Carlos Henrique Pereira

WebUML: Uma Ferramenta Colaborativa de Apoio ao Projeto e Análise de Sistemas Descritos em Classes UML

Florianópolis - SC

2007 / 2

Carlos Henrique Pereira

WebUML: Uma Ferramenta Colaborativa de Apoio ao Projeto e Análise de Sistemas Descritos em Classes UML

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação

Orientador:
Prof. José Eduardo De Lucca

Banca:
Rafael Savi
Wagner Saback Dantas

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE SISTEMAS DE INFORMAÇÃO

Florianópolis - SC

2007 / 2

Carlos Henrique Pereira

WebUML: Uma Ferramenta Colaborativa de Apoio ao Projeto e Análise de Sistemas Descritos em Classes UML

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação

Prof. José Eduardo De Lucca
Orientador

Rafael Savi
Banca examinadora

Wagner Saback Dantas
Banca examinadora

Resumo

O objetivo deste trabalho é especificar e elaborar uma ferramenta de auxílio ao desenvolvimento colaborativo de *software*, que apóie a análise e a implementação de *software* livre.

A ferramenta apresenta funcionalidades que permitem a criação e manipulação de especificações gráficas de artefatos sugeridos pela UML, sendo um passo importante para dar suporte a desenvolvimento distribuído de *software*.

A ferramenta abordada nesse projeto disponibiliza uma interface para que os usuários elaborem diagramas de classes seguindo o padrão especificado pela UML, no final da elaboração desses diagramas o usuário poderá gerar um arquivo XMI com a especificação do seu diagrama de classe.

O desenvolvimento e elaboração dessa ferramenta se focaram da utilização de ferramentas e padrões não proprietários. Utilizou-se **Eclipse** como Ide para desenvolvimento da ferramenta e **SVG**, **JavaSript** e **XMI** como linguagem base. O **Umbrello** foi utilizado com como *software* para geração de código fonte, a partir de XMI gerado pela ferramenta desenvolvida.

Palavras-chave: UML, Diagrama de classe, Web 2.0, SVG e XMI

Abstract

The purpose of this work is to specify and delevop a tool that supports the collaborative software development process which encompass the phases of analysis and implementation of open-source softwares.

The tool has features that allows the creation and manipulation of graphical elements from the UML specification, being a important step towards the support for the collaborative software development community.

The tool presented in this work provides to the user a graphical interface that enables the design of class diagrams following the UML specification. At the end of the process of design, the user can generate a XMI file with the diagram specification.

This work was developed using non-proprietary softwares and patterns. It uses **Eclipse** as the IDE for the developmet process and **SVG**, **Javascript** and **XMI** as the main languages. The software **Umbrello** is used to generate the source code based on the XMI specification provided by the tool.

Keywords: UML, Diagrama de classe, Web 2.0, SVG e XMI

Lista de Figuras

Lista de Tabelas

Lista de Códigos

Sumário

1 INTRODUÇÃO

1.1 PROBLEMA

O desenvolvimento colaborativo de *software* necessita de ferramentas que possam auxiliar na análise e implementação. Uma necessidade desse desenvolvimento é a possibilidade de especificar o *software* a ser desenvolvido, e compartilhar essa especificação entre o pessoal envolvido nos processos de desenvolvimento colaborativo de *software*.

Algumas ferramentas auxiliam a especificação de software com funcionalidades de criação e manipulação de gráficos de classes UML (ArgoUML[1], Umbrello [2], Jude [3] e *Enterprise Architect* [4]), mas não suportando um ambiente de trabalho colaborativo. Esta ausência de funcionalidade pode comprometer o andamento ágil, consistente e objetivo da etapa de planejamento de sistemas (inclusive de código aberto) segundo um modelo de desenvolvimento com vários indivíduos no qual um ambiente comum de desenho e documentação de especificações seria mais favorável por oferecer compartilhamento e maior aproveitamento das informações relativas ao sistema (evitando atrasos, ruídos e perda das informações) que são essenciais àqueles que participam do seu desenvolvimento.

Existindo uma necessidade de uma arquitetura que minimize a ausência de ferramentas para o desenvolvimento colaborativo de *software*. Possibilitando o intercambiar de informações entre o os desenvolvedores de *software*, utilizando um conjunto de ferramentas que agrega as especificações de Web 2.0 e UML.

[1] ArgoUML é uma ferramenta para desenhar UML com suporte cognitivo, licenciado sobre a BSD (TIGRIS, 2007).

[2] Umbrello UML Modeller é um modelador unificado para diagramas de linguagem de programação(UMBRELLO, 2007).

[3] JUDE/Professional é um sistema exclusivo para desenhar ferramenta que suporta UML, Diagrama entidade relacionamento, fluxograma, Mind Map (R) e CRUD(CHANGEVISION, 2007).

[4] *Enterprise Architect* é uma ferramenta que combina a mais recente especificação da UML 2.1 com um excepcional editor gráfico, licenciado de acordo com os termos e condições do EULA(SPARXSYSTEMS, 2007).

1.2 OBJETIVO GERAL

A implementação de uma ferramenta livre que explore as possibilidades tecnológicas da Web 2.0 para permitir a criação e o compartilhamento de especificações tipo UML (ou outras representações gráficas) através de um simples *browser* de *Internet*.

1.3 OBJETIVOS ESPECÍFICOS

Para cumprir com o objetivo principal deste projeto, são estabelecidos como objetivos específicos:

- Buscar os conceitos que estruturam a especificação de Web 2.0 e UML;
- Analisar as ferramentas UML existentes, levantando as especificações e requisitos mais significativos para compreensão das interações dessas ferramentas com os seus usuários;
- Planejar e apresentar as necessidades estruturais de uma plataforma de edição gráfica de elementos da UML (classe, interações e outro) com suporte às especificações de Web 2.0;
- Desenvolver uma ferramenta UML distribuída que possua características de Web 2.0.

1.4 METODOLOGIA

Os seguintes procedimentos são utilizados no desenvolvimento do projeto e implementação da ferramenta:

- Identificação e estudo dos conceitos e tecnologias relacionados ao desenvolvimento da ferramenta por intermédio de pesquisa bibliográfica, referências na *Internet* e analise de ferramentas existentes de edição de gráficos de classes UML.
- Planejamento e descrição da arquitetura da ferramenta. Os resultados desse procedimento são especificações da arquitetura, dos componentes dessa arquitetura e interações entre os componentes da arquitetura.
- Especificação da arquitetura da ferramenta. Os resultados são a entrega de documentos que descrevam as tecnologias envolvidas na estrutura da ferramenta e motivo que fizeram optar por essas tecnologias.
- Desenvolvimento e avaliação da arquitetura estudada e especificada em etapas anteriores.

2 *ESTUDOS REALIZADOS*

2.1 OBJETIVOS DOS ESTUDOS

Durante a elaboração do trabalho de conclusão, necessitou-se realizar pesquisas bibliográficas para compreender os conceitos envolvidos na elaboração da ferramenta proposta. Esses estudos tiveram como objetivos identificar quais conceitos e tecnologias deveriam ser empregadas e como elas poderiam ser melhor utilizados para confeccionar a ferramenta.

2.2 DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE

O processo típico de desenvolvimento de software no mundo do *software* livre é difícil de definir em sua totalidade, pois diversas comunidades adotam práticas particulares. Entretanto, algumas características são comuns a todas. Uma delas é a distância geográfica entre os membros da equipe de desenvolvimento. Esta é uma dificuldade que precisa ser superada e uma forma tradicional de fazê-lo é por meio de ferramentas informáticas da *internet* (LOPES,).

Ferramentas de suporte a comunicação síncronas (mensagens instantâneas, vídeo conferencias) e assíncronas (e-mail, fóruns) e mesmo de gestão do conhecimento vêm sendo usados para superar essas barreiras (AUDY, 2002).Outras classes de ferramentas também são fundamentais e estão bem consolidadas, como sistemas de controle de versões como:

- **SVN** - Subversion: http://subversion.tigris.org/
- **CVS** - *Concurrent Version System*: http://savannah.nongnu.org/projects/cvs/

E sistemas de gerenciamento de projetos como:

- **NetOffice**: http://sourceforge.net/projects/netoffice/
- **DotProject**: http://www.dotproject.net/

- **IBM RPM** - *Rational Portfolio Manager*: http://www-306.ibm.com/software/awdtools/portfolio/

Outras áreas ainda contam com esforços de pesquisas e desenvolvimento, como ferramentas para gerência de requisitos. Um exemplo é o **Jeremia** mantido pela comunidade SourceForge.

2.3 UML - *Unified Modeling Language*

"No período que precedeu o surgimento da UML (década de oitenta e primeira metade da década de noventa), foram propostas dezenas de metodologias voltadas ao desenvolvimento de especificações de metodologias voltadas ao desenvolvimento de especificações de análise de projeto orientados a objetos. Isso criou um ambiente caótico sobre vários aspectos. Um primeiro problema óbvio é que a diferença de notações propostas nas metodologias de análise e projeto orientadas a objetos dificultava alguém habilitado a uma notação compreender um projeto especificado em outra. Um outro problema envolvia as ferramentas de modelagem. Uma vez que os fabricantes tinham de optar por uma notação específica, a gama de escolha dos usuários acabava bastante limitada"(SILVA, 2007)

A UML propõe a unificação de algumas metodologias existentes, podendo ser mais facilmente aceita por usuários de outras metodologias. A primeira versão da UML foi lançada em 1995, tendo como autores Rumbaug e Booch, posteriormente juntaram-se ao grupo Jacobson surgindo à versão 0.9 em 1996. Possuindo o intuito de deixar a UML com padrão internacional, uniu-se em janeiro de 1997 o consórcio OMG (*Object Mangement Group*)e sendo apresentada a verão 1.1 da UML.

A versão 0.9 da UML especificava os seguintes diagramas:

- Diagrama de classes
- Diagrama de objetos
- Modelos de casos de uso
- Diagrama de seqüência
- Diagrama de colaboração
- Diagrama de estado
- Diagrama de componentes

- Diagrama de utilização
- Diagrama de atividade

A última versão da metodologia UML foi liberada em agosto de 2005, sendo constituída por quatro documentos disponíveis no site da OMG: Infra-estrutura de UML, Superestrutura de UML, Linguagem para Restrições em Objetos (OCL) e Intercâmbio de Diagramas.

