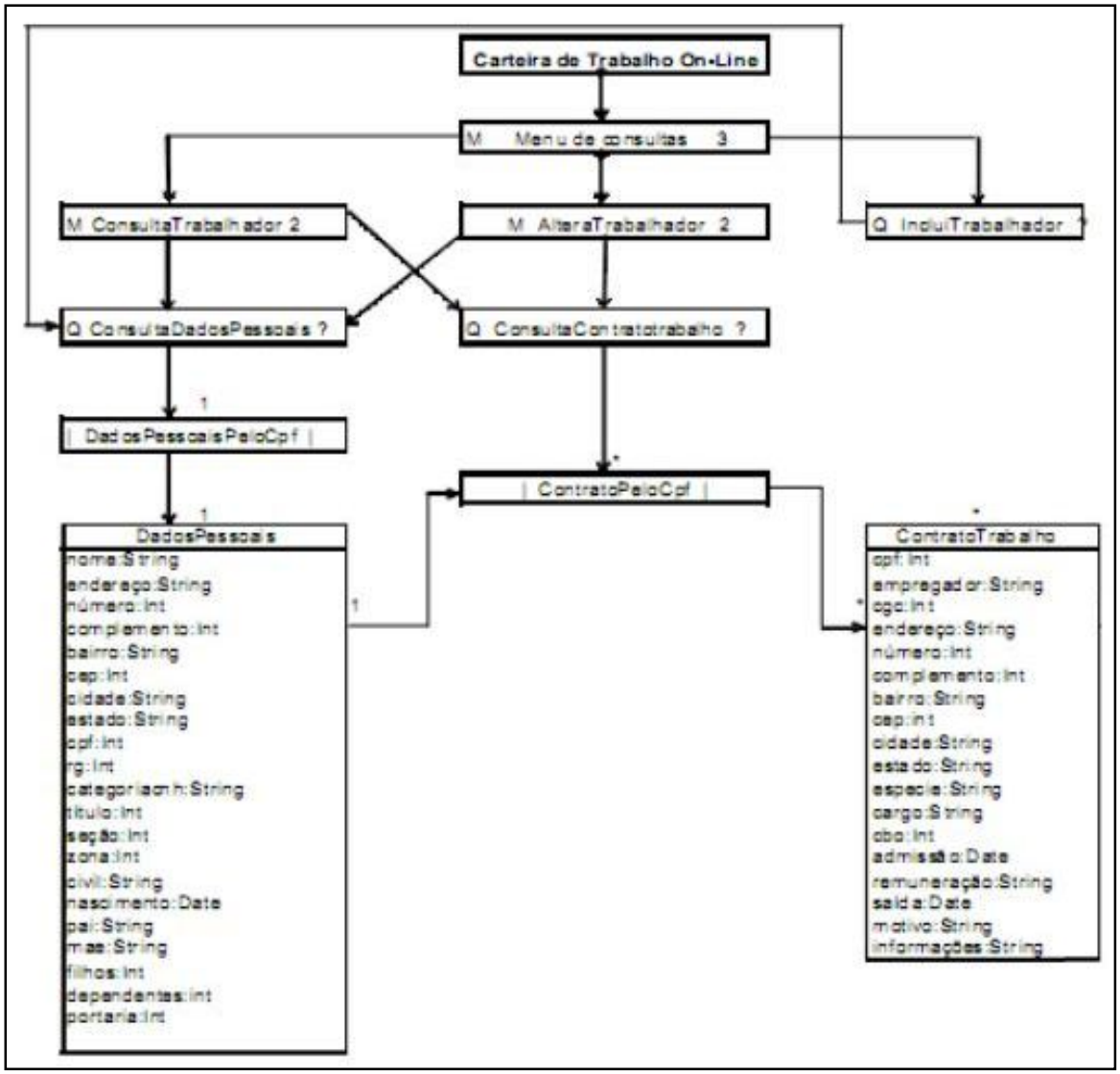

Figura 48. Estrutura do Modelo de Navegação com Acessos Primitivos

Fonte: ROCHA, Paulo R.S.; COSTA, Heitor A.X (2004)

c) Modelo de Apresentação: Na proposta de Rocha e Costa (2004), optou-se por criar uma Classe de Aplicação para cada elemento de navegação que é descrito no Modelo Navegacional. Este por sua vez, conecta-se a classes adjacentes por composição.

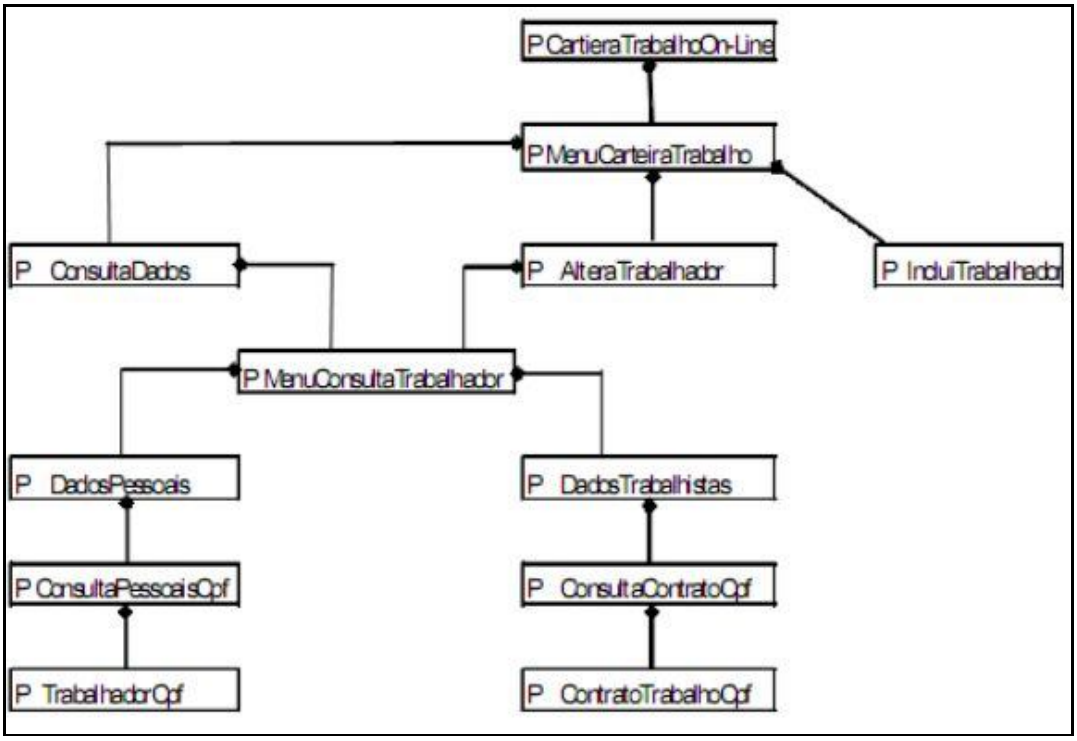

Figura 49. Diagrama de Apresentação

Fonte: ROCHA, Paulo R.S.; COSTA, Heitor A.X (2004)

Rocha e Costa (2004) concluem afirmando que:

> Com a modelagem proposta pela UWE, foi possível descrever o desenvolvimento da aplicação como um processo iterativo e incremental, que sofre alterações durante o processo de construção, gerenciando os requisitos de acordo com o seu progresso.

Este trabalho confirma que a utilização da UWE na modelagem de aplicações *Web* realmente é eficaz, pois possue ferramentas para descrever as particularidades encontradas nessas aplicações.

5.2 ARQUITECTURAS XML PARA CONCRETIZAÇÃO DE MODELOS HIPERMÉDIA:

No trabalho proposto por LOPES et al (2005) nominado de “Arquitecturas XML para Concretização de Modelos Hipermédia”, os autores relatam um caso de

estudo onde se empregou a UWE na modelagem de um sistema hipermídia com o tema “Museu de Arte Interactivo”. O estudo passou pela modelagem e implementação do sistema proposto, porém será exposta apenas a modelagem.

A Figura 50 mostra o modelo conceitual da aplicação proposta. Esta é formada pelos conceitos de classes e atributos, associação e generalização da UML.

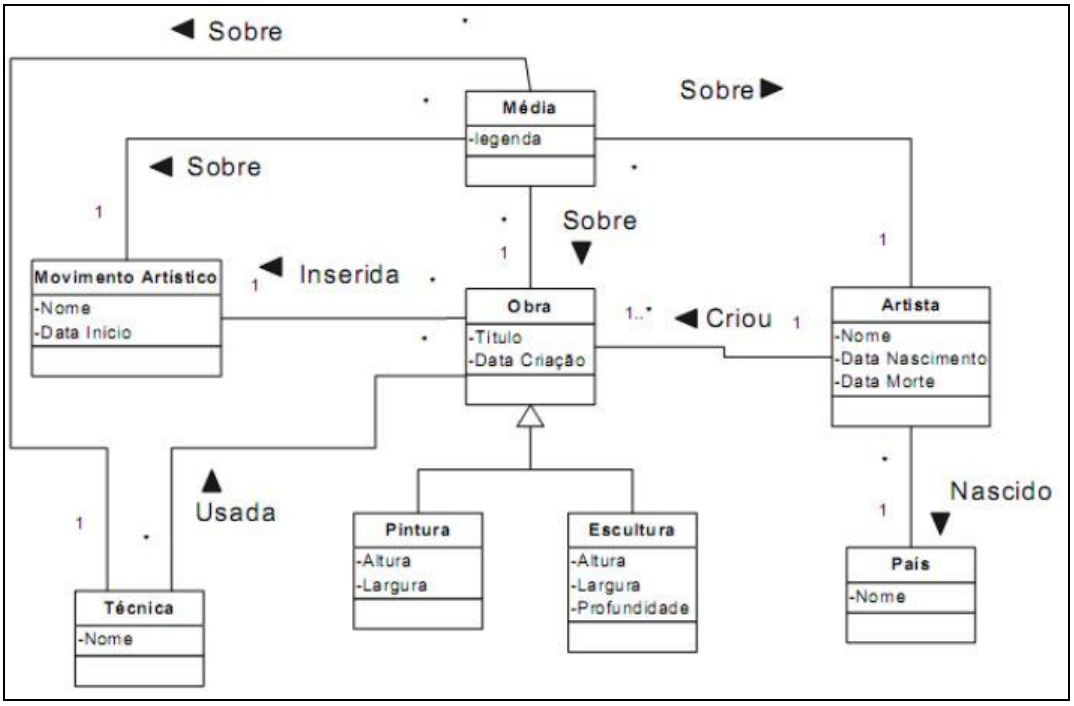

Figura 50. Fragmento do Modelo Conceitual

Fonte: LOPES et al. (2005)

Após a definição do modelo conceitual inicia-se a construção do modelo do espaço de navegação. Neste, emprega-se a UWE ao introduzir as estruturas de navegação como: menus, índices e visitas guiadas resultando em algo similar como na Figura 51.

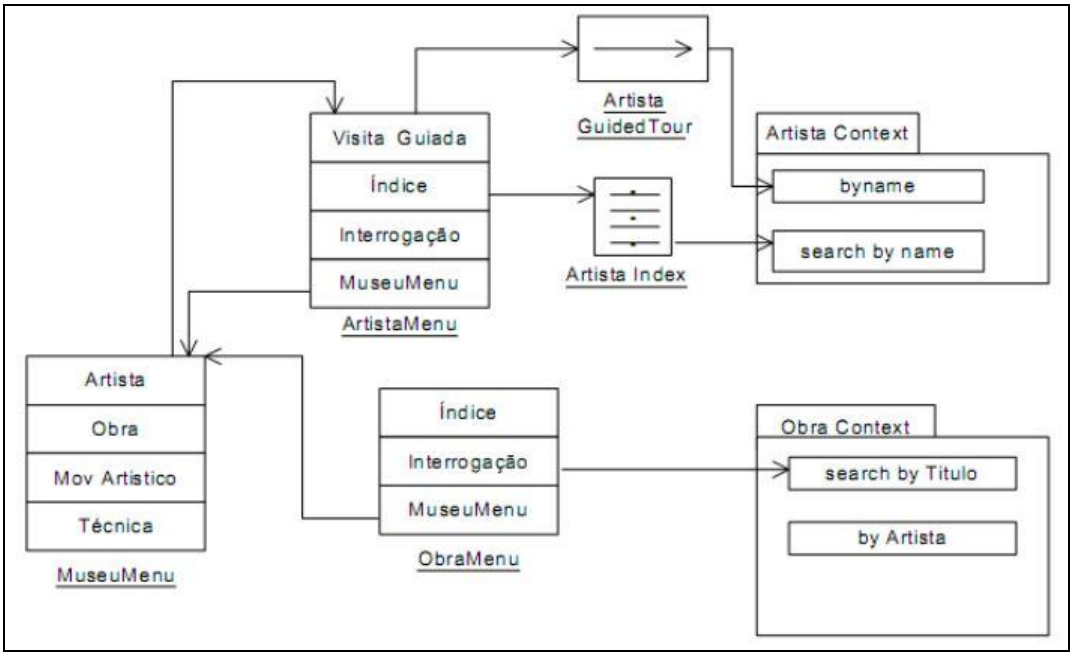

Figura 51. Fragmento dos modelos de Estrutura de Navegação

Fonte: LOPES et al. (2005)

Na finalização do trabalho apresentado, foi possível identificar os padrões e procedimentos necessários para a construção de uma plataforma XML que incorpora os conceitos propostos nas metodologias de sistemas hipermídia, adotando principalmente mecanismos de validação e verificação, a fim de garantir a coerência no desenvolvimento destes sistemas. Como resultado obteve-se, além do sistema, questionamentos acerca dos três níveis de suporte requeridos: meta-modelação, modelação e instanciação, além da sua concretização numa plataforma XML.

