

arquivar

Relatório Final
Projeto Orientado de
Computação II

**Gerador de Mapas de Site Para
Sistemas Web**

Orientadores: Guilherme Salles
Mariza Bigonha

Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Aluno: Diego Bigliani Solamito
Data: 11 de dezembro de 2006

Resumo:

Gerador de Mapas de Site Para Sistemas Web
Belo Horizonte, 2006. Projeto Orientado de Computação II, Departamento de
Ciência da Computação, Universidade Federal de Minas Gerais

A visualização de dados é uma área de especial interesse em diversas áreas do conhecimento, pois a representação gráfica é geralmente uma forma efetiva de descrever, explorar e sumarizar um grande volume de informações. Um mapa de site para um sistema web é definido como uma representação gráfica do fluxo do sistema, servindo como ferramenta de apoio, não só para usuários do sistema, mas como para a equipe que o desenvolve. Mais do que apenas uma ferramenta informativa, é uma ferramenta de auxílio na validação e melhoria de fluxos de um sistema web. Este projeto é a concretização de um gerador automático de mapas de sites de sistemas no contexto web.

1.	Introdução	5
1.1.	Contexto Atual.....	5
1.2.	ATAN Sistemas.....	5
1.3.	Framework ^[1]	5
1.4.	Framework-Web da ATAN.....	7
	Desenvolvimento	7
2.	Domínio do Problema.....	7
2.1.	Mapas de Sites	7
2.2.	Definição Formal de um Mapa de Site em um Sistema Web.....	8
2.3.	Componente para Geração Automática de Mapas de Sites.....	8
	O Problema	8
	Visão Informal da Solução	9
3.	Gerador de Mapas de Sites	9
3.1.	Parser	9
	Relação entre Web X Linguagens de Programação	11
	Escopo do Projeto.....	11
3.2.	Parte Gráfica.....	12
4.	Trabalhos Relacionados.....	12
4.1.	GraphViz ^[5]	12
	Arquivo DOT.....	12
5.	Implementação.....	13
5.1.	Parser	13
	Varredura do Projeto	13
	Varredura do Código-Fonte.....	14
5.2.	Desenho do Mapa	18
	Criação do arquivo .DOT	18
	Hierarquia da Classe.....	18
	Horizontalização da Imagem	20
	Instanciação do Mapa De Site	20
	Web-Service para GraphViz ^[5]	21
6.	Problemas Enfrentados	21
6.1.	Problema 1 – Submits Excepcionais	21
6.2.	Problema 2 – Horizontalização excessiva:	22
7.	Utilização.....	22
8.	Resultados.....	23
9.	Apêndices	25
9.1.	SVG ^[9]	25
9.2.	Web-Services.....	26
10.	Referências	28

Figura 1 – Frameworks X Bibliotecas	6
Figura 2 – Viabilidade da criação de um Framework	6
Figura 3 – Exemplo de Página de um Sistema Web desenvolvido pela Atan Sistemas.....	10
Figura 4 – Esquema de Arquitetura dos Sistemas Web na Atan Sistemas.....	10
Figura 5 – Código-Fonte de uma Página HTML desenvolvida em JAVA	11
Figura 6 – Arquivo .dot	13
Figura 7 – Figura gerada pelo software	13
Figura 8 – 1º. Passo do Parser	14
Figura 9 – Exemplos de submits.....	17
Figura 10 – Resultado do Parser (em memória)	18
Figura 11 – (a) Grafo Hierarquizado (b) Grafo Não-Hierarquizado	19
Figura 12 – Exemplo de Projeto com relacionamentos de hierarquias ambíguas	19
Figura 13 – Exemplo de Mapa de Site com horizontalização excessiva.....	20
Figura 14 – O mesmo exemplo da Figura 9 com a horizontalização excessiva eliminada ..	20
Figura 15 – Exemplo de Submit-Excepcional no código-fonte, e o seu resultado no mapa de site.....	22
Figura 16 – Mapa de Site com Horizontalização Excessiva sem grafos desconexos.....	22
Figura 17 – Mapa de Site do Sistema Despacho de Bobinas Terrestres – CST	23
Figura 18 – Fragmento do Mapa de Site de um sub-fluxo do Sistema de Despacho de Bobinas	24
Figura 19 – Mapa de Site de um Sistema desenvolvido para Pocket-PC - Magnesita.....	24
Figura 20 – Mapa de Site do Sistema SGQOM – basicamente com páginas de relatório ...	24
Figura 21 – Mapa de Site do Sistema de Embarque – Fluxos complexos e extensos	25
Figura 22 – Exemplo de Figuras desenvolvidas na linguagem SVG	26

1. Introdução

Nesta seção apresentamos o contexto geral onde o projeto foi desenvolvido.