- **Infra-estrutura de UML:** "O conjunto de diagramas de UML constitui uma linguagem definida a partir de outra linguagem que define os elementos construtivos fundamentais. Essa linguagem que suporta a definição dos diagramas é apresentada no documento *infra-estrutura de UML*. O elemento de suporte à definição dos modelos de UML de UML do nível do usuário é conhecido como *metamodelo de UML*. *Metamodelo* significa o "modelo do modelo, isto é, o modelo usado para definir os modelos"(SILVA, 2007);
- **Superestrutura de UML:** "Documento que complementa o documento de *Infra-estrutura de UML* e que define os elementos das linguagem no nível do usuário, isso é, o conjunto de diagramas"(SILVA, 2007)
- **Linguagem para Restrições em Objetos (OCL):** "Documento que apresenta a linguagem usada para descrever expressões em modelos UML, como pré-condições, pós-condições, invariantes. A adoção de OCL nos modelos leva a especificações de projeto independentes de linguagem-alvo de programação"(SILVA, 2007)
- **Intercâmbio de Diagramas:** "Para a primeira versão de UML foi produzido um padrão de notação baseada em XML, conhecido como XMI (intercâmbio de metadados XML), para permitir a troca de especificações UML entre ferramentas distintas. Uma limitação da proposta original é que não era possível representar informações referentes a aspectos gráficos dos modelos, como a disposição dos elementos sintáticos em um diagrama, por exemplo. A razão é que esse espécie de informação não faz parte do metamodelo de UML - apesar de ser relevante no contexto das ferramentas de edição de diagramas. XMI originalmente apresenta correspondência com a estrutura do modelo de UML. Assim, somente com XMI não seria possível representar algo que não fosse previsto no modelo. Para suprir essa deficiência, a segunda verão de UML, que apresenta uma extensão do metamodelo voltado a informações gráficas. A extensão permite a geração de uma descrição no estilo XMI original (baseado no metamodelo) permiti produzir representações portáveis de especificações UML por ferramentas distintas, sem perda de informação ou ocorrência de incompatibilidades"(SILVA, 2007).

A última versão da disponível da UML é a 2.1, de agosto de 2007. A última versão apresta um conjunto de diagramas, nove deles são provenientes das outras versões, e outros foram adicionados para aumentar a especificidade da UML.

2.3.1 Diagrama de Objetos

"Trata-se de uma variação do diagrama de classes em que, em vez de classes, são representadas instâncias e ligações entre instâncias. Sua finalidade é descrever um conjunto de instâncias existente algum momento da execução do software modelado, como uma espécie de fotografia do tempo de execução"(SILVA, 2007).

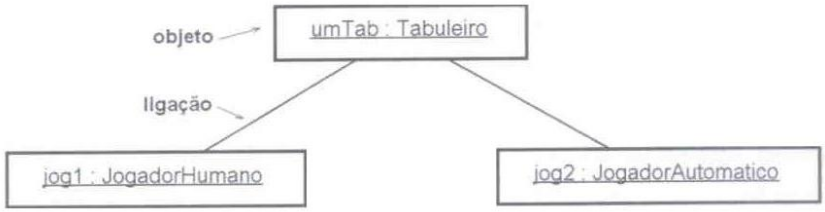

Figura 2.1: Exemplo de diagrama de objetos

2.3.2 Diagrama de Pacotes

"O pacote é o elemento sintático voltado a conter elementos sintáticos de uma especificação orientada a objetos. Esse elemento foi definido na primeira versão de UML para ser usado nos diagramas então existentes, como o diagrama de classes, por exemplo. Na segunda versão da linguagem, foi introduzido um novo diagrama, o diagrama de pacotes, votado a conter exclusivamente pacotes e relacionamentos entre pacotes"(SILVA, 2007).

2.3.3 Diagrama de Estrutura Composta

"O diagrama de estrutura composta é um dos novos diagramas propostos na segunda versão de UML. É voltado a detalhar elementos de modelagem estrutural, como classes, pacotes e componentes, descrevendo sua estrutura interna. Além disso, introduz a noção de porto, um ponto de conexão do elemento modelo, a que podem ser associadas interfaces. Também utiliza a noção de colaboração, que consiste em um conjunto de elementos interligados através de suas portas para a execução de uma funcionalidade especifica"(SILVA, 2007).

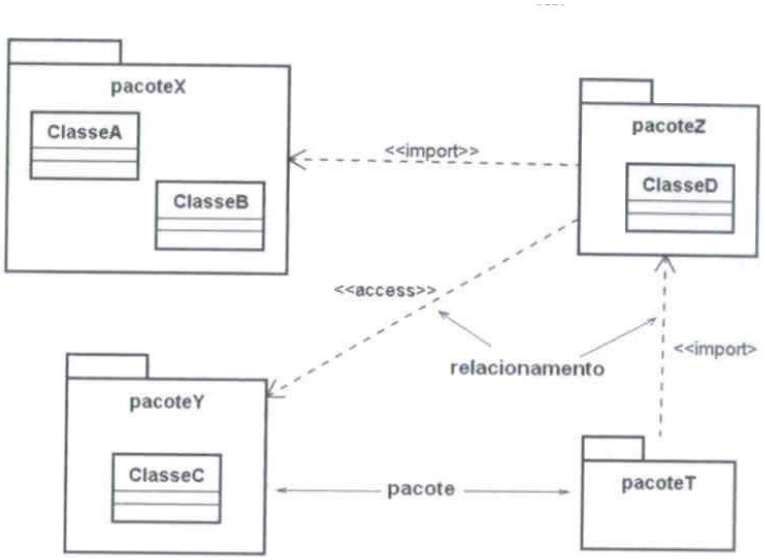

Figura 2.2: Exemplo de diagrama de pacotes

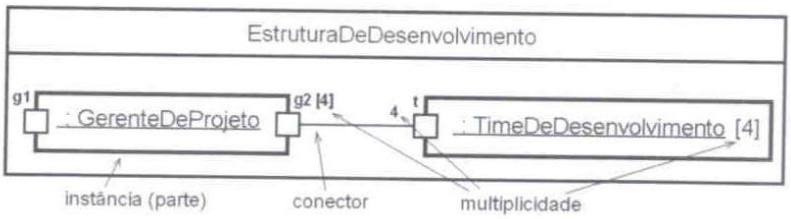

Figura 2.3: Exemplo de diagrama de estruturas composta

2.3.4 Diagrama de Componentes

"O diagrama de componentes é um dos dois diagramas de UML voltados a modelar software baseados em componentes. a finalidade do diagrama é especificar componentes e relacionamentos entre componentes"(SILVA, 2007).

Um componente pode ser representado como um conjunto de classes. O diagrama de componente representando a modelagem estrutural do sistema em um nível de abstração superior ao do diagrama de classes.

2.3.5 Diagrama de Utilização

"É votado a mostrar a organização do conjunto de elementos de um sistema computacional, para sua execução. O principal elemento sintático é o nodo, que representa um recurso computacional (*software* ou *hardware*). Podendo ser representados em um diagrama tanto nodos

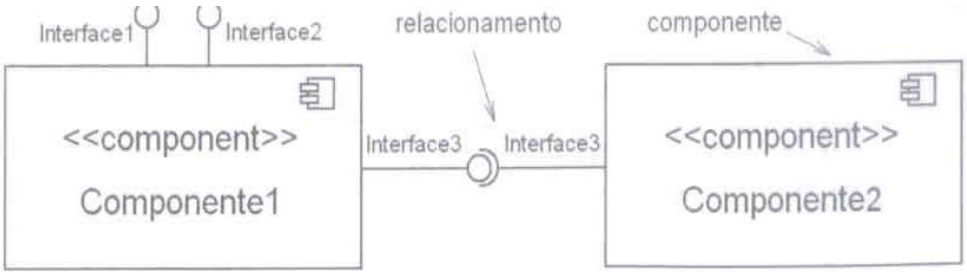

Figura 2.4: Exemplo de diagrama de componentes

(espécies de nodo) como instâncias de nodos (ocorrências de nodos)"(SILVA, 2007).

A imagem inferior representando o relacionamento de três nodos.

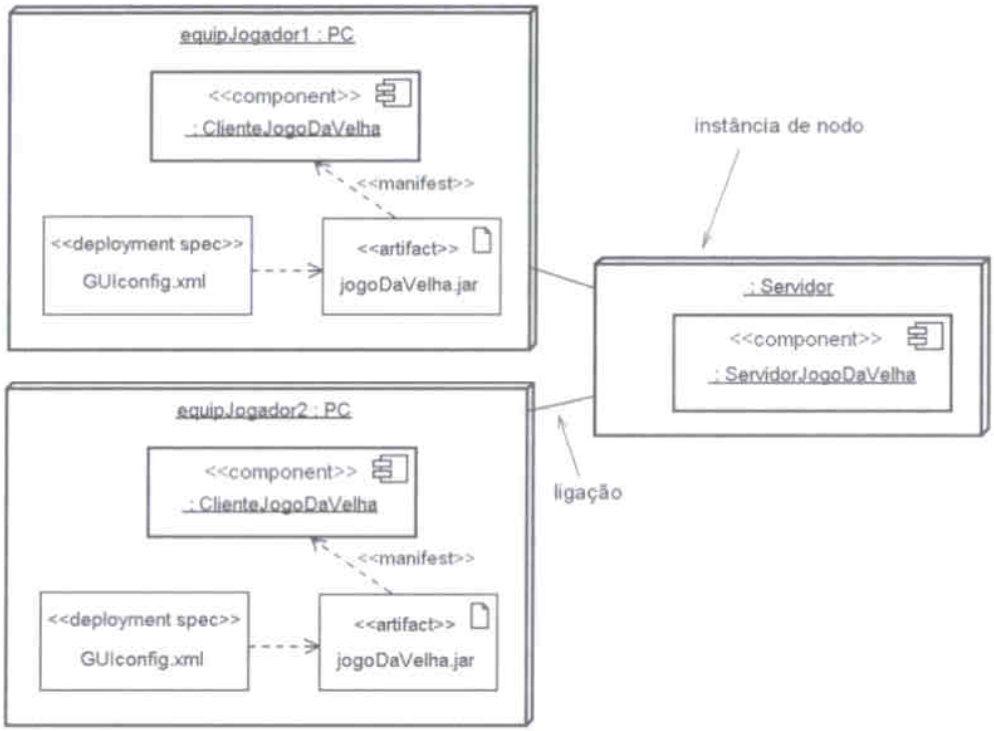

Figura 2.5: Exemplo de diagrama de utilização

2.3.6 Diagrama de Casos de Uso

"O diagrama de caso de uso corresponde ao diagrama que modela a dinâmica do sistema no mais alto nível de abstração proporcionado por UML. Relaciona as funcionalidades do sistema modelado, através do elemento caso de uso, e os elementos externos que interagem como o sistema modelado, através do elemento sintático ator. O diagrama é composto por esses elementos

e relações envolvendo pares desses elementos"(SILVA, 2007).