Apesar do foco deste trabalho residir no emprego de XML, a concretização tornou-se viável a partir da modelagem do sistema *Web*. Para tal, usou-se a UWE que melhor se adaptou às características desse tipo de sistema, e que permite um detalhamento maior na modelagem de sistemas para *Web*, garantindo assim a qualidade do projeto.

6. *FRAMEWORK* T2A - TESTES DE AVALIAÇÃO DE APRENDIZAGEM

A escolha de modelar um *framework* e não uma aplicação para a geração de testes de avaliação de aprendizagem baseou-se na flexibilidade encontrada no primeiro. O *framework* oferece diversos recursos e opções mais elaboradas, que podem ser configurados de acordo com a necessidade do professor facilitando assim, sua adaptação a uma nova ferramenta.

Para que a modelagem fosse bem especificada e detalhada empregou-se a UWE, pois, dentre as pesquisadas, essa possui mais artefatos especializados para aplicações *Web*. Além disso, a metodologia proposta por Carneiro (2003), que foca o desenvolvimento de *frameworks*, foi utilizada para o desenvolvimento da modelagem devido a sua construção em vários ciclos, onde o resultado de cada um é o *framework* mais completo a cada iteração.

Como a proposta desse trabalho era a modelagem do *framework*, a metodologia proposta por Carneiro (2003) foi adaptada suprimindo alguns passos de desenvolvimento e teste da aplicação.

6.1 METODOLOGIA

A metodologia aplicada para o desenvolvimento do *framework* T2A é formada primeiramente pelo levantamento bibliográfico acerca de *Framework*, UWE visando compreender os artefatos, aplicação e modelagem de aplicação *Web*, aprimorar os conhecimentos sobre exercícios de verificação de aprendizagem (EVA) *on-line* conhecendo de forma detalhada os conceitos, tipos, instrumentos e estratégias de aplicação além dos tipos de questões, orientações, elementos de composição e interpretação dos resultados.

Após ter o conhecimento suficiente para tomar as decisões foram definidas a teoria pedagógica de embasamento, a metodologia de elaboração dos EVAs e a especificação dos mecanismos de *feedback* e mediação.

Com o conceito fundamentado, passou-se para a etapa de estruturação do *framework* detalhando as classes, componentes e métodos e definindo os pontos de flexibilidades (*hot spots*) que permitem a adaptação do *framework* para geração de aplicações.

E por fim, diagramar os artefatos UML estendidos para *Web* (UWE) que compõem a estrutura do *framework* T2A especificando sua estrutura, pontos e formas de adaptações, mecanismos de *feedback* e de mediação pedagógica.

6.2 MODELAGEM DO *FRAMEWORK* T2A

O *framework* foi desenvolvido empregando-se a metodologia proposta por Carneiro (2003) e UWE para realizar a modelagem do mesmo. Essa metodologia foi escolhida por descrever, de forma detalhada, o seu processo de desenvolvimento. Além disso, esta metodologia apresenta-se vantajosa, em relação às demais, visto que, torna a complexidade controlável à medida que limita a descrição de um domínio à sub-domínios, reduz o tempo requerido para seu desenvolvimento, a realimentação é gerada mais cedo no processo e a documentação de cada fase serve para o entendimento do *framework* e sua futura utilização no desenvolvimento de aplicações.

Nos sub-domínios que compõe a modelagem têm-se a etapa onde se faz a definição do domínio do problema, o levantamento dos requisitos funcionais e não funcionais, a definição dos *hot spots* e *frozen spots*, além da modelagem em si onde foram criados os diagramas de casos de uso, classes e conceitual. Durante esse processo as etapas se repetiram em diversas iterações onde o trabalho desenvolvido era revisado e

melhorado para dar prosseguimento aos passos seguintes visando à concretização do *framework*, quando este atingisse todos os seus objetivos.

6.3 APLICAÇÃO DA METODOLOGIA NO DESENVOLVIMENTO DO *FRAMEWORK*

O desenvolvimento deste trabalho ocorreu conforme as etapas contidas na metodologia citada anteriormente. A cada nova interação a modelagem do *framework* tornava-se mais detalhada e novos componentes era atribuídos.

O objetivo de modelar este *framework* era projetar um recurso que abrangesse o maior número possível de atividades do interesses dos professores na elaboração de testes de verificação de aprendizagem, e não modelar mais uma ferramenta como as diversas encontradas na *Web*.

A modelagem desenvolvida será apresentada na mesma estrutura contida na metodologia utilizada.

6.3.1 Passo 1 - Analisar Domínio

6.3.1.1 Objetivo:

a) Prover uma estrutura reutilizável para criação de aplicações para elaboração, mediação e verificação de exercícios de aprendizagem *on-line*.

6.3.1.2 Limites:

a) Não será considerada correção de exercícios dissertativos.

6.3.1.3 Benefícios:

a) Diversificação na construção de testes com mecanismos de mediação não fixando uma linha de montagem, ou seja, as intervenções de mediação poderão acontecer em qualquer momento conforme configurado.
b) Permite a extensão de classes para adicionar novos tipos de questões e mediações.
c) Oferece *feedback* ao estudante em tempo real ou após o fechamento de um módulo/avaliação, por exemplo.
d) Fornece flexibilidade na montagem do teste optando por informar a resposta certa logo após a submissão do resultado ou não.

6.3.1.3 Modelo estático preliminar:

O *Framework* proposto é baseado no MVC (Model View Controller) onde a lógica encontra-se separada da interface do usuário. A *View* expõe as informações para o usuário por meio da interface; o *Controller* gerencia a entrada de dados do usuário com as funcionalidades da aplicação. O *Model* é responsável pela lógica e dados da aplicação.

O diagrama de classes preliminar ou modelo estático é ilustrado na Figura 52.