1.1.Contexto Atual

Desde o meio da década passada, quando a Internet se popularizou aqui no Brasil, assistimos uma corrida sem interrupções de novas tecnologias surgindo no mundo Web. Desde os bate-papos – a pré-história do sucesso na Internet – até os sites de relacionamento de hoje. Desde simples páginas HTML informativas a complexos sistemas em plataforma Web.

A área de TI, no mercado, se viu forçada a se desdobrar por sobre esse novo paradigma. Sistemas que antes ficavam limitados à execução em um terminal isolado, ou, quando muito, a uma rede local, agora se estendiam a uma rede da amplitude da Internet.

Atualmente, é raro ver um profissional de Tecnologia de Informação alienado ao mundo Web. Participar do mercado é estar inserido no meio de muitas siglas e expressões como: HTML, JavaScript, Framework-Web, CSS, Web-Services^[1], etc.

Paralelamente ao rápido crescimento dessas novas tendências, desenvolvedores de código estavam correndo atrás de melhores práticas de programação, novas abordagens como o estouro do paradigma Orientado Objeto, a criação de Padrões de Projeto (Design Patterns^[4]) etc. A agilidade e rapidez requerida no desenvolvimento dos sistemas sempre foram – e por certo continuarão sendo – exigências naturais por parte dos clientes. Aliás, essa exigência é cada vez maior. Estamos falando aqui de como as formas de se construir um software mudaram. Isto é, não só a forma de apresentar o produto ao cliente mudou (de Desktop para Web, grosso modo) como a forma de fazê-lo também.

1.2.ATAN Sistemas

Determinando assim, que o desenvolvimento de Sistemas Web é uma realidade em empresas voltadas para soluções em Tecnologia de Informação, falamos aqui, rapidamente, sobre a empresa Atan Sistemas e o seu processo de desenvolvimento de softwares.

A Atan Sistemas é uma empresa que na área de TI segue a tendência do mercado. Desenvolve sistemas para empresas como Companhia Siderúrgica de Tubarão, Pepsi, Coca-Cola, Gerdau, entre outras. Em sua grande maioria, os projetos que deslancham nessa empresa são desenvolvidos em plataforma Web.

Para consolidação desses sistemas em tempo hábil, a empresa utiliza-se de uma ferramenta própria para atender a exigência na agilidade de soluções por parte dos clientes. Trata-se de um Framework-Web.