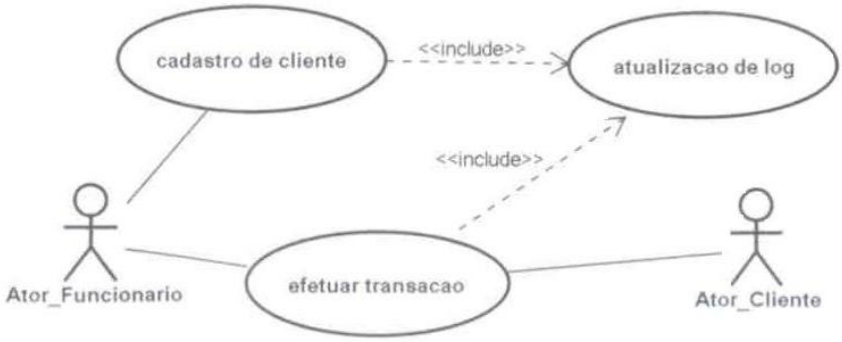

Figura 2.6: Exemplo de diagrama de casos de uso

2.3.7 Diagrama de Seqüência

"O diagrama de seqüência é voltado a descrever objetos interagindo. Seus principais elementos sintáticos são objeto e mensagem (enviada de um objeto para outro). É voltado à modelagem dinâmica do sistema e sua principal finalidade em uma modelagem orientada a objetos é o refinamento de casos de uso"(SILVA, 2007).

Representa as chamadas necessárias para realizar um caso de uso específico, mostrando todos os objetos e os seus métodos para a realização do caso de uso.

2.3.8 Diagrama de Comunicação

"É voltado a descrever objetos interagindo e seus principais elementos sintáticos são objetos e mensagem. É voltado a modelagem dinâmica do sistema e corresponde a um formato alternativo para descrever interação de objetos, disponível em UML. Ao contrário do diagrama de seqüência, o tempo não é modelado explicitamente"(SILVA, 2007).

2.3.9 Diagrama de Máquina de Estados

"O diagrama de máquina de estados tem como elemento principais o estado, que modela uma situação em que o elemento modelado pode estar ao longo de sua existência, e transição, que leva o elemento modelado de um estado para outro"(SILVA, 2007).

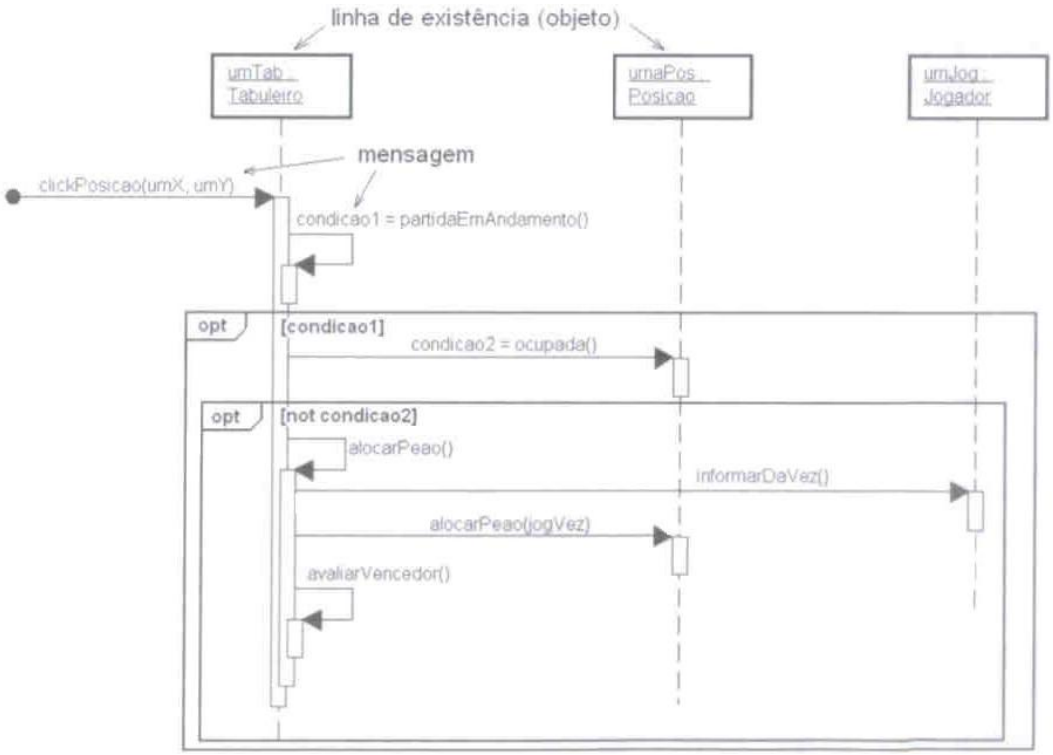

Figura 2.7: Exemplo de diagrama de seqüência

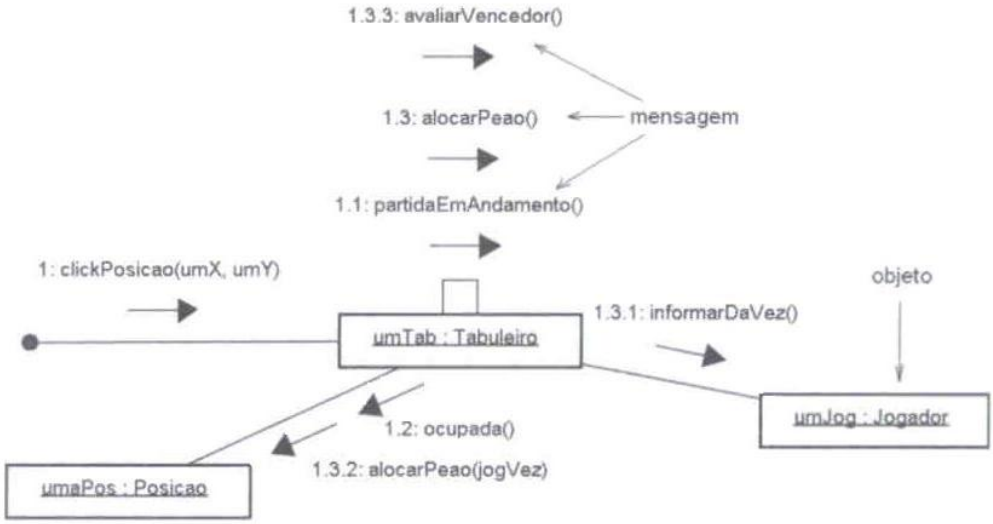

Figura 2.8: Exemplo de diagrama de comunicação

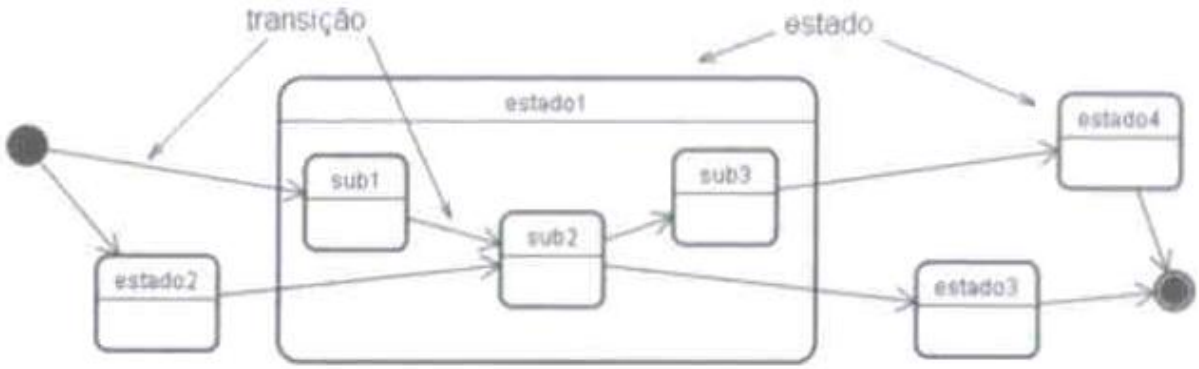

Figura 2.9: Exemplo de diagrama de máquina de estados

2.3.10 Diagrama de Atividades

"Um diagrama de atividades é voltado a descrever uma atividade, que corresponde a uma fração do comportamento do sistema modelado. Uma atividade pode ser descrita como um conjunto de ações e um conjunto de atividades"(SILVA, 2007).

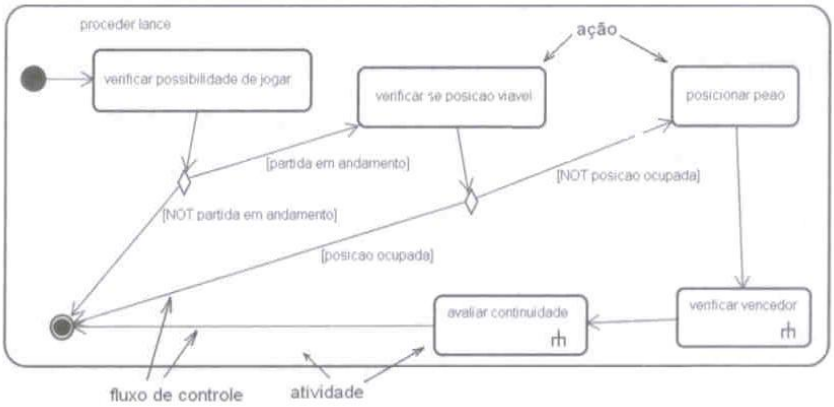

Figura 2.10: Exemplo de diagrama de atividades

2.4 Diagrama de Classes

O diagrama de classes representa a estrutura das classes de um sistema orientado a objetos. Esse diagrama se propôs a demonstrar a estrutura das classes, com seus atributos, métodos e relações existente entre as outras classes do programa orientado a objeto.

"O Diagrama de classes tem a capacidade de modelar os elementos de um programa orientado a objetos em tempo de desenvolvimento, isto é, as classes com atributos e métodos. Tem, porém a capacidade de modelar os relacionamentos entre classes de forma mais explícita que aquela do código"(SILVA, 2007)