6.3.1.4.1 Primeira Iteração

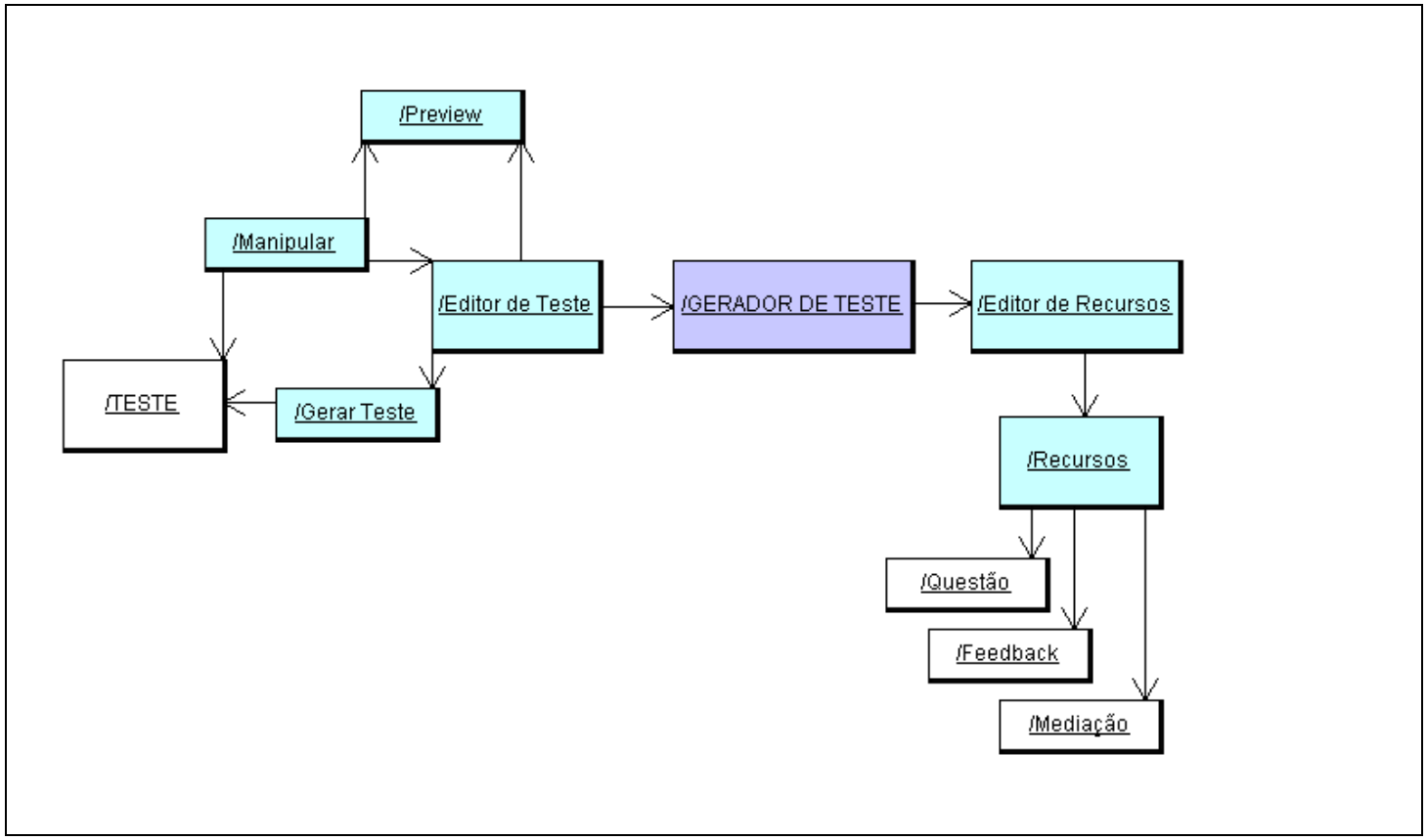

Figura 52. Modelo Estático 1ª Iteração

6.3.1.4.2 Segunda Iteração

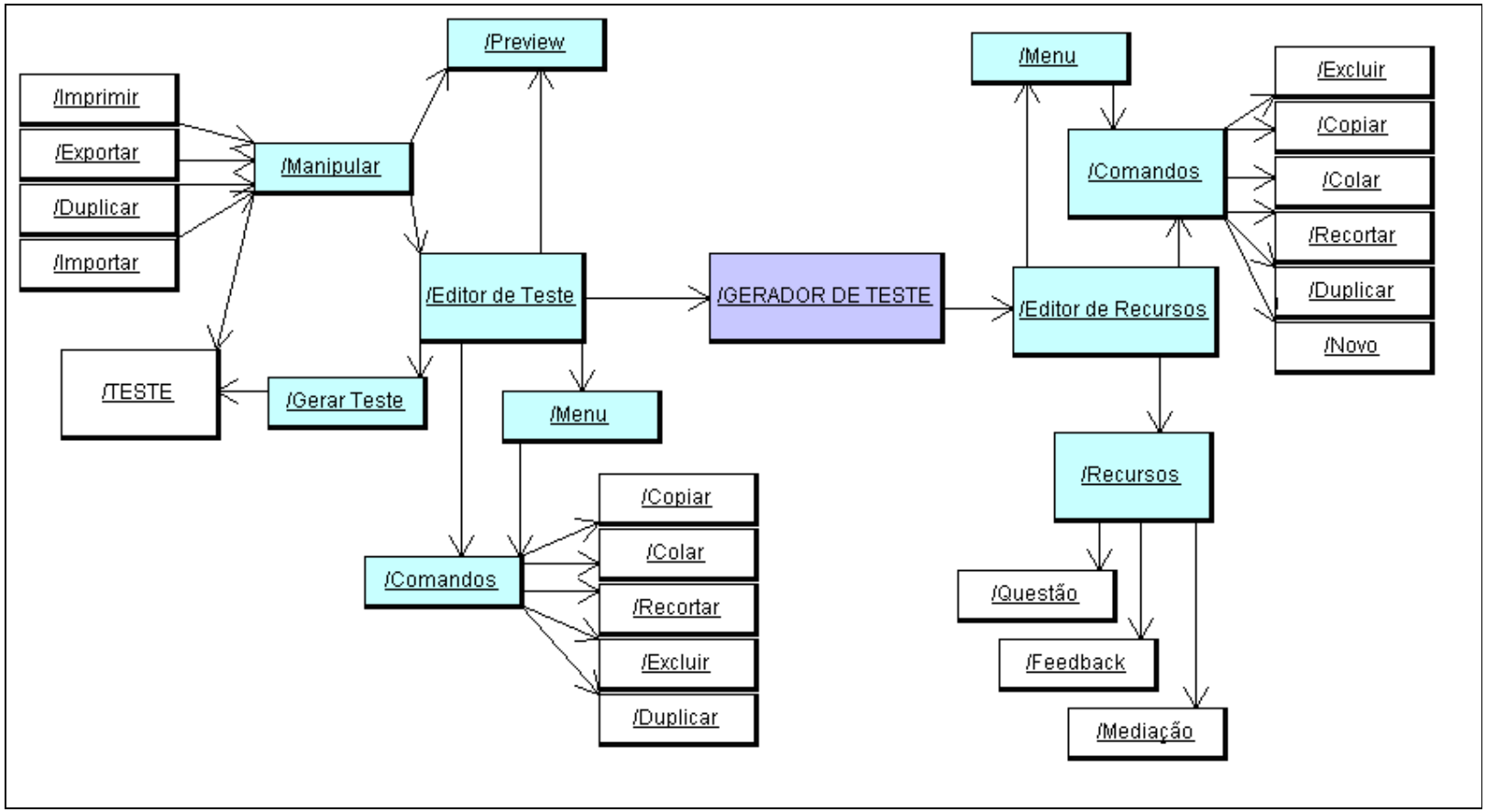

Figura 53. Modelo Estático 2a Iteração

6.3.1.4.3 Terceira Iteração

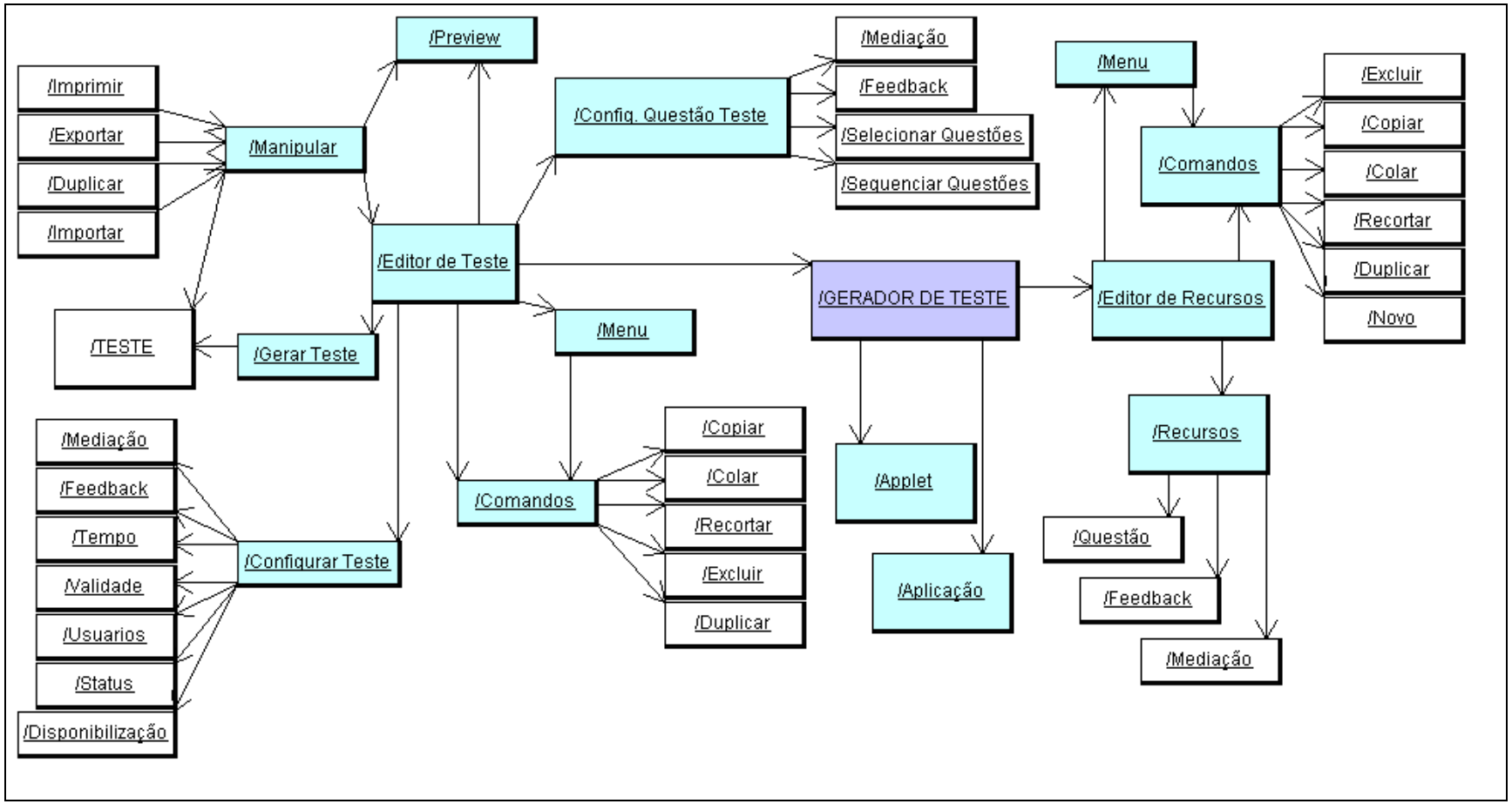

Figura 54. Modelo Estático 3a Iteração

6.3.2 Passo 2 - Construir

A construção do *framework* é sub-dividida em cinco (5) passos.

6.3.2.1 Passo 2.1 – Coletar Requisitos

O gerador de teste é um *framework* que oferece a possibilidade de criar questões para a montagem de uma avaliação, tais como: questões, mediações que podem ocorrer tanto antes (Ex: introdução à questão); durante (Ex: após determinado tempo liberar uma dica) ou ao final (Ex: informando a resposta correta).

Os recursos de teste são caracterizados pelo tipo e configurações flexíveis. Além disso, o professor pode usar as ferramentas disponíveis no próprio *framework* como o gerar teste, buscar testes arquivados, gravar os testes criados, remover ou editar.

6.3.2.1.1 Requisitos Funcionais:

a) restringir acesso por *login* e senha;
b) configurar questão;
c) buscar questão arquivada;
d) editar questão;
e) excluir questão
f) configurar mediação;
g) buscar mediação arquivada;
h) editar mediação;
i) excluir mediação
j) configurar *feedback*;
k) buscar *feedback* arquivado;

l) editar *feedback*;

m) excluir *feedback*;

n) gerar avaliação;

o) buscar avaliação arquivada;

p) editar avaliação;

q) excluir avaliação;

r) visualizar avaliação;

6.3.2.1.2 Requisitos não funcionais:

a) ser extensível possibilitando adicionar novos *feedback*;

b) ser extensível possibilitando adicionar novos tipos de *feedback*;

c) ser extensível possibilitando adicionar novas mediações;

d) ser extensível possibilitando adicionar novos tipos de mediações;

e) ser extensível possibilitando adicionar novas questões;

f) ser extensível possibilitando adicionar novos tipos de questões;

g) associar uma resposta à questão;

h) associar mediação à questão;

i) associar *feedback* à questão;

j) ter um *layout* simples e intuitivo;

k) possuir documentação;

l) possuir orientação (*help*).

6.3.2.1.3 Descrição dos Casos de Uso:

a) Caso de Uso Acesso ao gerador de teste:

Descrição: o acesso ao gerador de teste é feito por meio de um *login* e senha que é validado pelo módulo de gerenciamento de usuários.

b) Caso de Uso Gerar Teste:

Descrição: A geração do teste feita pelo agrupamento dos recursos previamente cadastrados.

c) Caso de Uso Buscar teste:

Descrição: A busca dos testes arquivados é feita por meio da ferramenta Busca disponibilizada no *framework*.

d) Caso de Uso Visualização de teste:

Descrição: A visualização ocorre por meio do ambiente *Preview* onde o professor pode interagir com a avaliação, mas os dados de uso do teste não serão salvos.

e) Caso de Uso Editar teste:

Descrição: Após a busca do teste que se deseja alterar o usuário acessa a opção de edição para alterar as configurações (questões associadas, dos mesmos).

f) Caso de Uso Exclusão de teste:

Descrição: Após a busca do teste que se deseja excluir o usuário acessa a opção de exclusão para remover o mesmo.

g) Caso de Uso Salvar o teste:

Descrição: Após a geração do teste, o mesmo é salvo quando o usuário acessa a opção Salvar;

h) Caso de Uso Criar Questão:

Descrição: O usuário escolhe um *template* de questão e preenche os dados necessários como a pergunta e resposta.

i) Caso de Uso Criar Tipo Questão:

Descrição: O usuário criar um *template* como modelo de questão e adiciona a base.

j) Caso de Uso Configurar Questão:

Descrição: O usuário associa uma mediação, antes, durante e/ou depois da questão. Além disso, associa um *feedback.*

k) Caso de Uso Criar Tipo Mediação:

Descrição: O usuário criar um *template* como modelo de mediação e adiciona a base.

l) Caso de Uso Configurar Mediação:

Descrição: O usuário configura o tipo de mediação a um objeto do mesmo.

m) Caso de Uso Criar Tipo *Feedback*:

Descrição: O usuário criar um *template* como modelo de *feedback* e adiciona a base.

n) Caso de Uso Configurar *Feedback*:

Descrição: O usuário configura o tipo de *feedback* a um objeto do mesmo.