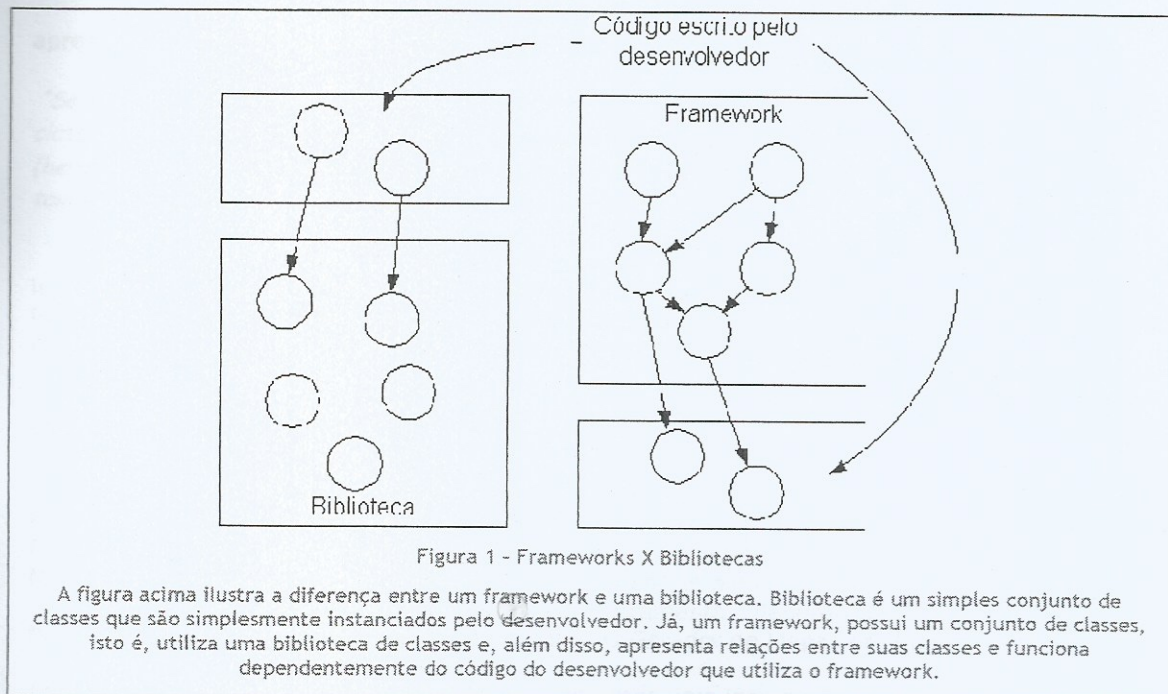
Na próxima seção, continuamos falando sobre a Atan, especificamente sobre o Framework^[1] que ela utiliza, mas antes, damos uma pequena introdução no conceito de Framework^[1].

1.3.Framework^[1]

Obs:Caso o conceito de Framework já esteja consolidado pelo leitor, pule para a seção 1.4.

Um Framework^[1] é um fragmento de software semipronto reusável que pode ser utilizado por vários projetos. Um framework^[1] normalmente é composto por um conjunto de classes abstratas que se relacionam entre si, determinando algum objetivo dentro de um software. Por exemplo: um Framework-Web é composto de um conjunto de classes abstratas que determinam componentes HTML's como: páginas, botões, tabelas, formulários, que interagem entre si para serem utilizados no desenvolvimento de um Sistema-Web. Os frameworks, em sua maioria, são orientado-objetos.

Costuma-se confundir frameworks com bibliotecas. Na verdade, as duas possuem objetivos iguais: reusabilidade de código. Porém, utilizam estruturas diferentes. Uma biblioteca é formada por classes independentes e o usuário deve saber quando instanciar cada objeto. Já, num framework^[1], as classes têm dependências entre si e o usuário embute particularidade nos componentes do framework^[1] para que ele se torne específico ao software onde está sendo utilizado. Embutir particularidade nos componentes significa modificar suas características através de técnicas que a Orientação-Objeto disponibiliza como: herança, sobrecarga e sobrescrita de métodos.



Para que um framework^[1] exista, as aplicações que irão utilizá-lo devem algo razoavelmente grande em comum, ou seja, pertencerem a um mesmo domínio de problema. Afinal, esse é o principal objetivo de um framework^[1], capturar as funcionalidades comuns a essas aplicações.

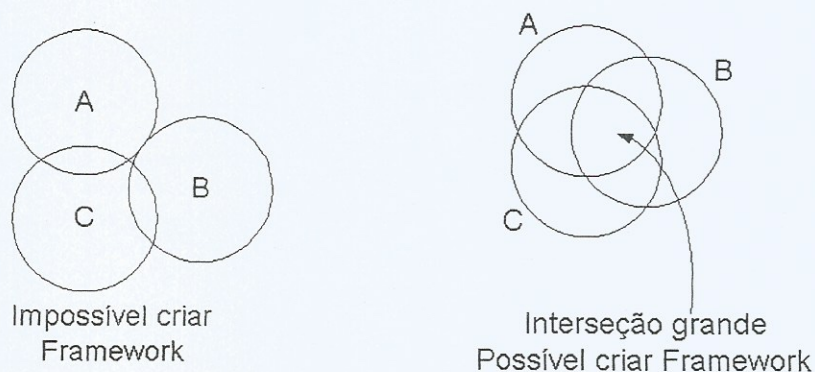


Figura 2 - Viabilidade da criação de um Framework

Um Framework-Web, portanto é a implementação de uma estrutura que cobre funcionalidades ou componentes comuns de Sistemas voltados para a Web. Exemplos de componentes comuns em Sistemas-Web: botões, tabelas, inputs, div's, submit's etc.