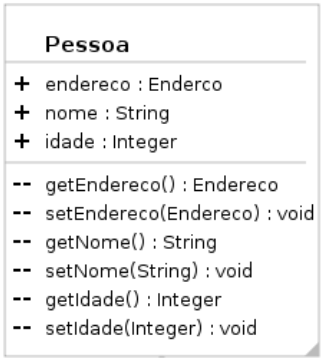

Figura 2.11: Exemplo de classe implementada pela WebUML

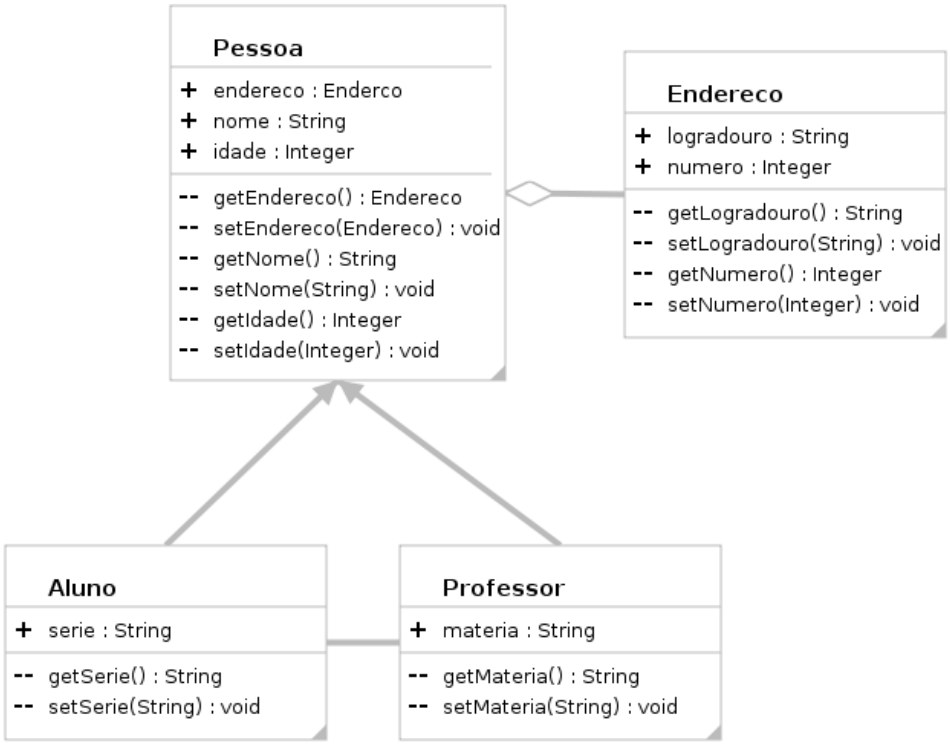

Figura 2.12: Exemplo de diagrama de classe implementado pela WebUML

2.4.1 Classe, Atributos e Métodos

O diagrama de classes possui a classes como um elemento básico, nela que são atribuídas as características dos objetos que vão ser instanciados. Uma classe é representada por um retângulo subdivido em três partes, na posição superior é reservada para o identificador da classe, no centro é destinada aos atributos da classe e na parte inferior aos métodos da classe.

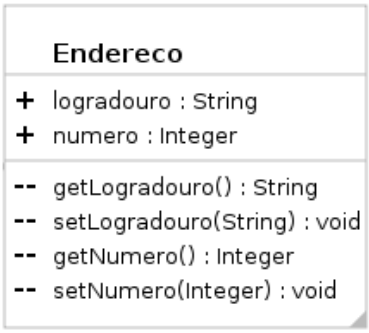

Figura 2.13: Classe Endereco

Atributos

O atributo é um elemento da classe que pode representar uma característica dos objetos instanciados ou valor de controle da classe. "A representação básica de um atributo consiste em seu identificador, ou seja, o nome do atributo, e seu tipo, sendo que explicitação do tipo é opcional"(SILVA, 2007). Podendo ser representado por um único nome, como:

aluno aluno: Pessoa

A UML especifica uma sintaxe para escrever o elemento atributo em uma classe (SILVA, 2007).

```
[<visibilidade>]['/']<nome>[':'<tipo>]['['<multiplicidade>']']['='<valor_inicial>]['{'<propriedade>[','<pro
```

Onde:

- <visibilidade> pode ser público (+), privado (-), protegido (#) e pacote ( );
- `'/'` significa que se trata de um atributo derivado - seu valor é computado a partir de outras informações;

- <nome>é o identificador do atributo (jamais iniciado com letra maiúscula);
- ':' <tipo> é o identificador do tipo do atributo;
- <multiplicidade> ::= <faixa>[''<ordem> [','<unicidade>]'']

  `<faixa> :: = [<valor_inferior> '..']<valor_superior>`

  <ordem> pode ser ordenado ou não ordenado

  <unicidade> pode ser único ou não único;
- <propriedade> pode ser usado para indicar outras caratcterísticas não expressas em outros campos, como atributo não permitir alterações de valor (*readOnly*), que é uma alternativa de um conjunto de valores preestabelecidos para a associar restrições ao atributo.

Métodos

O método é um elemento que representa uma chamada de procedimento de um objeto para outro objeto. "A sintaxe da representação de método inclui seu identificador, parâmetros com respectivo tipo, tipo de retorno (se houver) e, opcionalmente, visibilidade, e propriedade"(SILVA, 2007).

A especificação para escrever um método pela UML (SILVA, 2007).

```
[<visibilidade>]<nome>'('[<lista_parametros>]')'[':'[<tipo_retorno>]'{'<propriedade>[';'<propriedade>]*]'}'
```

Onde:

- <visibilidade>, <nome> e <propriedade> são equivalentes ou que foi apresentado para atributo;
- `<lista_prametros>` é a relação de parâmetros do método, com respectivos tipos;
- `<tipo_retorno>` é o tipo de retorno do método, se houver retorno (void pode ser utilizado para ressaltar nenhum retorno).

2.4.2 Relacionamentos entre Classes

Um diagrama de classes não trata simplesmente de classes soltas com seus elementos, mas também das interações existente entre as classes.

2.4.3 Herança

"A herança estabelece uma relação de especialização entre duas classes, em que uma delas corresponde a um conceito mais genérico e a outra, a um conceito mais específico"(SILVA, 2007).As classes que se destinam a receber os elementos (atributos e métodos) genéricos são criadas da observação de outras classes que possuam elementos equivalentes. Como nas classes Aluno e Professor, onde elas podem possuir atributos e métodos equivalentes. Com exemplo:

- **Atributos**: nome, idade e endereco
- **Métodos**: getNome(), setNome(), setIdade() e getIdade().

Podendo ser criada uma supreclasse (classe genérica) Pessoa que poderá agregar alguns elementos equivalentes das classes Aluno e Professor. As classes Aluno e Professor vão ser subclasses da superclasse Pessoa. "Quando duas classes estão interligadas por herança, os atributos e métodos da superclasse são herdados pelas subclasses"(SILVA, 2007).

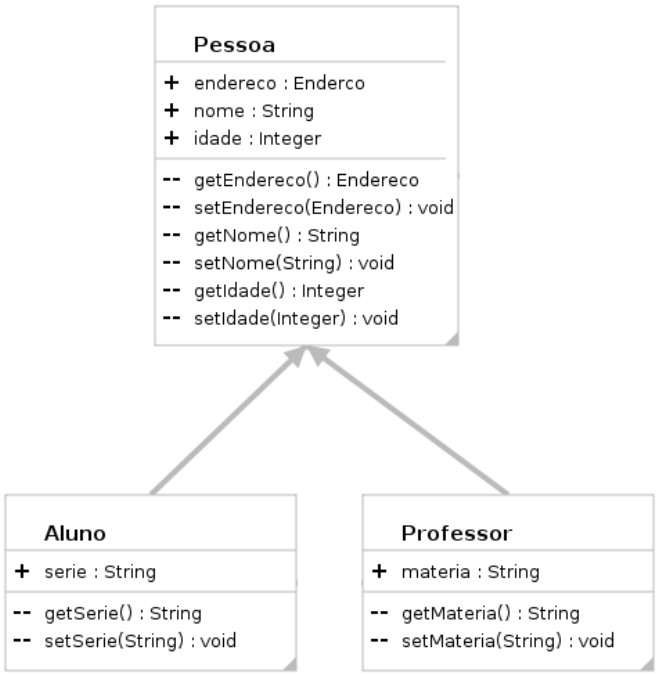

Figura 2.14: Exemplo de herença implementada pela WebUML

2.4.4 Agregação

"A agregação é um relacionamento entre duas classes e que estabelece que uma instância de uma classe agrupa uma outra mais instância de outra classe"(SILVA, 2007). A classe que vai ser agregada representa um elemento de outra classe (atributo), como na classe Pessoa que possui um atributo endereco, onde o elemento endereco pode ser representado por uma nova classe. A criação da nova classe Endereço, poderia vir da necessidade de agregar mais de um elemento em um já existente (endereco). A nova classe Endereco poderia ter como atributos: logradouro, número, cidade e estado. Ficando elegante a criação de uma nova classe, do que a colocação daqueles elementos em uma lista.

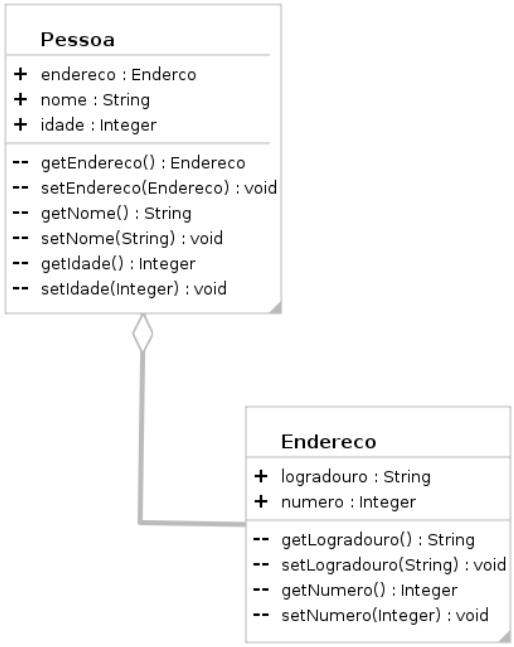

Figura 2.15: Exemplo de agregação implementada pela WebUML

Composição

"Um tipo de relação de agregação, também chamada de *agregação forte*, em que há um conjunto de requisitos na ligação entre parte e agregação "(SILVA, 2007). Uma composição é uma agregação, onde o objeto instanciado da classe agregada possui um vinculo com o tempo de existência do objeto principal. Um exemplo seria entre uma classe Pessoa que agrega a classe Mao, um objeto Mao necessita de um objeto Pessoa instanciado para ser criado, mas um objeto Pessoa não necessita que seu objeto agregado Mao seja instanciado para ser criado.

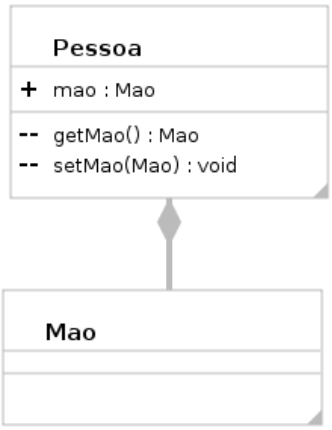

Figura 2.16: Exemplo de composição implementada pela WebUML

2.4.5 Associação

"Quando há o reconhecimento de um relacionamento entre classes, que não ser caracterizado nem como herança, nem como agregação ou composição, a associação surge como a alternativa"(SILVA, 2007). Uma associação é um tipo de relação mais simples entre duas classes, normalmente não possuindo uma ligação direta entre elas. Um exemplo poderia ser uma associação entre as classes Pneu e PedalDeFreio. Nenhuma das duas classes necessita possuir a outra classe como elemento interno para executar o procedimento de frenagem. Esse procedimento pode ser executado por uma objeto Carro que recebe uma mensagem do objeto PedalDeFreio para o objeto Pneu, ou então, o objeto Carro passa uma referência do objeto Pneu para o objeto PedalDeFreio, podendo assim o objeto PedalDeFreio enviar uma mensagem direta para o objeto Pneu. Fica claro que existe um vinculo entre as classes Pneu e PedalDeFreio, mas não temos a necessidade de um classe agregar a outra para realizar o procedimento de frenagem. O relacionamento entre essas classes e de associação

Figura 2.17: Exemplo de associação implementada pela WebUML

2.5 XML - eXtensible Markup Language

2.5.1 Apresentação do XML

Extensible Markup Language (XML) é simples, um formato de texto muito flexível derivou-se do SGML (ISO 8879). "Projetou originalmente receber com desafio a propagar eletrônica em grande escala, XML está recebendo também um papel cada vez mais importante na troca de uma variedade grande de dados na WEB e em outras partes"(W3C, 2007).