6.3.2.1.4 Diagrama dos principais Casos de Uso:

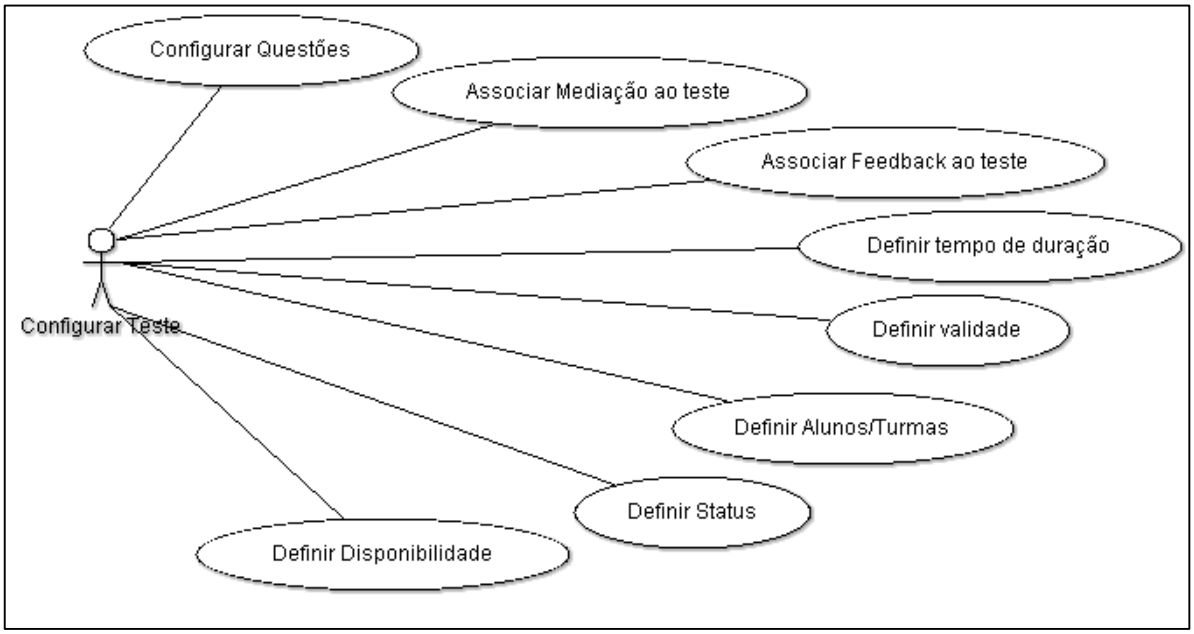

Figura 55. Caso de Uso Configurar Teste

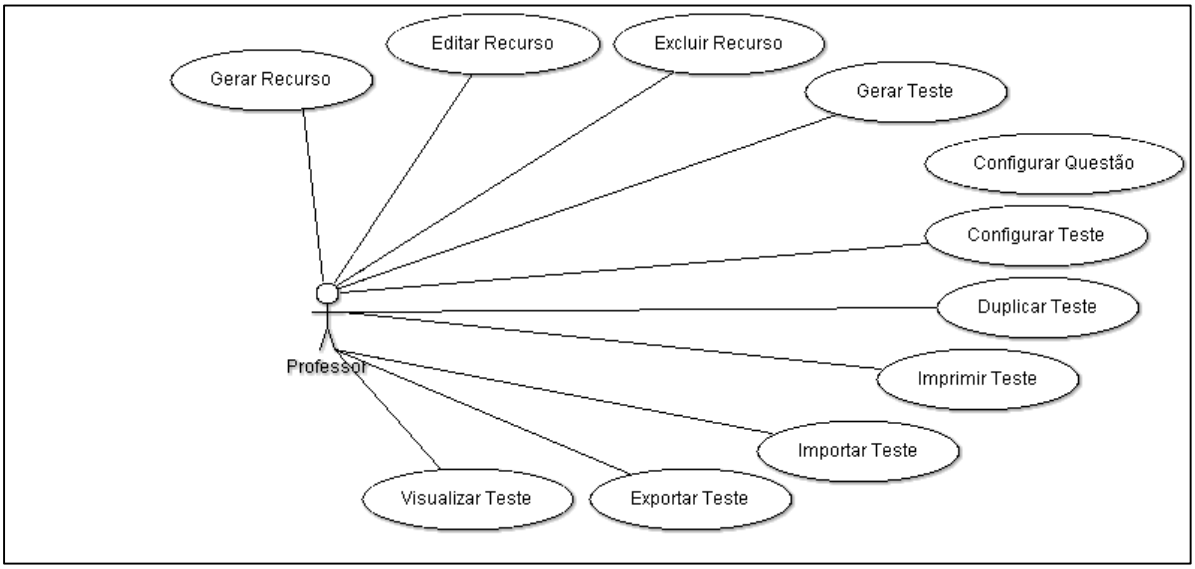

Figura 56. Caso de Uso Ações Professor

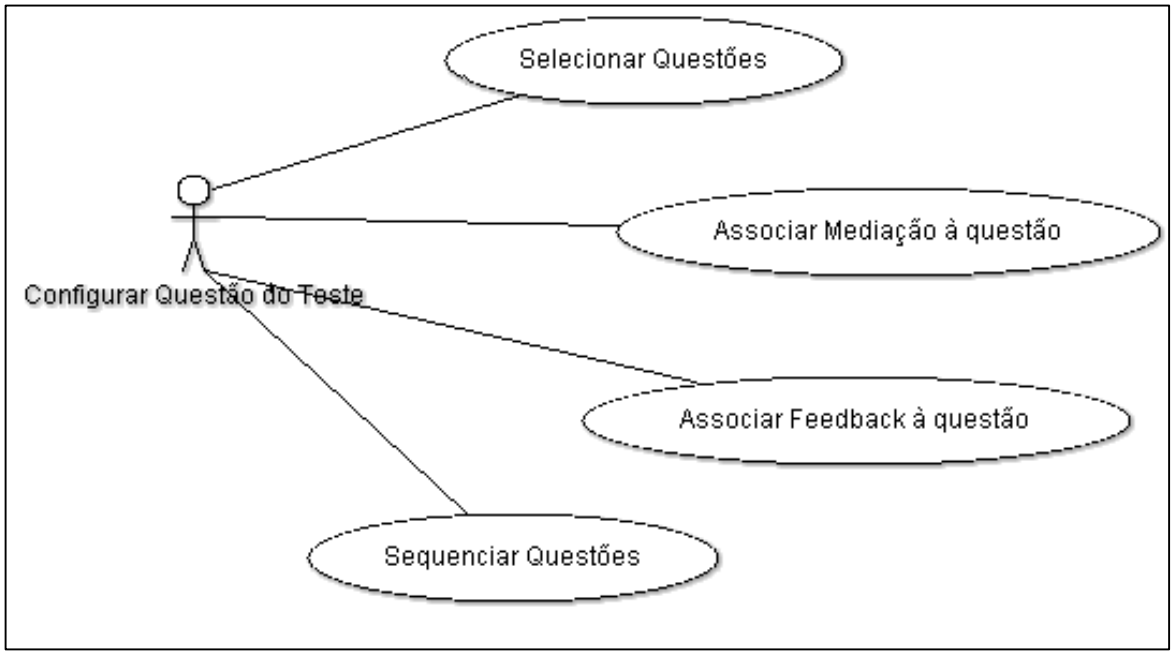

Figura 57. Caso de Uso Configurar Teste da Questão

6.3.2.2 Passo 2.2 – Análise e Detecção de *Hot Spots*

6.3.2.2.1 Hot Spot 1

Nome: variabilidade de tipo de questão.

Descrição1: deve ser permitida a escolha do tipo de questão.

Exemplos de utilização: somatória, objetiva, descritiva, falso ou verdadeiro, entre outros.

Tipo de adaptação requerida: 1 – Habilitar uma característica.

Grau de apoio: suporte padrão.

Descrição 2: deve ser permitida a criação de diferentes tipos de questões.

Exemplos de utilização: criar um novo tipo como extensão de um existente ou diferente dos fornecidos.

Tipo de adaptação requerida: 5 – Adicionar uma característica.

Grau de apoio: ilimitado.

6.3.2.2.2 Hot Spot 2

Nome: variabilidade de tipo de mediação.

Descrição1: deve ser permitida a escolha do tipo de mediação.

Exemplos de utilização: tempo, contextualização, fórmula, imagem, link externo, tópicos, gráficos, link de conteúdo.

Tipo de adaptação requerida: 1 – Habilitar uma característica.

Grau de apoio: suporte padrão.

Descrição 2: deve ser permitida a criação de diferentes tipos de mediação.

Exemplos de utilização: criar um novo tipo como extensão de um existente ou diferente dos fornecidos.

Tipo de adaptação requerida: 5 – Adicionar uma característica.

Grau de apoio: ilimitado.

6.3.2.2.3 Hot Spot 3

Nome: variabilidade de tipo de *feedback*.

Descrição1: deve ser permitida a escolha do tipo de *feedback*.

Exemplos de utilização: informar se a resposta esta correta ou não, apenas informar que está errada sem fornecer a resposta certa, arquivar a resposta e não informar se está ou não correta.

Tipo de adaptação requerida: 1 – Habilitar uma característica.

Grau de apoio: suporte padrão.

Descrição 2: deve ser permitida a criação de diferentes tipos de *feedback*.

Exemplos de utilização: criar um novo tipo como extensão de um existente ou diferente dos fornecidos.

Tipo de adaptação requerida: 5 – Adicionar uma característica.

Grau de apoio: ilimitado.

6.3.2.2.4 Hot Spot 4

Nome: variabilidade de associação de mediação à questão.

Descrição1: deve ser permitida a associação de inúmeras mediações a uma questão. Exemplos de utilização: inserir uma mediação introdutória, por tempo e/ou final.

Tipo de adaptação requerida: 1 – Habilitar uma característica.

Grau de apoio: suporte padrão.

6.3.2.2.5 Hot Spot 5

Nome: variabilidade de associação de *feedback* à questão.

Descrição1: deve ser permitida a associação ou não de *feedback* à questão.

Exemplos de utilização: habilitar ou não um *feedback*.

Tipo de adaptação requerida: 1 – Habilitar uma característica.

Grau de apoio: suporte padrão.

6.3.2.2.6 Hot Spot 6

Nome: variabilidade na montagem da questão.

Descrição1: deve ser permitido o preenchimento da pergunta e resposta.

Exemplos de utilização: informar a pergunta, as opções de resposta e a resposta correta.

Tipo de adaptação requerida: 1 – Habilitar uma característica.

Grau de apoio: suporte padrão.

6.3.2.2.7 Hot Spot 7

Nome: variabilidade no agrupamento de questões.

Descrição1: permitir agrupar quantas questões necessárias.

Exemplos de utilização: selecionar as questões relacionadas a um assunto e criar um módulo.

Tipo de adaptação requerida: 5 – Adicionar uma característica.

Grau de apoio: suporte padrão.

6.3.2.2.8 Hot Spot 8

Nome: variabilidade na geração de avaliação.

Descrição1: deve ser permitida a seleção de questões/módulos.

Exemplos de utilização: selecionar as questões referentes ao conteúdo que o teste irá abordar.

Tipo de adaptação requerida: 5 – Adicionar uma característica.

Grau de apoio: suporte padrão.

6.3.2.2.9 Diagrama de Classes Preliminar

A Figura 58 mostra o diagrama de classes preliminar onde as classes, subclasses e relacionamentos são detalhados.

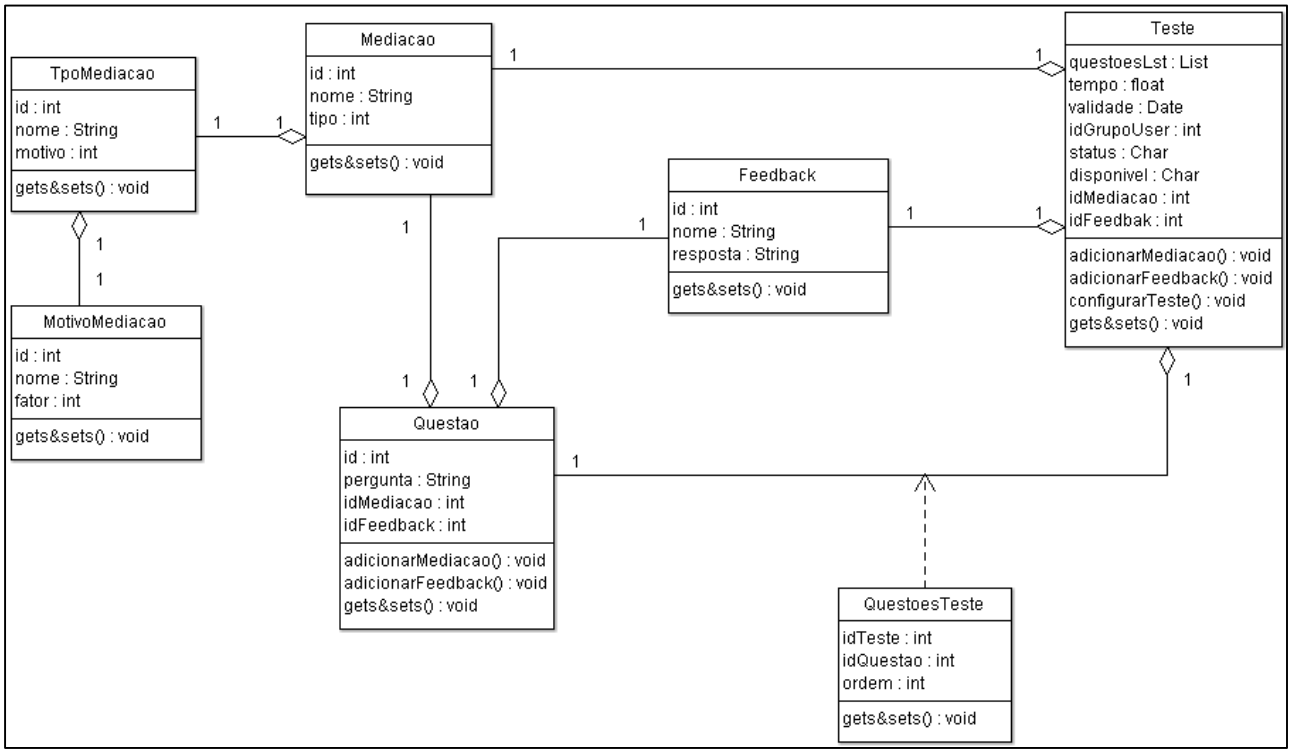

Figura 58. Diagrama de Classe Preliminar

6.3.2.3 Passo 2.3 – Projetar *Framework*

Este passo é composto pelos últimos diagramas antes da implementação do *framework*.