1.4. Framework-Web da ATAN

Esta sub-seção dá uma breve introdução ao Framework-Web da Atan. Tal Framework^[1] será citado em várias outras partes deste documento, por isso a importância desta introdução.

A seguir, está uma definição do Framework^[1] da Atan, encontrada no documento de apresentação da ferramenta, para treinamento interno:

"Se trata de uma arquitetura (geradora) projetada para reuso máximo, representada por uma coleção de classes concretas e abstratas, com um comportamento potencial para especializações em subclasses (herança). (...) O framework^[1] determina a arquitetura principal para uma aplicação ser construída e é também responsável pelo controle do fluxo dessa aplicação".

O Framework-Web da ATAN é construído por demanda, sendo modificado e incrementado sempre que possível. Um exemplo – apenas para contextualizar – dessa demanda foi à incorporação da Web 2.0 dentro do funcionamento do Framework^[1]. É utilizado pela empresa como uma ferramenta para desenvolver de modo mais eficaz e mais rápido.

Desenvolvimento

A inclusão de novos componentes ao Framework^[1] ocorre de forma natural e gradual. Podemos citar como componentes do framework^[1]: Table (tabela HTML), Button (botão de form), Img (uma imagem embutida no HTML), Link (link HTML) Div's, entre outros. Note que estamos falando de um framework^[1] voltado totalmente para o contexto web.

Frente a várias necessidades nessa ferramenta, deparou-se com um componente fundamental que ainda não está implementado: O Gerador de Mapas de Sites.

Antes de falar do Gerador de Mapas de Site, daremos uma introdução ao Mapa de Site em si.

2. Domínio do Problema

O objetivo da seção Error! Reference source not found. é dar uma definição clara de um Mapa de Site, delimitando o domínio do problema a ser resolvido.

2.1. Mapas de Sites

Um mapa de site nada mais é do que uma simples página HTML. Essa página informa todos os "caminhos" (links) do sistema que se está representando. Se encararmos o Mapa de Site como uma funcionalidade de um site qualquer (como os que vemos por aí pela Internet), podemos pensar em duas características principais do Mapa: O usuário-alvo da funcionalidade (ator) e o objetivo da funcionalidade (função).

Normalmente, em sites comerciais e pessoais, os mapas de sites têm como usuário-alvo o visitante da página (exterior a empresa que hospeda o Site) e as funções do mapa são:

- Serve de guia de navegação sobre o site.
- Serve como página informativa (o que a página oferece).
- Informa o assunto do site.

Dentro de um Sistema Web – um conjunto de páginas feitas sob medida para o cliente – e não exatamente para Internet – temos um cenário diferente para o propósito de um Mapa de

Site. Interessantemente, o Mapa de Site não terá como alvo apenas o usuário final (o cliente), mas também o pessoal envolvido no processo de desenvolvimento do site.

É importante fazer um breve parêntese aqui sobre a congruência de conceitos que se introduziu desde o início do documento: falamos basicamente da importância e influência de sites (e finalmente, dos mapas de sites) na Internet e falamos do desenvolvimento de softwares – que atualmente tangem o mundo Web.

Então, agora, aqui, temos um cenário de desenvolvimento de softwares sob plataforma Web, onde, especificamente estamos tratando de um Mapa de Site.

2.2. Definição Formal de um Mapa de Site em um Sistema Web

Um Mapa de Site é um componente de Software que ilustra o fluxo de algum Sistema. O seguinte quadro mostra de forma resumida as relações entre atores e funções do mapa de site que estamos tratando.

Função	A t o r	Desenvolvedor	Líder da Equipe de Desenvolvimento	Tester da Equipe de Desenvolvimento	Usuário do Sistema
Auxílio na validação de fluxos		•	•	•	
Guia para entendimento do sistema				•	•
Auxílio na busca de falhas no código-fonte do sistema		•			
Auxílio na detecção de falhas no fluxo do sistema		•	•	•	•
Auxílio na detecção de melhorias no sistema		•		•	•

Tabela 1 - Cross-Over Atore X Funções do Mapa de Site

2.3. Componente para Geração Automática de Mapas de Sites

Uma vez definido um Mapa de Site em um contexto de Sistemas Web, podemos definir o nosso problema.