2.5.2 Elementos do XML

XML é uma linguagem de marcação extensível, possuindo a sua especificação derivado do SGML. 'Em termos construtivos, os documentos XML são documentos que atendem ao padrão SGML'(W3C, 2007). Sendo utilizado para descrever diversas formas de dados (XMI - XML Metadata Interchange, XHTML - The Extensible HyperText Markup Language, etc), propossionando o envio de grande massas de dados pela internet.

"O XML dá aos seus desenvolvedores a capacidade de criar seus próprios conjuntos de elementos, seus próprios atributos e até mesmo suas própias entidades"(PITTS, 2000).

Os Objetivos para o projeto do XML, pela o W3C são (JR., 2002):

- O XML deve ser utilizado de forma direta e objetiva na internet;
- O XML deve suportar uma ampla gama de aplicativos;
- O XML deve ser compatível com SGNL;
- O XML deve ser fácíl o bastante para escrever programas que processem documento XML;
- O número de recursos adicionais no XML deve ser mantido em um nível mínimo, idealmente zero;
- Os documentos XML precisam ser legíveis e relativamente claros;
- O projeto de XML deve ser preparado rapidamente;
- O design do XML deve ser formal e conciso;
- Os documentos XML devem ser fáceis de serem criados;

- A concisão na marcação do XML é de pouca importância.

O que torna o XML simples são as tecnologias adotas pela a linguagem. O XML é constituído de um agrupamento de vários padrões, possuindo dois elementos principais para declarar a estrutura de um XML, que são:

- DTD - *Document Type Definition*
- XML Schema

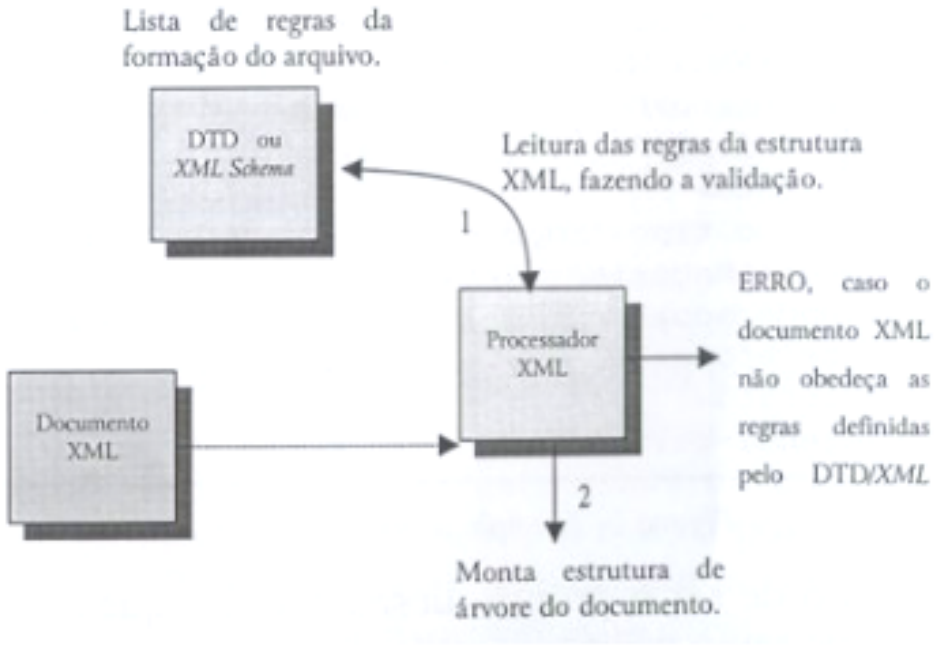

Figura 2.18: Processo de análise de um documento XML

DTD - *Document Type Definition*

Document Type Definition, ou simplesmente DTD, contém as regras que definem quais as tags que podem ser usadas em um documento XML e quais os valores válido.

"DTD é, basicamente, um conjunto de regras que define as informações que serão enviadas ao analisador sintático para o documento que será analisado. Um DTD pode incluir um conjunto de declarações de elementos, atributos, entidades, notações e comentários que deseja utilizar. As diversas declarações de componentes determinam como o documento será estruturado e depois enviam essas informações, para o analisador sintático O analisador sintático, por sua vez, envia os resultados ao aplicativo de visualização"(JR., 2002).

```
<?xml version="1.0"?>
<!DOCTYPE lista [
        <!ELEMENT pessoa   (nome,idade,endereco)>
        <!ELEMENT nome     (#PCDATA)>
        <!ELEMENT idade    (#PCDATA)>
        <!ELEMENT endereco (#PCDATA)>
]>

<lista>
        <pessoa>
                <nome>Carlos Henrique Pereira</nome>
                <idade>25</idade>
                <endereco>Florianópolis</endereco>
        </pessoa>

        <pessoa>
                <nome>Eduardo Salomão Pereira</nome>
                <idade>20</idade>
                <endereco>Florianópolis</endereco>
        </pessoa>
</lista>
```

Código 2.5.1: *Exemplo de um XML estuturado pelo DTD*

A especificação de um XML elabora por um DTD possui alguns problemas.

"Primeiro, porque as DTDs são escritas numa sintaxe que possui pouca relação com XML e que não podem ser analisadas por um *parser* XML. Em segundo lugar, todas as declarações em uma DTD são globais, o que significa que você não pode definir dois elemento diferentes como o mesmo nome, mesmo se aparecerem em contextos separados. Finalmente, e talvez o mais importante, é que não se pode especificar para cada elemento os tipos de dados, tais como inteiro, data, real e assim por diante"(JR., 2002).

XML Schema

"O XML *Schema* possui, basicamente, a mesma funções de uma DTD, que é a definir as partes de um documento e descrever como elas podem ou não ser usadas, o que pode ser colação em seus interiores e se são ou não elementos obrigatório do documento"(JR., 2002).

Os problemas relacionados com o DTD, como o analisado por um *parser* XML, todas as declarações são globais e a especificação dos tipos dos elementos. São supridos pela especificação do XML *Schema*.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="quali

<xs:element name="lista">
        <xs:complexType>
                <xs:element name="pessoa">
                        <xs:sequence>
                        <xs:element name="nome" type="xs:string"/>
                        <xs:element name="idade" type="xs:decimal"/>
                        <xs:element name="endereco" type="xs:string"/>
                        </xs:sequence>
                </xs:element>
        </xs:complexType>
</xs:element>

</xs:schema>
```

Código 2.5.2: *Exemplo de arquivo XSD da estrutura XML Schema*

```
<?xml version="1.0"?>

<lista xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation=

        <pessoa>
                <nome>Carlos Henrique Pereira</nome>
                <idade>25</idade>
                <endereco>Florianópolis</endereco>
        </pessoa>

        <pessoa>
                <nome>Eduardo Salomão Pereira</nome>
                <idade>20</idade>
                <endereco>Florianópolis</endereco>
        </pessoa>

</lista>

```

Código 2.5.3: *Exemplo de um XML estuturado pelo XML Schema*

2.6 Web 2.0

Web 2.0 é uma expressão criada pela empresa O'Reilly Media e apresentada em 2004 para especificar as novas relações que surgiram na Web. O aparecimento das comunidades e dos serviços que utilizam à plataforma Web, como os Wikis e o Orkut [1] onde os usuários possuem uma grande interação entre eles. Web 2.0 não significa uma nova versão da Web, mas é sim, uma nova forma de interagir e pensar nas relações com os usuários da Web (WIKIPEDIA, 2007).

"Em uma conferência entre as empresas O'Reilly Media e Media Live International ficou

[1] Orkut é uma rede social mandida pela empresa Google

claro para os participantes que a web estava cada vez mais importante, com novas aplicações e sites aparecendo regularmente. Os sites (e as empresas) que sobreviveram ao estouro da bolha se destacavam e tinham algo em comum. Para eles o estouro da bolha foi o nascimento do conceito 'Web 2.0'"(CERDEIRA, 2006).

2.7 JavaScript

"O Javascript é uma linguagem em *script* orientado a objeto desenvolvida pela Netscape, sendo utilizada por milhões de paginas Web e por aplicações de usuários em todo o mundo. O Javascript de Netscape é um super-extensão do ECMA-262 (ECMAScript)"(CONTENTS, 2007).

O JavaScript não é uma extensão da linguagem Java, mas uma sintaxe baseada em Java e C++ [2], tentando reduzir o número de conceitos novos requerido para a utilização da linguagem JavaScript. Possuindo elementos semelhantes dessas linguagens, como: *if*, *for* e *while*, *try* e *catch*, *switch* e *function*.

JavaScript aceita a evocação de funções nos modelos procedural e linguagem orientada a objetos. Os objetos instanciados na linguagem JavaScrip,t podem ter seus métodos e procedimento modificados em tempo de execução.

[2] C++ é uma linguagem de programação de alto nível, multiparadigma e de uso geral

```
<html lang="pt-br">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Apresentação do JavaScript</title>

        <script language="JavaScript">

        var color = "bbb";

        /*
        * É super classe do exemplo, possuidora do atributo "string" e
        * seus métodos de acesso (get e set).
        */
        function superClasse() {

                var string = "superClasse";

                this.getString = function(){
                        return string;
                }

                this.setString = function( pString ){
                        string = pString;
                }
        }

        /*
        * É sub classe do exemplo
        */
        function subClasse(){

                // procedimento para declarar herança em Javascript.
                this.inheritFrom = superClasse;
                this.inheritFrom();