6.3.2.3.1 Diagrama de Classes do Projeto

A Figura 59 determina as camadas onde as classes do projeto se encontram, de acordo com a arquitetura MVC. Em azul está a classe apresentada para o usuário, em verde, as classes de persistência e em branco a classe de controle.

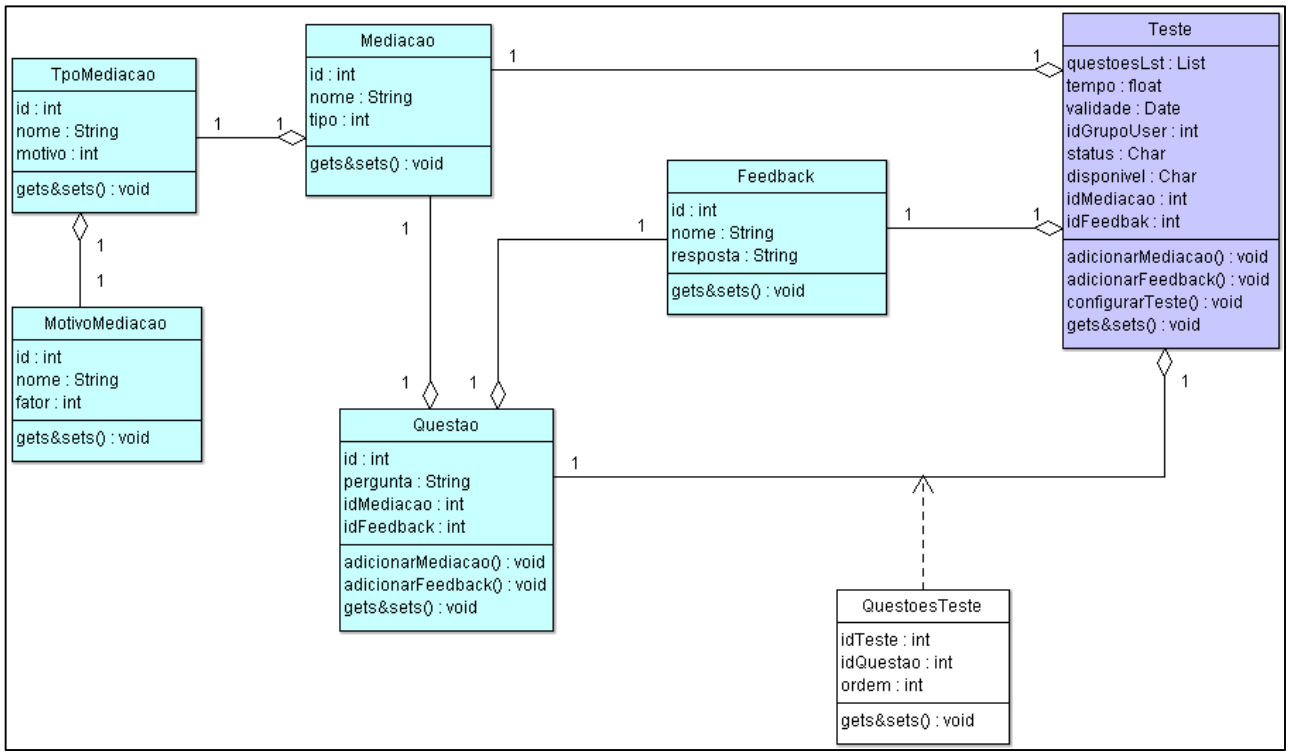

Figura 59. Diagrama de Classes do Projeto

6.3.2.3.2 Diagrama de Navegação

As Figuras demonstram a navegação no *framework* T2A.