É desejável, para uma empresa (como a ATAN Sistemas) que a utilização de componentes comuns (no nível de programação) seja o mais reutilizável quanto possível. Um Mapa de Site é um potencial funcionalidade comum a vários projetos, digna de possuir um componente do framework^[1] que o gere, assim como um simples botão, ou uma tabela HTML.

O Problema

O problema a ser atacado é a criação de um Gerador de Mapas de Sites. É preciso que o Gerador fique disponível para os usuários do Framework^[1] (os desenvolvedores), para que possam incluir tal funcionalidade para os usuários do sistema desenvolvido.

Visão Informal da Solução

A primeira coisa que pensamos quando falamos sobre mapas de sites é "*link*". Um sistema web possui um conjunto de links – isto é – transferência de fluxo de uma página para a outra. Logo, o primeiro a se preocupar é como recuperar os *links* de um sistema feito sobre o Framework-Web em questão.

Supondo que tenhamos, então, recuperado esse conjunto de links - o imaginamos dentro de alguma estrutura (um grafo, por exemplo) o próximo passo do projeto é a parte gráfica, isto é, como a informação guardada até agora será exibida.

Podemos, portanto, dividir o nosso problema em duas frentes principais:

- Foco de detecção dos links de um projeto
- Foco da parte Gráfica

3. Gerador de Mapas de Sites

3.1. Parser

Nesta primeira parte do projeto, estamos interessados em recuperar todos os links do sistema e guardá-los de alguma forma. Como fazer isso?

Antes de responder a tal pergunta (que ficará pendente até o fim da seção 3.1), é necessário dar uma breve introdução ao processo de desenvolvimento de sistemas na empresa Atan Sistemas.

Os sistemas desenvolvidos pela ATAN são sistemas para soluções operacionais: (sistemas que realizam ações de operações: como movimentação de produtos, viagem de produto, estoques etc), gerenciais (geração de relatórios, de listas etc) e estratégicas (telas de gerenciamento de negócio, por exemplo). Para ilustrar, somente, mostramos a seguir um print screen de uma tela inicial de um sistema web desenvolvido pela Atan para a Companhia Siderúrgica de Tubarão (CST).