                // Método responsavel por mostrar como trabalhar com o
                // atributo "string" da super classe.
                this.trabalhaString = function(){
                        var string2 = this.getString();
                        string2 = string2+" - modificado";
                        return string2;
                }
        }

```

Código 2.7.1: *Exemplo de código em JavaScript*

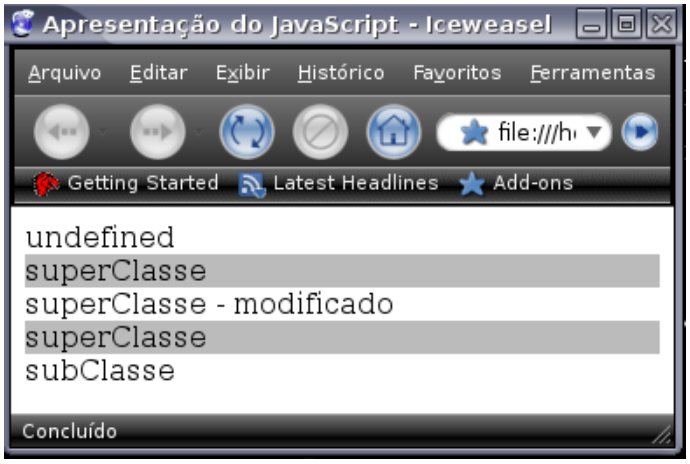

Figura 2.19: Saída do código de exemplo em JavaScript

2.8 SVG - *Scalable Vectorial Graphics*

"SVG é uma linguagem para descrever gráficos bidimensionais e aplicações gráficas utilizando XML. SVG 1.1 é uma recomendação de W3C e sendo o formato principal dos desenvolvimentos atuais de SVG. SVG *Tiny* 1.2 é a especificação atualmente mais nova desenvolvida"(W3C, 2007). SVG é uma plataforma para gráficos bidimensionais. Possuindo duas formas de ser utilizada: através de um arquivo de XML simples ou por um API de programação para aplicações gráficas. As características chaves incluem figuras, textos e gráficos, possuindo diversas formas de elabora uma imagem em SVG. Suporta *script* especificado pela linguagem ECMAScripte [3] e inclui suporte para animações.

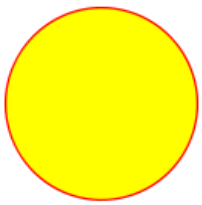

Figura 2.20: Exemplo de imagem em svg

[3]ECMAScript é uma linguagem de programação baseada em scripts, sendo base para outras linguagem como JavaScript/JScript

```
<?xml version="1.0" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
    "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg"
         xml:space="preserve"
         width="130" height="130"
         viewBox="0 0 100 100"  >

    <circle cx="50%" cy="50%" r="48"
        fill="yellow" stroke-width="1" stroke="red"/>
</svg>
```

Código 2.8.1: *Código XML de uma imagem SVG*

```
<?xml version="1.0" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
    "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg"
         xml:space="preserve"
         width="130" height="130"
         viewBox="0 0 100 100"  >

    <script type="text/ecmascript"><![CDATA[

    function color(elemento) {
        var color = elemento.getAttribute('fill');
        if(color != 'red'){
                if(color == 'yellow'){
                        elemento.setAttribute('fill','red');
                        elemento.setAttribute('stroke','green');
                }else{
                        elemento.setAttribute('fill','yellow');
                        elemento.setAttribute('stroke','red');
                }
        }else{
                elemento.setAttribute('fill','green');
                elemento.setAttribute('stroke','yello');
        }

    }
    ]]></script>


    <circle cx="50%" cy="50%" r="48" fill="yellow" stroke-width="1" stroke="red"
        onclick="color(this)"
        />