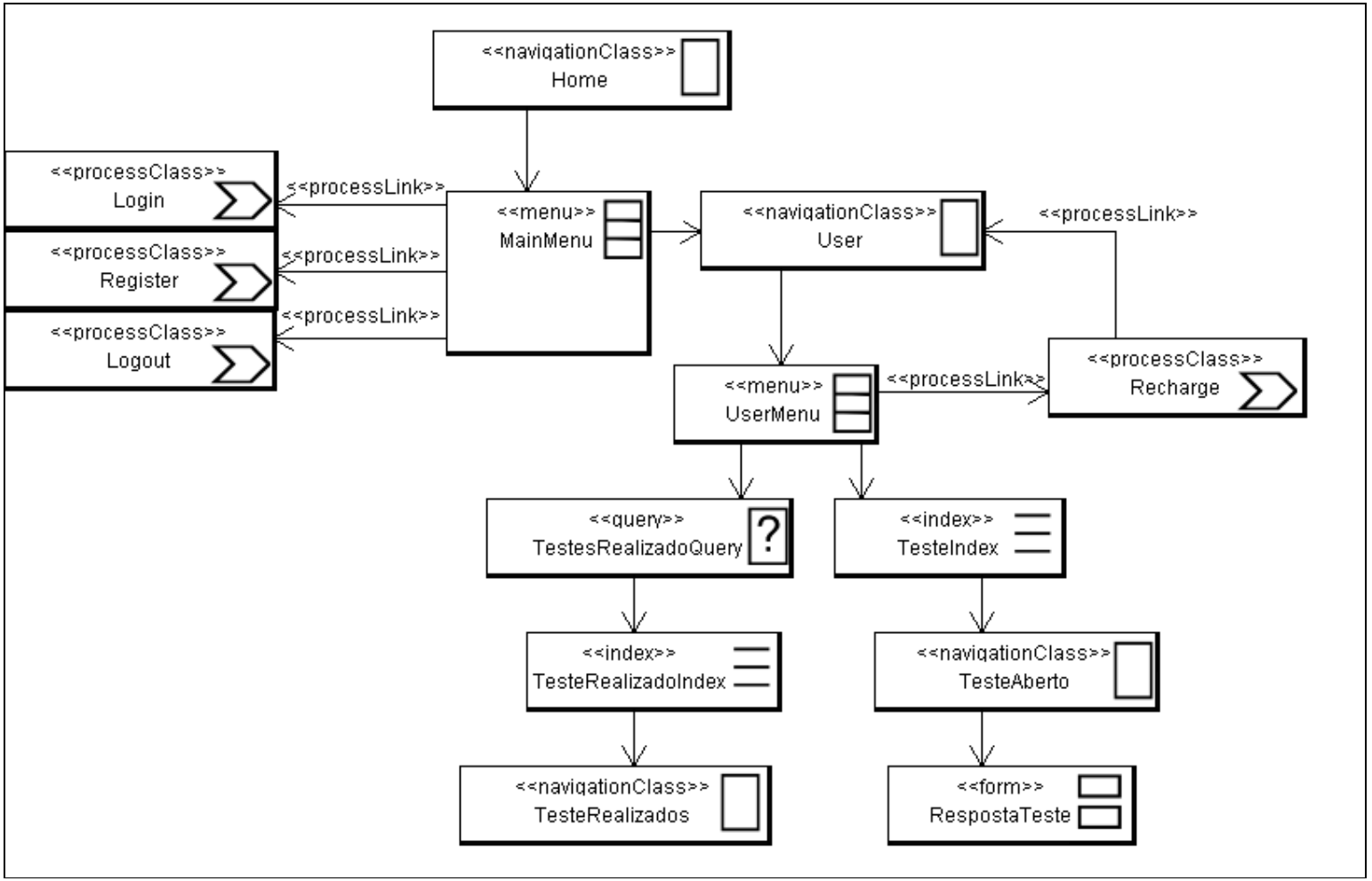

Figura 60. Diagrama de Navegação do Aluno

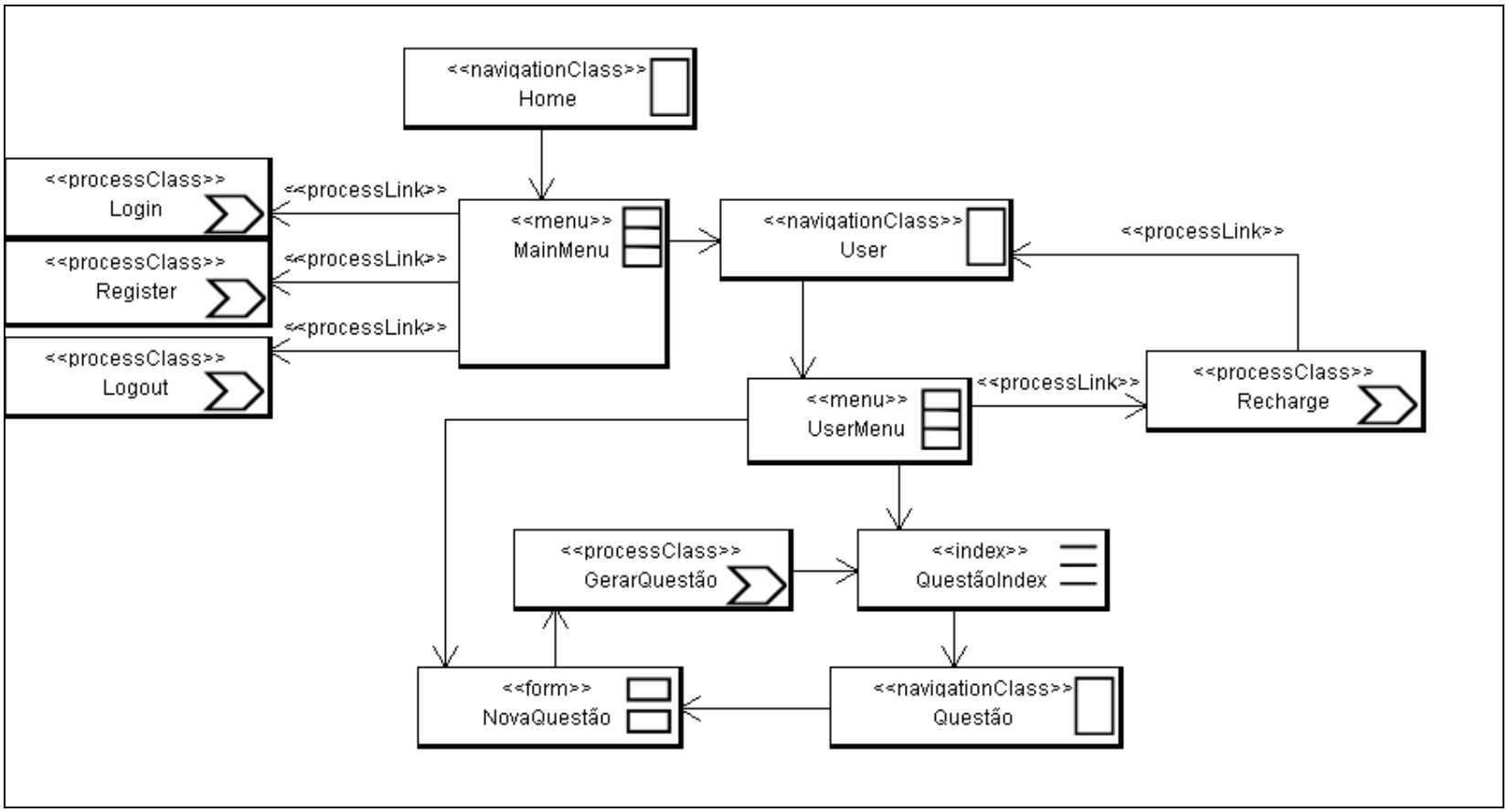

Figura 61. Diagrama de Navegação de Geração de Questões

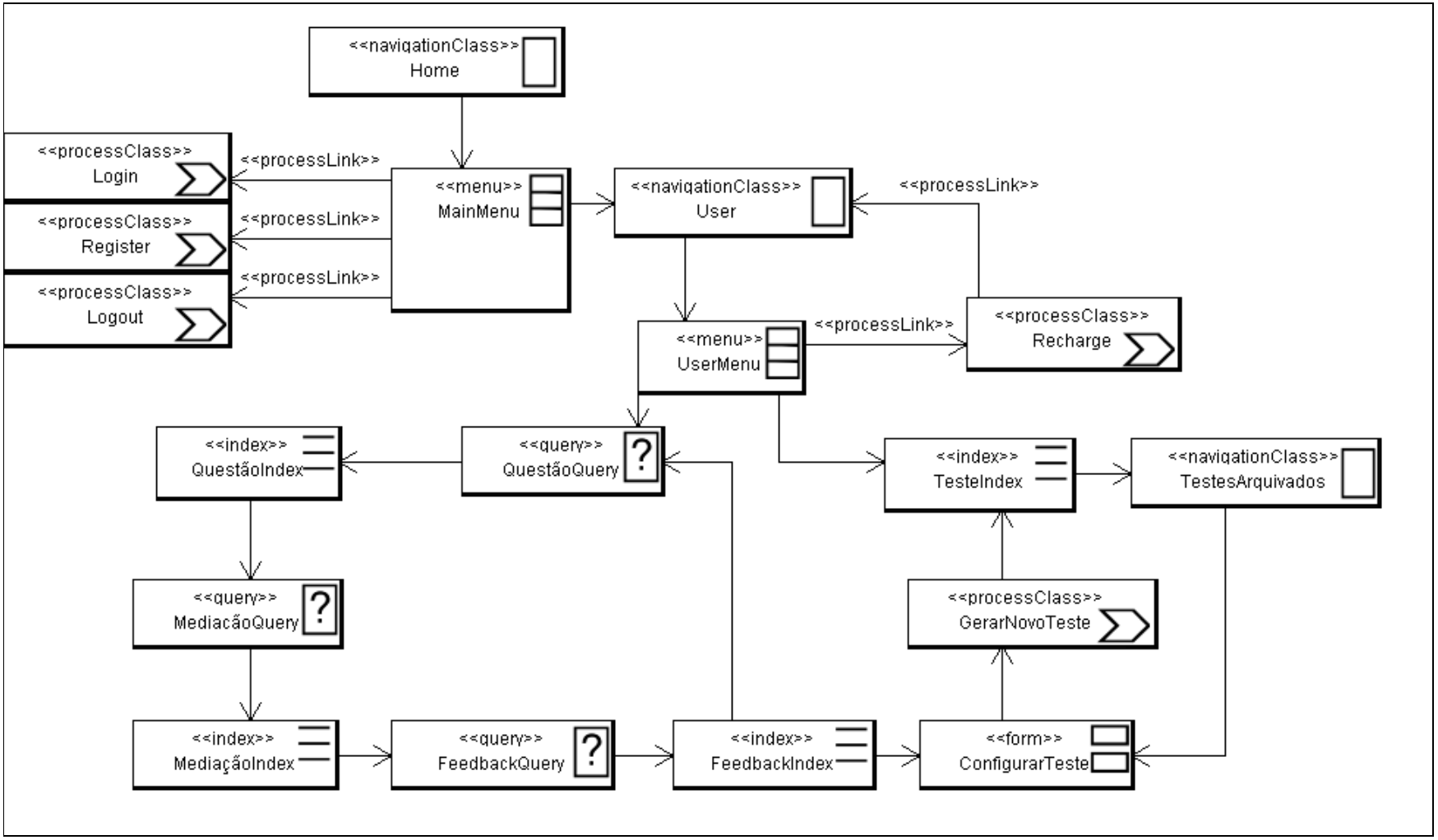

Figura 62. Diagrama de Navegação de Geração de Teste

6.3.2.4 Passo 2.4 – Implementação e Instanciação exemplo

Essa etapa não foi desenvolvida neste trabalho, pois o objetivo não se estendida até a implementação, e sim apenas em modelar o *framework*.

6.3.2.5 Passo 2.5 – Elaborar Roteiro

A elaboração do roteiro é de grande valia para os testes e para o usuário final, pois esse documento serve como um manual indicando os passos a serem seguidos para realizar as tarefas oferecidas pelo *framework*.

6.3.2.5.1 Construir Testes Reutilizando a Estrutura do Framework.

Um teste pode ser composto de questões, *feedback* e mediações. Para construir um teste utilizando o Gerador de Teste é necessário definir e configurar as questões, configurar atributos do teste como mediações, *feedbacks*, validade, status, entre outros.

6.3.2.5.2 Para Criar Testes

A elaboração do teste com o uso do gerador é iniciada com a seleção das questões desejadas, que estejam armazenadas no repositório de recursos.

A configuração das questões consiste em atribuir a cada uma delas mediações e *feedbacks*, se o professor julgar necessário. Por ser flexível, o *framework* não obriga a associação destes atributos a todas as questões.

As configurações do teste consistem na definição de mediações e *feedbacks*; o tempo para resolução; a validade, com definição de prazo, ou seja, até que data o teste pode ser realizado ou prazo indeterminado; os usuários que terão acesso a esse teste; o status do mesmo, podendo ser rascunho, publicado ou fechado; e a disponibilização, que pode ser pública ou privada, característica que permite ou não o uso deste teste por outros professores.

Após estas configurações, defini-se a seqüência das questões de acordo com a estratégia do professor.

6.3.2.5.3 Para Criar Recursos

No Gerador de recursos podem ser criados os recursos: questões, *feedback* e mediações. As seguintes opções podem ser utilizadas para a criação de um recurso:

a) Usar um recurso como modelo: copia-se o recurso que se deseja modificar e altera as configurações do mesmo. Deve-se salvar com um nome diferenciado para evitar que sobrescreva o recurso anterior.
b) Criar um recurso novo: seleciona-se a opção de Novo Tipo, preenche-se as informações obrigatórias para o mesmo e salva.
c) Grupo de Questões: O grupo de questões é um agrupamento por assunto. Selecionam-se as questões que tratam do mesmo assunto e cria-se um grupo para elas. Esse grupo é usado na geração de testes automatizados e/ou aleatórios.

6.3.3 Passo 3 - Instanciar Aplicações

O terceiro passo não foi efetuado por este trabalho, pois abrange apenas a modelagem.

6.4 ALGORITMO

O algoritmo desenvolvido visa reforçar o entendimento do uso do *framework* por meio de um pseudo-código.

```
//--------------gerarTeste---------------
início
  variáveis
         String acao = "";
         List listaQuestao;
         Teste novoTeste;
         Questao questao;

  enquanto acao = "" faça
  inicio-faça
       questao = selecionarQuestao();
       adicionarMediação(questao);
       adicionarFeedback(questao);
       listaQuestao.add(questao);
  fim-faça

  listaQuestoes = sequenciar(listaQuestoes);

  novoTeste.adicionarQuestoes(listaQuestoes);
  configurarTeste(novoTeste);
fim

//--------------funçoes-------------------
funcao adicionarMediação(Questao questao)
inicio
  variaveis
       Mediacao mediacao;

  mediacao = selecionaMediacao();

  questao.mediacao = mediacao;

  retorna questao;
fim
//-----------------------------------------
```

```
funcao adicionarFeedback(Questao questao)
inicio
  variaveis
       Feedback feedback;

  feedback = selecionaFeedback();

  questao.feedback = feedback;

  retorna questao;
fim
//----------------------------------------
funcao adicionarMediação(Teste teste)
inicio
  variaveis
       Mediacao mediacao;

  mediacao = selecionaMediacao();

  teste.mediacao = mediacao;

  retorna teste;
fim
//----------------------------------------
funcao adicionarFeedback(Teste teste)
inicio
  variaveis
       Feedback feedback;

  feedback = selecionaFeedback();

  teste.feedback = feedback;

  retorna teste;
fim
//----------------------------------------
funcao configurarTeste(Teste novoTeste)
inicio
  variaveis
       float tempo;
       Date validade;
       int idGrupoUser;
       char status;
       char disponivel;

  novoTeste = adicionarMediação(novoTeste);
  novoTeste = adicionarFeedback(novoTeste);

  imprimir "Informe tempo de resolução:";
  ler tempo;
```

```
          imprimir "Informe validade do teste:";
          ler validade;

          imprimir "Informe grupo de usuários permitidos:";
          ler idGrupoUser;

          imprimir "Status: (R)ascunho | (P)ublicado |
(F)echado";
          imprimir "Informe status do teste:";
          ler status

          imprimir "Esse teste será liberado para outros
professores (S/N)?";
          ler disponivel;

          novoTeste.tempo = tempo;
          novoTeste.validade = validade;
          novoTeste.idGrupoUser = idGrupoUser;
          novoTeste.status = status;
          novoTeste.disponivel = disponivel;

        fim
        //---------------------------------------
```

6.5 APRESENTAÇÃO E ANÁLISE DOS DADOS

Pode-se afirmar que a principal característica contida no *framework* T2A é a alta flexibilidade que ele oferece na elaboração das questões, partindo desde a criação do tipo de questão e mediação, até a configuração desses dois na avaliação. Diferentemente das ferramentas pesquisadas, que forneciam um número limitado de tipos de questões.

O *framework* proposto possibilita que o usuário possa:

a) criar novos tipos de questões e mediações;

b) habilitar mediações e/ou *feedback* na questão;

c) habilitar mediações e/ou *feedback* na avaliação;

d) gerar um grupo de questões que referenciam o mesmo conteúdo;

e) utilizar esse grupo de questões para a geração automática e/ou aleatória de avaliações;

f) compartilhar os recursos (questões e mediações) com outros usuários do sistema ou deixa restrito ao usuário que os criou;
g) direcionar determinada avaliação a uma ou diversas turmas de estudantes;
h) exportar as avaliações desenvolvidas para um formato de impressão;
i) importar novos recursos em formato específico.

Dessa forma pode-se fazer um comparativo entre as ferramentas pesquisadas e o *framework* proposto como demonstra a tabela 4:

Tabela 4. Comparativo entre Ferramentas de Geração de Teste

	Sisa-*Web*	AvalWeb	WebTest	HotPotatoes	QuestComp	Quiz	T2A
Acesso *On-line*	X	X	X	X	X	X	X
Acesso *Off-line*			X				
Gerar teste automático		X	X			X	X
Agrupar as questões por assunto		X	X	X		X	X
Índice de dificuldade	X	X		X			
Tipos fixos de questões	X	X	X	X	X	X	
Relatórios de desempenho do estudante	X	X	X		X	X	X
***Feedback* a cada questão**			X			X	X
***Feedback* final com a nota do estudante**	X	X	X	X	X	X	X
Necessário *login*	X	X	X	X	X	X	X
Consulta de gabarito e provas anteriores			X	X	X	X	X
Aspecto da pagina configurável				X		X	
Mediação na questão (Ex: ajuda)				X		X	X
Mediação na prova (Ex: tempo de execução)				X			X
Envio de resultados por e-mail				X			
Associar mídias e *links* às questões	X			X		X	X
Questões randômicas			X			X	X
Perfil de usuários limitando o acesso	X	X	X	X	X	X	X
Criar tipos novos de mediações							X
Criar tipos novos de *feedback*							X
Criar tipos novos de questões							X

Alguns aspectos que não estão selecionados na tabela acima ficarão a cargo da implementação, pois a modelagem não chega a este nível de abstração.

CONCLUSÃO

Por meio do levantamento bibliográfico realizado foi possível entender a complexidade do processo de ensino aprendizagem. Compreendeu-se também o emprego de exercícios com tipos variados que visam identificar, não apenas se o estudante absorveu o conteúdo ministrado, mas também outros fatores como compreensão, síntese, aplicação entre outros.

A modelagem de um *framework* comprovou ser a melhor opção para a elaboração de exercícios, que objetivam identificar diversos fatores, pois com tamanha flexibilidade o professor conseguirá aplicar sua metodologia de ensino sem estar limitado a recursos pré-estabelecidos.

O *Framework* T2A abrangeu a maioria das características oferecidas por aplicações na área de elaboração de exercícios de verificação de aprendizado *on-line*, contudo depositou maior atenção ao fator principal de um *framework*: a reutilização. Para isso tornou diversas opções, antes rígidas, flexíveis, como é o caso da criação de tipos de questões e mediações. Assim, o professor pode adaptar seu teste de acordo com a característica da turma.

O presente trabalho propõe diagramas que visam detalhar ao máximo o *Framework* T2A e para isso, empregou-se a UWE que conforme as pesquisas realizadas, foi a metodologia mais indicada para a modelagem de aplicativos *Web* devido aos seus artefatos especializados.