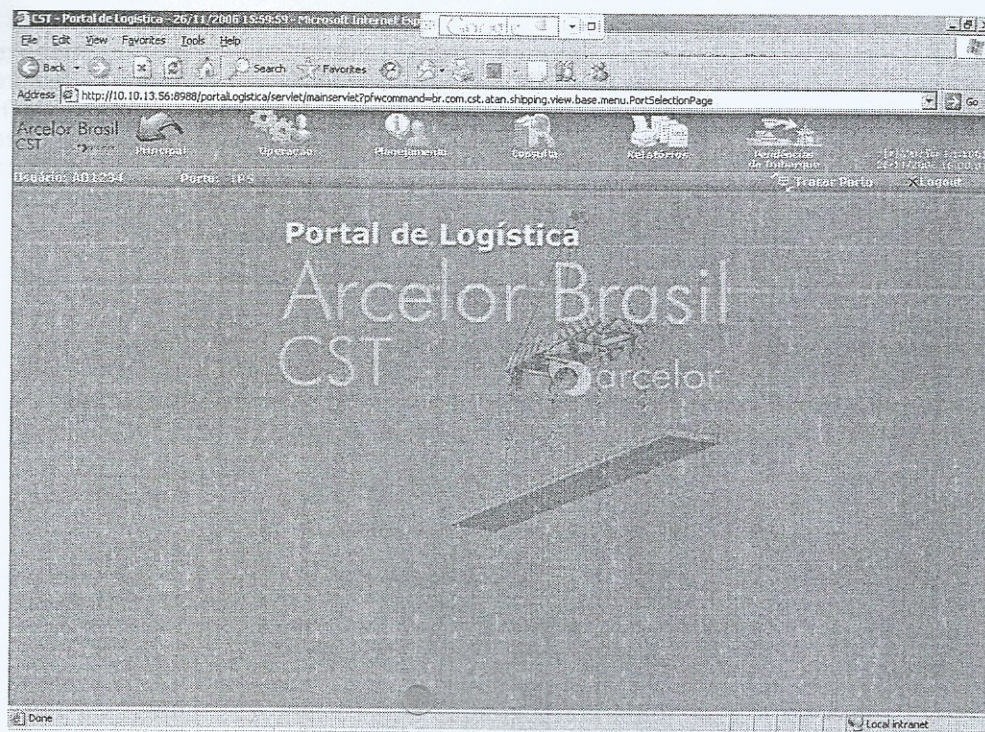


Figura 3 - Exemplo de Página de um Sistema Web desenvolvido pela Atan Sistemas

De modo genérico, podemos dizer que os sistemas desenvolvidos na ATAN têm esse tipo de interface, isto é, uma tela inicial com um menu, que leva para telas de cadastro, telas de operação, telas de relatórios etc.

Por trás das páginas HTML que constituem o sistema, existe um desenvolvimento feito em linguagens como C++, Java, Visual Basic entre outros, implementado sobre o Framework-Web da Atan.

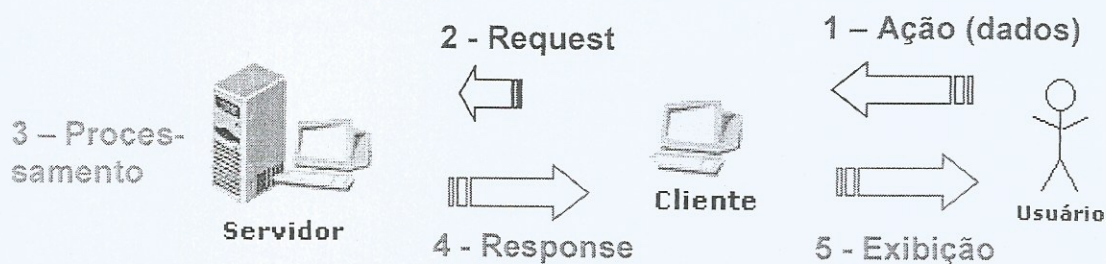


Figura 4 - Esquema de Arquitetura dos Sistemas Web na Atan Sistemas

Em baixo nível, o fluxo que o Framework^[1] determina, funciona como no desenho acima: O projeto é desenvolvido em uma certa linguagem de programação e disponibilizado em algum servidor. O usuário, com acesso a algum terminal faz requisições a esse servidor, que por sua vez responde exibindo os resultados em páginas HTML.

Uma das primeiras questões que deve surgir sobre o projeto é:

Por que não utilizar uma ferramenta já existente para geração de mapa de sites? Por exemplo, o Google Map Site Generator^[8]? Por uma simples razão. As ferramentas de geração de mapas existentes na Internet têm o seguinte mecanismo: Elas varrem um conjunto de arquivos HTML e constroem o mapa de site. A impossibilidade de utilizarmos uma ferramenta já pronta está no fato de que nos sistemas da Atan, as páginas "HTML" desenvolvidas não ficam armazenadas no servidor. Elas são construídas dinamicamente, a partir do código fonte, escrito em Java, C++ e outros.

Relação entre Web X Linguagens de Programação

Quando o desenvolvedor utiliza o Framework^[1] da Atan, ele abstrai-se da Web (de certa forma), pois desenvolve uma página HTML utilizando uma linguagem POO (Java), por exemplo. Para clarear definitivamente a relação entre linguagens de programação e Programação Web, vamos dar um exemplo de uma implementação de uma simples página HTML em Java.

```
public class PaginaExemplo extends WebPage
{
    public PaginaExemplo(WebPage p_Page) throws Exception
    {
        super(p_Page);
    }

    public void init() throws Exception
    {
        this.addOnShow(m_OnShow);
    }

    private Table m_Table = new Table(this);
    private OnShow m_OnShow = new OnShow(this)
    {
        public void exec() throws Exception
        {
            m_Table.add("Olá Mundo HTML!");
            m_Table.flush();
        }
    }
}
```

Figura 5 - Código-Fonte de uma Página HTML desenvolvida em JAVA

Acima está um exemplo simples de uma página "HTML". O Framework^[1] lê o código binário (compilado dessa classe) (que foi escrito em Java), realiza um parser e transformar o código Java em HTML.