</svg>
```

Código 2.8.2: *Código XML com inserção de código JavaScript em uma imagem SVG*

Figura 2.21: Exemplo de saída do código XML apresentado, levando em consideração um clique realizado por período de tempo

3 FERRAMENTA PROPOSTA

3.1 ESPECIFICAÇÃO DA FERRAMENTA

Desenvolver uma ferramenta livre que explore as possibilidades tecnológicas da Web 2.0 para permitir a criação e o compartilhamento de diagramas de classes, através de um simples *browser* de *internet*. Este aplicativo deverá oferecer uma interface em que os usuários criarão diagramas de classe e, ao final da construção dos mesmos, permitirá gerar um XMI que represente o diagrama elaborado. Este sistema deve ser adaptável a sistemas de apoio ao desenvolvimento de colaborativo de projetos (como SourceForge [1], Gforge [2], etc).

3.1.1 Necessidades paraIimplementar a Ferramenta

Para desenvolver a ferramenta foram levantadas algumas necessidades, para que a implementação pude-se ocorre no período previsto. As necessidades levantadas foram essa:

- Identificação e aprofundamento das especificações básicas de funcionamento e interações do sistema com os usuários;
- Planejamento de uma plataforma em camada que suporte as especificações do sistema;
- Levantamento das necessidades de cada camada;
- Estudos de quais tecnologias são mais adequadas para cada uma das camadas

3.1.2 Atividades previstas

As atividades previstas para realização do trabalho de conclusão de curso. As atividades do trabalho são:

[1] SourceForge é um *site* de desenvolvimento colaborativo para aplicações de código aberto.
[2] Gforge é um sistema que possibilita a criação de comunidades de código aberto

- Levantar as necessidades de funcionamento com os contratantes. Sendo realizado por intermédio analise de ferramentas, como Jude e Umbrello, e leitura sobre os temas envolvidos para elaboração do trabalho. O resultado dessa atividade de ser a apresentação de um diagrama das interações.

- Criação de uma estrutura em camada para o sistema a ser implementado. O resultado é um diagrama com as camadas e explicando cada uma delas e suas interações.

- Especificação das camadas. O resultado da atividade é a entrega de documentos que descrevam as tecnologias envolvidas em cada camada e porque foram escolhidas.

- Desenvolvimento das camadas. Essa atividade representa um ciclo de análise, desenvolvimento, teste e avaliação das camadas que estão sendo implementadas. O resultado final é a entrega do código completo do sistema previsto nos serviço.

3.1.3 Levantamento das necessidades

Para compreender as interações finais e auxiliando na implementação das funcionalidades da ferramenta proposta, confeccionou-se um diagrama que representa as seqüências de interações de um usuário ao utilizar o sistema. Esse diagrama é uma síntese de todas as seqüências de atividade do editor UML.

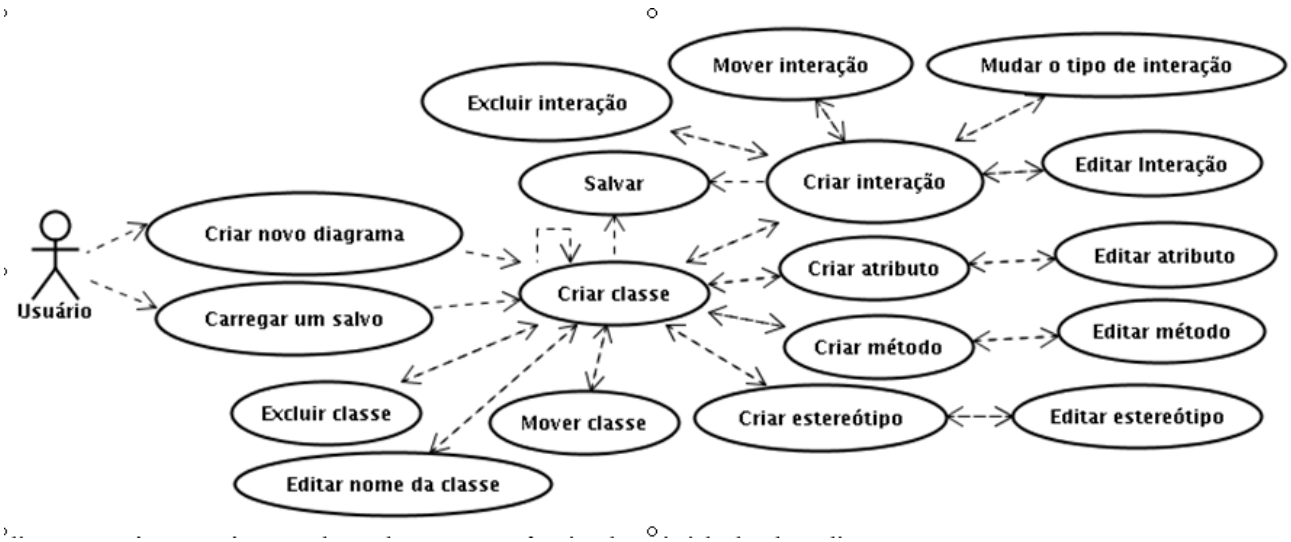

Figura 3.1: Diagrama de atividades do WebUML

3.1.4 Estrutura das camadas

Para desenvolver o sistema, foi executado um estudo para identificar qual seria a melhor arquitetura. O objetivo era especificar uma arquitetura simples e modular. O resultado é um estrutura em camadas para facilitar a compreensão e desenvolvimento da aplicação.

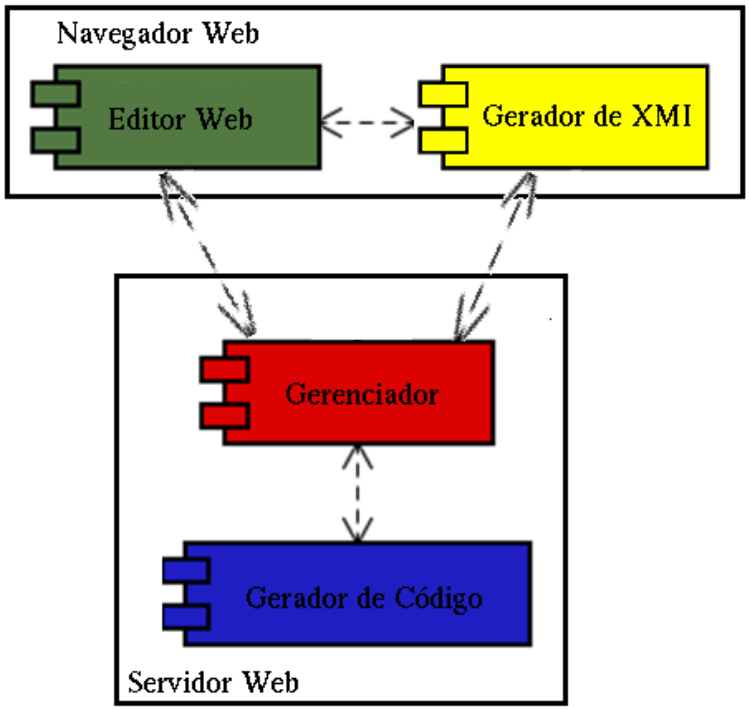

Figura 3.2: Estrutura das camadas do WebUML

Editor Web

Essa camada é responsável pela interação direta com os usuários da aplicação, ficando sobre cargo dela toda a interface da edição de UML. Ela necessita ter a capacidade, através do navegador Web, possibilitar que os usuários editem os digramas UML sem gerar nenhuma chamada ao servido da aplicação. Todo o processamento deverá ser realizado no cliente.

Gerador de XMI

Camada responsável pela geração de documentos XMI a partir das especificações geradas pelo Editor Web. A opção de colocar o Gerador XMI na navegado web do cliente é para aumentar a portabilidade do sistema entre outras linguagem, facilitando a adaptação do WebUML em sistemas desenvolvidos em linguagem diferente.

Gerenciador

É a camada que cuida da comunicação de todas as outras. Responsável por gerenciar todo o sistema, ficando a cargo dela as ações de gerenciar as requisições geradas pelos usuários e cuidar da criação e apresentação do código fonte. Essa camada repassa a especificação XM gerada pelo Gerador de XMI, onde o Gerador de Código receberá a especificação e gerará o código fonte.

Gerador de Código

É a última camada do sistema, responsável por receber um arquivo XMI e a parti dele devolver um código fonte. Essa camada deverá gerar código para varias linguagem diferente.

3.2 Tecnologias Envolvidas em Cada Camada

3.2.1 Tecnologoias das Camadas

Editor Web

Na camada de Edição Web - foi escolher utilizar a especificação SVG, e relação às outra tecnologia como Adobe flash [3], a escolha dela se deu por ela possuir essa características:

- Baseado em XML;
- Padrão aberto;
- Código aberto;
- Gráficos vetoriais e matriciais;
- Pode ser acessada pelo browser;
- Diferentes possibilidades de visualizar uma área no mapa;
- Zoom e Pan interativos sobre o mapa;
- Interação e animação;
- Visualização seletiva de características geográficas (camadas);

[3] Adobe flash é um software para geração de gráficos vetoriais, possibilitando a criação de animações na Web. Mantido pela empresa Adobe Systems.

- Pode ser englobada dentro de arquivos HTML, ASP, PHP, etc.;
- Interage com várias linguagens de script, como: JavaScript, VBScript, etc.

Gerador de XMI

No Gerador de XMI foi gerado a necessidade de ser executado num navegador Web, para suprir esse requerimento, optou-se em desenvolver o gerador XMI na linguagem JavaScript. No trabalho com XML para geração de XMI, escolheu-se em não utilizar nenhuma biblioteca de XML, o desenvolvimento dessa funcionalidade será feita sobente com a linguagem JavaScript.

Gerenciador

Nessa camada optou-se por utilizar a linguagem PHP [4], e relação às outras linguagens de programação como o Java [5]. Essa escolha se deu por ser uma linguagem de estrutura, mas simples (não necessitando de frameworks para desenvolvimento web e instalação de interpretadores para compilação do código), facilitando que outras pessoas possam modificar o código fonte dessa camada.

Gerador de Código

Na camada de gerador de código escolheu-se utilizar um programa pronto em vez de desenvolver um novo módulo, devendo possuir as seguintes funcionalidades::

- Importar e exportar XMI;
- Gerar código de algumas linguagens de programação, necessariamente C++ e Java;
- Possuir uma documentação que auxilie na manipulação da ferramenta.

[4] PHP é uma linguagem de programação interpretada
[5] Java é uma linguagem de programação orientada a objeto

4 IMPLEMENTAÇÃO DA FERRAMENTA

4.1 Etapas do desenvolvimento

Para desenvolver Editor UML, optou-se por dividir as etapas de em têS partes bem caracterizadas: Implementação do editor de diagramas de classes UML, elaboração da estrutura do servidor e implementação da estrutura do servidor.

4.2 Implementação do Editor de Diagramas de Classes UML

4.2.1 Solidificação dos Conceitos de Envolvidos

Nessa etapa foi realizado estudo para capacitar sobre as tecnologias envolvidas para o desenvolvimento da aplicação. Foi realizado estudos com manuais da W3C, com o intuito de melhora o conhecimento na área de na manipulação de objeto SVG por JavaScript, CSS [1] e AJAX [2].

4.2.2 Elaboração dos objetos SVG-JavaScript

Teve por objetivo especificar, através das informações levantamento em uma etapa anterior de analise de requisitos, a estrutura do Editor Web. Descrevendo os objetos SVG e sua interações e a funções JavaScripts que serão necessárias para controlar aqueles objetos. O diagrama de classe elaborado expressa os objetos do domínio do problema, com seus atributos, métodos e interações.

[1] CSS - Cascading Style Sheets, tecnologia utilizada para formatar documentos HTML, XML e XHTML

[2] AJAX - *Asynchronous Javascript And* XML, Tecnologia utilizada para gernciar as solicitações assíncronas de informações entre o *Browser* e o servidor de Web.

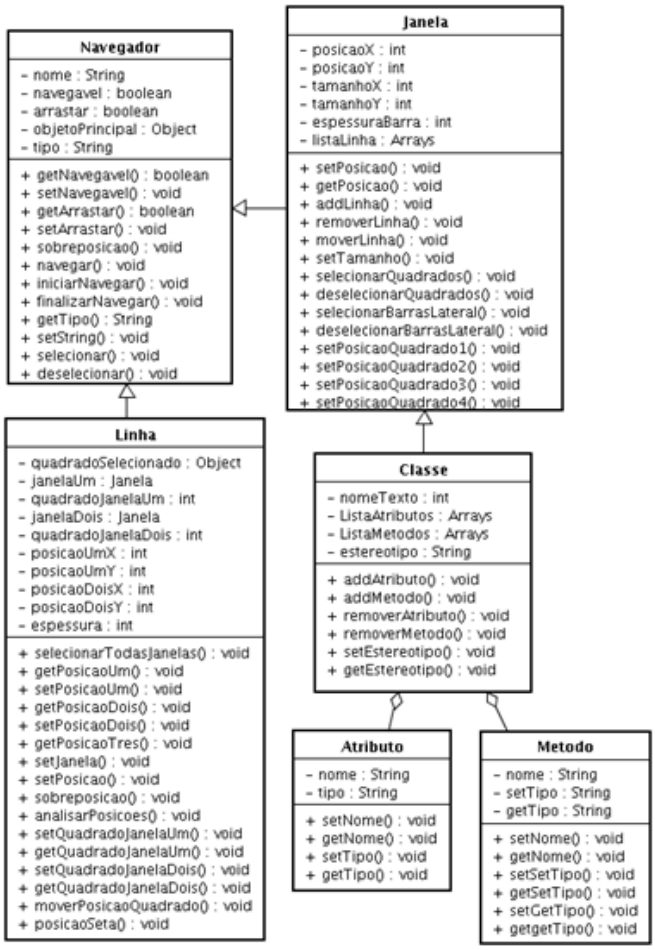

Figura 4.1: Diagrama de classes de alguns objetos JavaScript

4.2.3 Implementação dos Objetos SVG-JavaScript

Nessa etapa foi desenvolvido o Editor Web seguindo a especificação elaborada na etapa anterior. Foi construído uma biblioteca de JavaScript para gerenciar a criação e o ciclo de vida dos objetos SVG. Essa biblioteca possui os seguintes arquivos JavaScript:

- **navegador.js** – responsável pelo gerenciamento da concorrência na navegabilidades dos objetos SVG.
- **Janela.js** – cuida da navegabilidade e da interação dos usuários com os objetos que estende a classe Janela.
- **linha.js** – gerencia todas as interações dos usuários com a linhas do Editor UML.
- **classe.js** – responsável por cuidar de todas as característica da classe e da interação da classe com os usuários.

Além da biblioteca JavaScript também foi implementado alguns documentos svg, para serem responsáveis por receber os objetos criado e cuidar da biblioteca JavaScript.

4.3 Elaboração da Estrutura do Dervidor

Para elaborar a estrutura do servidor levou-se em consideração desenvolver uma estrutura simples para gerenciara a troca de informação entre as camadas, com uma estrutura de classe bem funcional. O objetivo é desenvolver uma aplicação com um grau de complexidade não muito elevado, para facilitar a reutilização e a manutenção da ferramenta por outras pessoas.

No principio a primeira opção para a implementação da ferramenta era a utilização do Java JEE com o *framework* JSF (*Java Server Face*). Mas tarde, optou-se por utilizar a linguagem PHP, pois foram observador alguns aspectos do projeto como:

- Não possuir necessidade de muitos elementos HTML, seria utilizado mais elementos SVG;
- A linguagem utiliza deveria possuir boas bibliotecas para comunicação utilizando AJAX e para tratamento de String;
- Tecnologias envolvidas deveria ser as mesma utilizada em projetos *open source*.

Na implementação escolheu-se dividir as classe envolvida em três bibliotecas, cada uma com suas características bem definidas em relação.

Uma biblioteca para cuidar das requisições AJAX geradas pelos Ediot Web e de algumas paginas HTML para melhorar a usabilidade do Editor Web.

Outra biblioteca para gerenciar todos os processos da aplicação. Sendo responsável por controlar todas as ações da aplicação.

Mais uma biblioteca para controlar a comunicação entre a aplicação que está sendo desenvolvida com o software de geração de código fonte.

4.4 Implementação da Estrutura do Servidor

Para desenvolver a camada do **Gerador de Código** foi observada a necessidade de realizar uma buscar por informações sobre ferramentas que possam gerar código a partir de XMI.

4.4.1 Atividades Previstas

As atividades previstas para o desenvolvimento desta camada são:

- Realizar um estudo sobre os softwares que tenha capacidade de gerar código fonte;
- Planejar e estruturar da camada de Geração de Código;
- Implementação da estrutura da camada de Geração de Código;

4.4.2 Estudo Sobre as Ferramentas de Geração de Código

Foram realizados estudos para identificar qual ferramenta poderia atender os requisitos para geração de código proposto no serviço. A ferramenta deveria possuir alguns requisitos básicos:

- Possuir uma estrutura que possibilite pegar um documento xmi e gerar código em pelo menos em três linguagens de programação. Sendo duas dessas linguagens obrigatoriamente Java e C++;
- Possuir uma documentação que possa auxiliar na integração com WebUML;
- Possuir uma licença que seja similar a GPL [3] e BSD [4];
- A ferramenta deverá ter funções que possibilite que outra aplicação faça chamadas remotas para suas funcionalidades.

As ferramentas analisadas em um primeiro momento foram Umbrello e o ArgoUML, e os resultados alcançados estão representados na tabela 4.1.