Os resultados obtidos com o emprego das metodologias escolhidas (UWE e o desenvolvimento iterativo e incremental) foram gratificantes, tendo em vista que os diagramas expressam os detalhes necessários para uma posterior implementação e a metodologia de desenvolvimento tornou a construção dessa modelagem algo efetivo e funcional.

Como prosseguimento deste trabalho propõem-se a implementação do *Framework* T2A e posteriormente a integração destes a um ambiente virtual. Outra proposta seria o

emprego de agentes inteligentes que possam monitorar as ações dos usuários e fazer uso do histórico gerado pelo *framework* para sugerir ao professor à metodologia que será mais efetiva no combate a determinada dificuldade encontrada pelos estudantes.

Juntamente à implementação é de grande valia desenvolver a documentação para o *framework* de modo que facilite a aplicação do *framework* para comprovar sua eficiência ou não, identificar dificuldades encontradas na utilização do mesmo para a elaboração de testes por parte dos professores.

Por outro lado é interessante verificar se as mediações e *feedbacks* estão sendo efetivo no seu objetivo que é auxiliar o estudante no seu processo de ensino aprendizagem. Para isso deve-se acompanhar a utilização do mesmo por um tempo determinado e fazer o levantamento dos resultados.

REFERÊNCIAS

ABRÃO, Iran Calixto; ABRÃO, Maria Adriana Vidigal Lima; RAYEL, Felipe. **QUESTCOMP Ambiente via WEB para Auxílio na Avaliação de Ensino-Aprendizagem à Distância.** PUC-Minas. Poços de Caldas, 2003. Disponível em: <http://www.rayel.org/PaperCLEI2003-QuestComp.pdf>. Acesso em: 10 jun. 2009.

ABREU, Guilherme R.; LARA, Rafael B.; RAGAZZO, Rafael S. **UWE UML-Based Web Engineering**. 2007 Disponível em: <www.dc.ufscar.br/~rosangel/mds/Seminarios/2007/UWE-Slides.ppt>. Acesso em: 04 mai. 2009.

ANASTASI, Anne; URBINA, Susana. **Testagem psicológica.** 7. ed. Porto Alegre: Artes Médicas, 2000. 575 p.

ANASTASIOU, Lea das Graças; ALVES, Leonir Pessate (orgs). **Processos de Ensinagem na Universidade**: pressupostos para as estratégias de um trabalho em aula. 3. ed. Joinville: Univille, 2003. 146 p.

ARANGO, Guillermo; PRIETO-DIAZ, Ruben. **Domain analysis concepts and research direction**. In: DOMAIN ANALYSIS AND SOFTWARE MODELING. Los Alamitos: IEEE Computer Society Press, 1991. p. 9-26.

BASTIN, Georges. **As Técnicas Sociométricas.** 2. ed. Lisboa: Moraes Editores, 1980. p. 15-19.

BARBOSA, Jane Rangel Alves. **A Avaliação da Aprendizagem como Processo Interativo:** Um Desafio para o Educador. Democratizar, v.II, n.1, jan./abr. 2008. Disponível em: <http://www.faetec.rj.gov.br/isezonaoeste/publicacoes/democratizar/ed2/artigo_jane2.pdf>. Acesso em: 10 jun. 2009.

BARTOLOMEIS, Francesco de. **Avaliação e orientação: objetivos, instrumentos e métodos.** Lisboa: Livros Horizontes, 1977.

BAUMEISTER, Hubert; KOCH, Nora; MANDEL, Luis. **Towards a UML extension for hypermedia design**. In UML99 The Unified Modeling Language - Beyond the Standard, LNCS 1723, Fort Collins, USA, Springer. 1999.

BLOOM, Benjamin S. **Taxonomia de objetivos educacionais: domínio cognitivo**. Porto Alegre: Globo, 1972.

BLOOM, Benjamin S. et al. **Taxionomia de Objetivos Educacionais e Domínio Cognitivo:** Domínio Cognitivo. Porto Alegre: Globo, 1983. 1 v.

BONNIOL, Jean Jacques. **Déterminants et mécanismes des comportements d.** 1981. Tese (Doutorado) - Universidade de Bordeaux Ii, Bordeaux, 1981.

BOOCH, Grady; JACOBSON, Ivar; RUMBAUGH, James. **The Unified Software Development Process.** Massachusetts: Addison-wesley Professional, 1999. 512 p.

BROWN, Alan W.; SHORT, Keith. **On Components and Objects: The Foundation of Component-Based Development**. In: INTERNATIONAL SYMPOSIUM ON ASSESSMENT OF SOFTWARE TOOLS AND TECHNOLOGIES (SAST 97), 5, 1997. Pittsburgh, PA: IEEE Press, 1997. p.112-121

BUSCHMANN, Frank et al. **Pattern-Oriented Software Architecture:** A System of Patterns. Massachusetts: John Wiley & Sons, 1996. 476 p.

CARNEIRO, Cristiane Marise Pérez Da Silva. **FRAMEWORKS DE APLICAÇÕES ORIENTADAS A OBJETOS – UMA ABORDAGEM ITERATIVA E INCREMENTAL.** 2003. 110 f. Dissertação (Mestrado) - Universidade Salvador, Salvador, 2003.

CERI, Stefano et al. **Web Modeling Language (WebML): a modeling language for design sites**. In: 9TH INTERNATIONAL WORLD WIDE WEB CONFERENCE THE WEB: THE NEXT GENERATION, 15-19 de Maio 2000, Amsterdan. Anais eletrônico em <http://www9.org/w9cdrom/177/177.html> Acessado em 10 jun. 2009. Amsterdan, 2000.

CERI, Stefano et al. **Designing Data-Intensive Web Applications.** Amsterdan: Morgan Kaufmann Publishers, 2004. 562 p.

COFFMAN, William E. **Achievement tests**. In: MITZELL, H.E. ENCYCLOPEDIA OF EDUCATIONAL RESEARCH. Nova York. Macmillan Publishing Co, p. 7-15, 1964

CONALLEN, Jim. **Modeling Web Application Architectures with UML.** Communications of the ACM, 1999. Disponível em: <http://www.wthreex.com/rup/papers/pdf/webapps.pdf>. Acesso em: 10 jun. 2009.

CONALLEN, Jim. **Desenvolvendo Aplicações Web com UML.** Rio de Janeiro: Campus, 2003. 476 p.

PINTO, Sérgio Crespo C. da Silva. **Composição em WebFrameworks.** 2000. Tese (Doutorado) - Departamento de Informática, Puc Rio, Rio de Janeiro, 2000.

DEMO, Pedro. **Avaliação Qualitativa. 6**. ed. Campinas: Autores Associados, 1999. 102 p.

ESTEBAN, Maria Teresa. **Que Sabe Quem Erra? Reflexões Sobre Avaliação e Fracasso Escolar.** 3. ed. Rio de Janeiro: Dp&a / Lamparina, 2001. 200 p.

FAYAD, Mohamed; SCHMIDT, Douglas C. **Object-oriented application frameworks.** New York: Acm, 1997. 32-38 p. (Communications of the ACM).

FAYAD, Mohamed; SCHMIDT, Douglas C.; JOHNSON, Ralph E. **Implementing Application Frameworks: Object-Oriented Frameworks at Work.** New York: John Wiley & Sons, 1999a. 729 p.

FAYAD, Mohamed; SCHMIDT, Douglas C.; JOHNSON, Ralph E. **Building Application Frameworks: Object-Oriented Foundations of Framework Design.** New York: John Wiley & Sons, 1999b. 688 p.

FAYAD, Mohamed; JOHNSON, Ralph E. **Domain-specific application frameworks:** frameworks experience by industry. New York: John Wiley & Sons, 2000. 681 p.

AVALIAÇÃO: Tendências e Tendenciosidades. Rio de Janeiro: Ensaio: Revista da Fundação Cesgranrio, v. 1, n. 2, 1994. Thereza Penna Firme.

FONS, Joan; PELECHANO, Vicente; PASTOR, Oscar. **Extending an OO Method to Develop Web Applications. 2001.** Disponível em: <http://www2003.org/cdrom/papers/poster/p329/p329-fons.htm>. Acesso em: 12 jun. 2009.

FROEHLICH, Garry et al. **Hooking into Object-Oriented Application Frameworks.** 1997a. Disponível em: <http://www.cs.ualberta.ca/~softeng/papers/icse10.pdf>. Acesso em: 12 jun. 2009.

FROEHLICH, Garry et al. **Reusing Application Frameworks Through Hooks.** 1997b. Disponível em: <http://www.cs.ualberta.ca/~softeng/papers/af1051.pdf>. Acesso em: 12 jun. 2009.

GAMMA, Erich et al. **Design Patterns:** Elements of Reusable Object-Oriented Software. Portland, Oregon: Addison-wesley Professional, 1995. 416 p. (Addison Wesley Professional Computing).

GIMENES, Itana Maria de Souza; HUZITA, Elisa Hatsue Moriya. **Desenvolvimento Baseado em Componentes:** Conceitos e Técnicas. Rio de Janeiro: Ciencia Moderna, 2005. 304 p.

GONÇALVES, Mariangela Barbi. **Programa de educação continuada no seio cavernoso**. Dissertação (Mestrado). Universidade Federal do Estado do Rio de Janeiro. Centro de Ciências Biológicas e da Saúde. Mestrado em Neurologia, 2006.

HADJI, Charles. **Avaliação Desmistificada.** Porto Alegre: Artmed, 2001. 136 p.

HAYDT, Regina Célia Cazaux. **Avaliação do Processo Ensino-Aprendizagem**. São Paulo: Ática, 1998. 160 p.

HAYDT, Regina Célia Cazaux. **Curso de Didática Geral.** São Paulo: Atica, 2000. 327 p.

HAYDT, Regina Célia Cazaux. **Curso de didática geral**. São Paulo: Ática, 2003. **322** p.

HENNICKER, Rolf; KOCH, Nora. **A UML-based Methodology for Hypermedia Design Method. 2000.** Disponível em: <http://www.pst.informatik.uni-muenchen.de/~kochn/Uml2000.pdf>. Acesso em: 12 jun. 2009. .

HENNICKER, Rolf; KOCH, Nora. **Modeling the User Interface of Web Applications with UML**. In: PRACTICAL UML-BASED RIGOROUS DEVELOPMENT METHODS - COUNTERING OR INTEGRATING THE EXTREMISTS**,** Workshop of the pUML - Group at the UML 200. 2001.

HOFFMANN, Jussara Maria Lerch. **Avaliação mediadora: uma relação dialógica na construção do conhecimento. 1994.** Disponível em: <http://www.crmariocovas.sp.gov.br/int_a.php?t=008>. Acesso em: 12 jun. 2009.

HOFFMANN, Jussara Maria Lerch. **Avaliação mediadora: uma prática em construção da pré-escola à universidade. 1998.** Disponível em: <http://www.diaadia.pr.gov.br/cfc/arquivos/File/Grupo%20de%20Estudo%20Arte/Ensino%20Religioso/4encontro.pdf>. Acesso em: 12 jun. 2009.

HOFFMANN, Jussara. **Avaliação:** mito e desafio. 36. ed. Porto Alegre: Mediação, 2005. 104 p.

JOHNSON, Ralph E.; FOOTE, Brian. **Designing Reusable Classes.** Journal of Object-Oriented Programming June/July 1988, Volume 1, Number 2, pages 22-35. Disponível em: <http://www.laputan.org/drc/drc.html>. Acesso em: 12 jun. 2009.

JOHNSON, Ralph. **Reusing Object-Oriented Design.** University of Illinois, Technical Report UIUCDCS 91-1696, 1991.

JOHNSON, Ralph. **How to Design Frameworks**. In: CONFERENCE ON OBJECT ORIENTED PROGRAMMING, SYSTEMS, Languages and Systems. 1993.

JOHNSON, Ralph E. **Frameworks = (components + patterns).** New York: Acm, 1997. 39-42 p. Communications of the ACM. Volume 40 , Issue 10 (October 1997).

JUSTULIN, Fernando; COSTA, Kelton Augusto Pontara da; BARROS, Renato Correia de. **O Perfil das Ferramentas de Avaliação para Instrução Baseada na Web**. Disponível em: <http://inf.unisul.br/~ines/workcomp/cd/pdfs/2354.pdf>. Acesso em: 10 jun. 2009.

KNAPP, Alexander et al. **Modeling Business process in Web applications with ArgoUWE. 2004.** Disponível em: <http://www.pst.ifi.lmu.de/veroeffentlichungen/knapp-et-al:uml:2004.pdf>. Acesso em: 12 jun. 2009.

KOCH, Nora et al. **Extending UML to Model Navigation and Presentation in Web Applications**. In: THE UML WORKSHOP (UML'2000), outubro, 2000.

KOCH, Nora; KRAUS, Andreas; HENNICKER, Rolf. **The Authoring Process of the UML-based Web Engineering Approach**. In: Daniel Schwabe (editor), FIRST

INTERANATIONAL WORKSHOP ON WEB-ORIENTED SOFTWARE TECHNOLOGY (IWWOST01), publicação on-line, junho, 2001.

KOCH, Nora; KRAUS, Andreas. **The expressive Power of UML-based Web Engineering**. In: SECOND INTERNATIONAL WORKSHOP ON WEB-ORIENTED SOFTWARE TECHNOLOGY. D. Schwabe, O. Pastor, G. Rossi, and L. Olsina, editors. 2002.

KRATHWOHL, David R.; BLOOM, Benjamin S.; MASIA, Bertram B. **Taxonomy of Educational Objectives:** The Classification of Educational Goals Handbook II: Affective Domain. New York: David Mckay Co., 1964.

LEARNLOOP. **Manual LearnLoop**. 2007. Disponível em: <http://www.ead.unesc.net/learnloop_man>.Acessado em: 20/05/2009.

LIMA, Fernanda; SCHWABE, Daniel. **Application Modeling for the Semantic Web.** 93-102 p. In:Web Congress, 2003. Proceedings. First Latin American, Santiago, Chile, IEEE-CS Press, 2003.

LMU – LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN. 2009. **About UWE:** An approach based on standards. Disponível em: <http://www.pst.informatik.uni-muenchen.de/projekte/uwe/aboutUwe.html>. Acesso em: 12 jun. 2009.

LOPES, Rui et al. **Arquitecturas XML para Concretização de Modelos Hipermédia.** Faculdade de Ciências da Universidade de Lisboa . 2003. Disponível em: <http://hcim.di.fc.ul.pt/hcimwiki/images/6/66/Rlopes-capsi-2003.pdf>. Acesso em: 12 jun. 2009.

LUCKESI, Cipriano Carlos. **Avaliação Da Aprendizagem Escolar.** 18. ed. São Paulo: Cortez, 2003. 182 p.

MATTSSON, Michael. **Object-oriented Frameworks - A survey of methodological issues.** In> TECHNICAL REPORT, LU-CS-TR: 96-167, DEPARTMENT OF COMPUTER SCIENCE, Lund University, 1996.

MATTSSON, Michael. **Evolution and Composition Object-Oriented Frameworks.** 2000. Tese (Phd) - Departamento de Software Engineering And Computer Science, University Of Karlskrona, Karlskrona, 2000.

MEDIANO, Zélia Domingues. **Módulos instrucionais para medidas e avaliação em educação**. 2. ed. Rio de Janeiro: Livraria Francisco Alves, 1977.

MELCHIOR, Maria Celina. **Avaliação pedagógica:** função e necessidade. 2. ed. Porto Alegre: Mercado Aberto, 1999. 150 p.

MEZZAROBA, Leda; ALVARENGA, Georfrávia Montoza. **A trajetória da avaliação educacional no Brasil**. In: AVALIAR: um compromisso com o ensino e a aprendizagem. Londrina: Núcleo de Estudos e Pesquisas em Avaliação Educacional, 1999, p29-81.

MOURA, Sabrina Silva de; SCHWABE, Daniel. **Interface Development for Hypermedia Applications in the Semantic Web.** Departamento de Informática, PUC-Rio. 2004 . Disponível em: <http://www.w3.org/2008/10/mbui/SSMoura_LAWEB2004_Final.pdf>. Acesso em: 12 jun. 2009.

FONS, Joan, et al. **Development of Web applications from Web enhanced conceptual schemas.** In: PROCEEDINGS OF THE 22TH INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING, LNCS, Chicago, IL, USA, October 13-16, vol. 2813, Springer, Berlin, 2001.

OBERON MICROSYSTEMS. **BlackBox.** Disponível em: <http://www.oberon.ch/blackbox.html>. Acesso em: 12 jun. 2009.

APPLE INC. **Documentação do framework de componentes OpenDoc.** Disponível em: <http://developer.apple.com/documentation/macos8/Legacy/OpenDoc/opendoc.html>. Acesso em: 12 jun. 2009.

PINTO, Sérgio Crespo Coelho da Silva. Composição em WebFrameworks**.** 2000. 150 f. Tese (Doutorado) - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2000.

PASTOR, Oscar et. al. **The OO-Method Approach for information systems Modelling:** From Object-Oriented Conceptual Modeling to Automated Programing. Information Systems, Volume 26, Numero 7, pp. 507-534, Nov. de 2001.

PELECHANO, Vicente et al. **Developing Web Applications from Conceptual Models**. A Web Services Approach. In: ECOMO03 - MCYT Project with ref. TIC2001-3530-C02-01, Universidad Politecnica de Valencia, Espanha, 2003.

PESSOA, José Marques; MENEZES, Crediné Silva de. **Um Framework para Construção Cooperativa de Ambientes Virtuais de Aprendizagem na Web**. 2003. Disponível em: <http://www.nce.ufrj.br/sbie2003/publicacoes/paper28.pdf>. Acesso em: 06 mar. 2009.

PRATA, David Nadler. **Estratégias para o Desenvolvimento de um Framework de Avaliação da Aprendizagem a Distância**. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 14., 2003, Recife. Artigo. Rio de Janeiro: Nce - Im / Ufrj, 2003. p. 1 - 10. Disponível em: <http://www.nce.ufrj.br/sbie2003/publicacoes/paper16.pdf>. Acesso em: 27 abr. 2009.

PREE, Wolfgang. **Design Patterns for Object-Oriented Software Development.** Linz, Austria: Addison Wesley Longman, 1995. 268 p.

PREE, Wolfgang. Hot-Spot-Driven Development. In: FAYAD, Mohamed; SCHMIDT, Douglas C. (Eds.). **Building Application Frameworks: Object-Oriented Foundations of Framework Design.** ... John-Wiley & Sons, 1999.

QUEIROZ, Vera Cristina. **Curso em ambiente virtual de aprendizagem:** canteiro para germinação de comunidade de aprendizagem on-line. 2005. 270 f. Tese (Doutorado) - Faculdade de Educação (FE), São Paulo, 2005. Disponível em: <http://www.teses.usp.br/teses/disponiveis/48/48134/tde-23102007-142652/>. Acesso em: 15 jun. 2009.

RIBEIRO, Lucie Carrilho. **Avaliação da aprendizagem.** Lisboa: Texto Editora, 1999. 226 p.

ROCHA, Paulo Rogério Souza; COSTA, Heitor Augustus Xavier. **Modelagem do Software Carteira de Trabalho On-line Usando UWE.** Disponível em: <http://www.dcc.ufla.br/infocomp/artigos/v3.1/art11.pdf>. Acesso em: 15 jun. 2009.

ROMANOWSKI, Joana Paulin; WACHOWICZ, Lílian Anna. Avaliação formativa no ensino superior: que resistências manifestam os professores e os estudantes?. In: ANASTASIOU, Leia das Graças Camargo e ALVES, Leonir Pessate. **Processos de ensinagem na universidade: pressupostos para as estratégias de trabalho em aula**. Joinville, SC: UNIVILLE, 2003, p. 121-139

SACRISTAN, J. Gimeno. **Compreender E Transformar O Ensino.** 4. ed. Porto Alegre: Artmed, 1998. 398 p.

SALINAS, Dino. **Prova amanhã! A avaliação entre a teoria e a realidade.** Porto Alegre: Artmed, 2004. 128 p.

SANT´ANNA, Ilza Martins. **Por que avaliar? Como avaliar? Critérios e instrumentos.** 9. ed. Petrópolis: Vozes, 1995. 137 p.

SAUL, Ana Maria Avela. **A avaliação educacional.** Série Idéias n. 22, São Paulo: FDE, 1994. p. 61-68.. Disponível em: <http://www.crmariocovas.sp.gov.br/pdf/ideias_22_p061-068_c.pdf>. Acesso em: 15 jun. 2009.

SCHMIDT, Douglas C.; FAYAD, Mohamed; JOHNSON, Ralph E. **Software patterns.** 1997. Volume 39 , Issue 10. p. 37 - 39 . Disponível em: <http://portal.acm.org/ft_gateway.cfm?id=236164&type=pdf&coll=GUIDE&dl=GUIDE&CFID=39595424&CFTOKEN=81697511>. Acesso em: 15 jun. 2009.

SCHWABE, Daniel; ROSSI, Gustavo. **An Object Oriented Approach to Web-Based Application Design**. Rio de Janeiro, 1998. Disponível em: <http://www.telemidia.puc-rio.br/oohdm/oohdm.html>. Acesso em: 15 jun. 2009.

SILVA, Maria Urbana da. **Evolução histórica da avaliação**: breve relato. Disponível em: <http://www.uninova.edu.br/UniNova/Arquivos_Downloads/materialum.ppt>. Acesso em: 10 jun. 2009.

SOMMERVILLE, Ian. **ENGENHARIA DE SOFTWARE.** 8. ed. São Paulo: Pearson / Prentice Hall, 2007. 568 p.

SZYPERSKI, Clemens; BOSCH, Jan; WECK, Wolfgang. **Sumary of the Second International Workshop on Component-Oriented Programming**. In: SECOND INTERNATIONAL WORKSHOP ON COMPONENT-ORIENTED PROGRAMMING (WCOP'97), 1997, Jyväskylä, Finlândia.

TALIGENT. **Building Object-Oriented Frameworks**, A Taligent White Paper. Technical report, Taligent Inc. 1994

TEIXEIRA, Gilberto. **Avaliação da Aprendizagem**. Disponível em: <http://www.serprofessoruniversitario.pro.br/ler.php?modulo=4&texto=81>. Acesso em: 09 mai. 2009.

VASCONCELLOS, Celso dos Santos. Reprovação: concretização da logica da exclusao na escola. In: **Revista de Educaçao AEC –** Nº. 100/1996, p. 167-187. VEIGA, I. P. A. Ensino e avaliação: uma relacao intrínseca 'a organizacao do trabalho pedagógico. In. VEIGA, I. P. A. (Org.). **Didática:** o ensino e suas relacoes. Campinas, SP: Papirus, 1996, p 149-169.

VIANNA, Heraldo Marelim. **Termos Técnicos Em Medidas Educacionais.** São Paulo: Fundação Carlos Chagas, 1981.

VIANNA, Heraldo Marelim. **Testes em educação.** São Paulo: Ibrasa, 1987.

VILCACHAGUA, Oscar Dantas et al. **SISTEMA MODULAR DE AVALIAÇÃO DA APRENDIZAGEM VIA WEB:** WEBTEST. Escola Politécnica da Universidade de São Paulo, 2001. Disponível em: <http://rmav-sp.larc.usp.br/Documentos/paper_webtest.pdf>. Acesso em: 15 jun. 2009.

VILJAMAA, Antti. **Pattern-Based Framework Annotation and Adaptation:** A systematic approach. 2001. 118 f. Tese (Doutorado) - University Of Helsinki, Helsinki, 2001. Disponível em: <http://practise.cs.tut.fi/files/publications/Fred/AVi_thesis.zip>. Acesso em: 01 mar. 2009.

VILLAS BOAS, Benigna Maria de Freitas. A avaliacao formativa: em busca do desenvolvimento do estudante, do professor e da escola. In: VEIGA, Ilma Passos Alencastro. **As dimensoes do projeto político-pedagogico:** novos desafios para a escola. Campinas, SP: Papirus, 2001. p. 175-212.

ZAINA, Luciana Aparecida Martinez. **Acompanhamento do aprendizado do estudante em cursos a distância através da Web: metodologias e ferramenta**. 2002. 169 f. Tese (Mestrado) - Escola Politécnica da Universidade de São Paulo, São Paulo, 2002.

ZAFIRIS, Paraskevas A. et al. **A Practitioners's Approach to Evolving and Remodeling Large-Scale WWW Sites.** 34th Annual Hawaii International Conference on System Sciences (HICSS-34) Volume 7 . 2001 . Disponível em:

<http://csdl2.computer.org/comp/proceedings/hicss/2001/0981/07/09817078.pdf>. Acesso em: 15 jun. 2009.