Essas duas aplicações se mostram com boa capacidade de serem integradas na ferramenta proposta, entretanto o ArgoUML mostrou uma dificuldade no trabalhar com arquivos XMI. Diferentemente, o Umbrello se mostrou bem mais atrativo por já existir algumas ferramentas que possuam a capacidade de importe e exporte XMI para ela.

[3] GNU General Public License (Licença Pública Geral) é uma licença para *software* livre idealizada por Richard Stallman, mantida pelo projecto GNU

[4] Licença BSD é uma licença de código aberto utilizada inicialmente em sistemas operacionais *Berkeley Software Distribution*

Ferramentas > Requisitos v	**ArgoUML**	**Umbrello**
Gerar código	Java, PHP, C++ e C#	ActionScript, Ada, C++, CORBA IDL, Java, JavaScript, PHP, Perl, Python, SQL e Esquema XML
Documentação	Excelente, documentação . bem detalhada e completa	Excelente, documentação bem detalhada e completa
Licença	BSD	GPL
XMI	Foram realizados testes de importação e exportação de XMI e o ArgoUML apresentou diversos problemas ao tentar realizar estas operações.	Os projetos são salvos no formato XMI que facilita a compatibilidade.
Modular	Não foi encontrado nenhum software que desse suporte para o ArgoUML	Foram encontrados alguns software que pode ser integrados através dos arquivos XMI. Exemplo: como o projeto "*Php to Xmi converter*" do repositório sourceforge.

Tabela 4.1: *Tabela de comparação entre o ArgoUML e o Umbrello*

4.4.3 Planejar e Estruturar a Camada de Geração de Código

O planejamento da integração da WebUML com o Umbrella, foi idealizado para ser através de caixa preta. Uma aplicação não conhece como a outra executa as suas funcionalidades, simplesmente executam chamadas para suas funcionalidades.

Deverá ser realizados estudos para conhecer a estrutura de XMI que a ferramenta Umbrella utiliza em sua especificação. Esses estudos podem ser executáveis através de outras ferramentas que importavam ou exportavam para a Umbrella.

Foi planejada uma arquitetura para a camada de Geração de Código, que possuirá três módulos:

- **Gerador XMI**, responsável por cuidar da transposição dos objetos JavaScript gerado pela camada **Editor Web** em arquivos XMI.
- **Gerenciador Umbrella**, responsável por cuidar de todas as chamadas geradas entre as duas aplicações. Esse módulo repassava o arquivo XMI para o Umbrello.
- **Gerenciador de Código** que ficou responsável por empacotar o código fonte gerado pelo Umbrella e repassar para o modulo **Editor Web**.

5 *TESTES NA FERRAMENTA*

5.1 Verificação e Teste do Desenvolvimento da Ferramenta

Foram realizados testes no âmbito do Projeto Via Digital, com o intuito de validar e testar o desenvolvimento ocorrido na primeira etapa do processo de desenvolvimento. Disponibilizou-se uma versão de teste numa URL para que o projeto Via Digital tivesse uma forma de testar os requisitos implementados, podendo assim realizar um *feedback* da situação do desenvolvimento. Os pontos levandados:

- Lentidão na navegação da classes e janelas;
- Interface sem muitos *feedback*;
- Baixo nível de interações:
- Dificuldade de modificação dos elementos (Classes e Relacionamentos) pré-selecionados;
- Dificuldade de perceber as características dos elementos criados.

5.2 Implementação dos Melhoramentos Levantados

Uma das melhorias levantadas, foi agilizar a navegabilidade dos elementos do editor UML, realizaram-se estudos que buscasse minimizar em código a ocorrência de laços, responsáveis por grandes carga de processamento e ou consumo de memória. Foi observado por esse estudos que o gerenciamentos dos objetos JavaScript estava todo centrado em uma única lista, a qual erá utilizada em vários momentos para realização de consultas no objetos javaScripts. Para resolução dessa questão foi dividido o gerenciamento dos objetos JavaScript em varias lista que agrupava os elementos pelos seus tipos(janelas, linha, classe, texto, ...).

Outra observação para melhorar a agilidades dos elementos gráficos foi restrutura a gerencia das ações dos objetos JavaScript. que anteriormente cabia ao próprio elemento gerenciar as suas

ações, passando para uma única classe o gerenciamento das ações dos objetos JavaScript.

Mais um ponto observado foi a baixa ausência de ações que gerassem resposta ao usuário segundo suas interações com a ferramenta. Esse ponto foi resolvido criando duas classe: MsnTexto e MsnJanela. MsnTexto é uma classe responsável por mostra uma mensagem na tela da ferramenta, com o intuito de alerta, negar ou informar ao usuários sobre as ações tomadas por ele em determinado momento da sua utilização. MsnJanela é uma classe que gerencia as mensagem transmitida para o usuário através de janelas, possuindo o objetivo de informar ou pedir confirmação das ações tomada pelo usuário durante a utilização da ferramenta..

Observou-se também a necessidade de melhora a forma de interação ente os usuários e a ferramenta. Anteriormente havia sido adotada uma postura muito simples e direta para as interações realizadas na ferramenta. Essa abordagem não se mostrou muito eficiente, deixando o usuários em alguns momento desorientado em relação as funcionalidades. A modificação do menu da classe ocorreu por motivo de maior comodidade para os usuários nas ações repetitivas. No formato do menu anterior, o usuário necessitavam clicar varias vez para conseguir fazer algumas ações como criar atributo ou métodos. O menu só aparecia quando o usuário clica-se sobre a classe e desaparecia depois de selecionar qualquer ação do menu. Para melhorar esse ponto optou-se por colocar o menu no lado da classe, o usuário agora necessita clicar uma vez só para fazer ter todas as opções de ações do menu por um período de tempo bem maior.

Foi criado uma região para auxiliar na edição dos elementos gráficos. Através das interações com os usuários, observou que eles não conseguiam identificar e editar todos os atributos dos elementos gráficos da ferramenta. Para solucionar esse empecilho criou-se no lado esquerdo um campo para edição das classes e das interações entre as classes, esse novo campo facilita as ações do usuário, dando maior agilidade e confiança nas ações tomadas.

6 CONCLUSÕES

Para realizar o trabalho de conclusão de curso foram levantados e traçados alguns objetivos durante o planejamento do trabalho.

O primeiro objetivo, que era buscar os conceitos que estruturam a especificação de Web 2.0 e UML, chegou a um resultado satisfatório, e está detalhado no capítulo 2 (Estudos Realizados). Nesse capítulo também são trabalhados os assuntos a seguir, que complementaram os estudos planejados: Desenvolvimento distribuído de*software*, W3C – World Wide Web Consortium, XML – eXtensible Markup Language, XMI - XML Metadata Interchange, SVG - Scalable Vectorial Graphics e JavaScript.

Também foram analisadas algumas ferramentas de manipulação gráfica de especificações UML: ArgoUML e Umbrello. Foram levantadas as especificações e as funcionalidades mais significativas para a compreensão das possíveis interações dessas aplicações com a ferramenta proposta. A análise realizada na tabela 4.1 do trabalho registra o resumo destes estudos.

Depois do estudo bibliográfico, planejou-se uma plataforma para dar suporte às especificações de interatividade assíncrona pela Web (comumente chamado de Ajax ou Web 2.0) para um editor gráfico que permitisse a edição de elementos da UML (classe, interações e outros). Foram levantadas as necessidades estruturais dessa plataforma e criou-se uma estrutura em camadas para a ferramenta que foi descrita no capítulo 3.

Em especial, a maior parte do trabalho foi focado na especificação e elaboração de uma ferramenta de apoio ao desenvolvimento de *software* que permitisse a edição de Diagramas de Classe (segundo as especificações UML 2) em um navegador Web. Esta implementação foi considerada uma prova de conceito para avaliar a viabilidade das hipóteses e propostas iniciais. Para elaborar essa ferramenta optou-se por dividir a implementação em quatro camadas:

- Editor Web – É camada responsável pela interface gráfica com o usuário final, onde é possível editar diagramas de classes da UML.
- Gerador XMI – Camada responsável pela geração de documentos XMI a partir das espe-

cificações criadas pelo Editor.

- Gerenciador – É uma camada de comunicação entre as outras camadas.
- Gerador de código – É a última camada da estrutura, responsável pela comunicação com a ferramenta Umbrello que, em última análise é a responsável pela geração de código.

Essas partes tiveram sua estrutura especificada durante a elaboração desse trabalho. Foram realizados estudos para saber quais tecnologias e ferramentas seriam as mais adequadas para implementá-las com eficiência em cada camada da ferramenta.

A partir da plataforma especificada foi desenvolvida a camada do Editor Web. As demais camadas (Gerador XMI, Gerenciador e Gerador de código) encontram-se em um estado inicial de desenvolvimento, mas seu desenvolvimento deverá ser realizado durante o final desse ano.

Durante o trabalho observaram-se alguns pontos: dificuldades para realização da ferramenta, viabilidade do uso das tecnologias envolvidas e potencial futuro da ferramenta.

- As dificuldades encontradas para realizar esse trabalho se deram por dois motivos: escolher quais tecnologias deveriam ser utilizadas e encontrar bibliografia sobre as tecnologias escolhidas para implementação do trabalho. Houve bastante dificuldade para encontrar artigos que definissem como utilizar SVG juntamente com JavaScript, e também sobre as aplicações que deveriam ser utilizadas na camada de gerador de código.
- Foi observada a viabilidade do uso das tecnologias envolvidas, em especial a tecnologia central do projeto: a SVG. Por ser uma tecnologia recentemente difundida notou-se um grave problema nas especificações utilizadas pelos *browsers*. Alguns seguem a especificação da W3C e outros nem implementam um renderizador para SVG.
- Outro problema observado é o potencial futuro da ferramenta, por utilizar SVG em um *browser*, ela é concorrente direta do Adobe Flash, uma linguagem bem difundida pela comunidade de desenvolvedores web.

A implementação das outras funcionalidades do Gerenciador e todo o desenvolvimento do Gerador de código poderão ser realizadas em trabalhos futuros, ou com o auxílio da comunidade SourceForge, onde o projeto vai ser mantido depois da entrega desse trabalho de conclusão de curso.

Referências Bibliográficas

TIGRIS, O. S. S. E. T. *ArgoUML*. Open Source Software Engineering Tools, 2007. Disponível em: <http://argouml.tigris.org/>.

UMBRELLO, C. da. *Umbrello UML Modeller*. SourceForge, 2007. Disponível em: <http://uml.sourceforge.net/index.php>.

CHANGEVISION. *Jude System Design Tool*. ChangeVision, 2007. Disponível em: <http://jude.change-vision.com/jude-web/index.html>.

SPARXSYSTEMS. *Enterprise Architect*. Sparx Systems, 2007. Disponível em: <http://www.sparxsystems.com.au/ea.htm>.

LOPES, A. M. e. J. L. N. A. L. T. *Uma proposta para processo de requisitos em ambientes de desenvolvimento distribuído de software*.

AUDY, R. P. J. L. N. *Um Modelo de Referência para Desenvolvimento Distribuído de Software*. Pontifícia Universidade Católica do Rio Grande do Sul, 2002. Disponível em: <http://www.inf.pucrs.br/ jaudy/WTES2003_prikladnicki.pdf>.

SILVA, R. P. e. *UML2 em Modelagem Orientada a Objetos*. Florianópolis: Visual Books, 2007.

W3C, T. W. W. W. C. *Extensible Markup Language (XML)*. The World Wide Web Consortium, 2007. Disponível em: <http://www.w3.org/XML/>.

PITTS, N. M. *XML Black Book - Solução e Poder*. São Paulo: Makron Books, 2000.

JR., J. R. T. *XML, Schema*. Florianópolis: Visual Books, 2002.

WIKIPEDIA, C. da. *Web 2.0*. Wikipedia, 2007. Disponível em: <http://pt.wikipedia.org/wiki/Web_2.0>.

CERDEIRA, F. M. *WEB 2.0*. O Globo Online, 2006. Disponível em: <http://oglobo.globo.com/blogs/tecnologia/post.asp?cod_Post=11919>.

CONTENTS, M. D. C. *About JavaScript*. Mozilla, 2007. Disponível em: <http://developer.mozilla.org/en/docs/About_JavaScript>.

W3C, T. W. W. W. C. *Scalable Vector Graphics (SVG) - XML Graphics for the Web*. The World Wide Web Consortium, 2007. Disponível em: <http://www.w3.org/Graphics/SVG/>.