Assim, supondo que um usuário fez uma requisição ao servidor de PaginaExemplo, o servidor irá ler PaginaExemplo.class e irá construir a PaginaExemplo.html dinamicamente, retornando o seguinte código para o usuário:

***colocar HTML

Já temos a resposta para a pergunta original desta seção: Como iremos saber quando existe um link de uma página para outra? Resposta: Essa informação está contida nos códigos-fonte do projeto. A transferência de fluxo de uma página para outra é feita via chamada de um comando.

Obs: Esse comando será explanado na seção 13 (Implementação).

Escopo do Projeto

É hora de delimitar até onde a solução que propomos irá valer. O Gerador de mapa de site será um componente específico, isto é, dependente da linguagem sobre o qual o sistema foi

construído. Neste projeto, propomos um componente Gerador de Mapas de Sites para sistemas web desenvolvidos em Java, sobre o Framework-Web da empresa Atan-Sistemas.

3.2. Parte Gráfica

Uma vez que definimos como iremos recuperar os links das páginas do Sistema, e pressupondo que já temos uma estrutura com esses links em mãos, devemos pensar em como disponibilizar isso para o usuário.

Um dos pré-requisitos do Projeto Gerador de Mapa de Site é que ele seja totalmente implementado sobre o Framework-Web da Atan. Isto é, que ele seja independente de outros softwares (desejável para a empresa). Assim sendo, a decisão tomada para desenhar o mapa de site foi utilizar a linguagem SVG^[9].

Infelizmente, durante o tempo deste projeto, não foi possível cumprir esse requisito, devido ao pouco tempo disponível. No meio da implementação, decidiu-se utilizar uma ferramenta externa para obter resultados concretos até a data de apresentação do Projeto. Mais adiante serão explicados os motivos para a tomada desta decisão.

A ferramenta externa escolhida para gerar o mapa de site, efetivamente foi o software: Graphviz^[5], software livre desenvolvido por AT&T Research Labs

4. Trabalhos Relacionados

Nesta seção, citamos alguns trabalhos e pesquisas relacionados a nosso projeto, inclusive a ferramenta externa que utilizamos para geração do desenho do mapa, o graphViz^[5].

4.1. GraphViz^[5]

GraphViz^[5] é, na verdade, não uma ferramenta, mas uma coleção delas para manipulação de estruturas e geração de layouts gráficos. Para utilizar tal software, basta escolher uma forma de executá-lo (ele disponibiliza várias) e dar, como artefato de entrada, um arquivo que descreva o grafo que quer se representar. O GraphViz^[5] possui, por exemplo, interfaces gráficas, interfaces via linha de comando, interface PERL, interface Java, etc.

O que mais interessaria no momento seria utilizar sua interface Java – uma vez que o gerador que estamos desenvolvendo está implementado em Java. Porém, o site oficial da ferramenta não disponibiliza uma versão atualizada compatível com a última versão do GraphViz^[5] que é disponibilizada. Foram encontradas e testadas outras API's Java para o GraphViz^[5] (independentes) porém, todas descontinuadas e incompatíveis com a versão do GraphViz^[5] utilizada neste projeto. Dessa forma, não houve como aproveitar alguma interface JAVA.

Os grafos gerados pelo GraphViz^[5] podem ser direcionados ou não-direcionados e podem conter ciclos. Apenas essas duas características já são suficientes para atender a demanda do Gerador de Mapas de Site que estamos desenvolvendo.

Uma das interessantes ferramentas que o GraphViz^[5] disponibiliza é a utilização da interface DOTTY. É uma interface muito simples e poderosa capaz de manipular três tipos de objetos: grafos, nodos e arestas. Essa interface é executada via linha de comando e recebe como entrada um arquivo no formato.DOT.

Arquivo DOT

DOT é uma linguagem textual de descrição de grafos. É uma linguagem muito intuitiva e simples para descrevê-los.

A seguir, mostramos um exemplo de um arquivo.DOT e sua respectiva saída, utilizando-se a ferramenta DOTTY - presente no software GraphViz^[5].

extensão
a idê

```
digraph G {
  node [shape=box]
  { node [shape=plaintext, fontsize=16];
  0 -> 1 -> 2; }
  Pagina3 -> Pagina1[label="1"];
  Pagina2 -> Pagina3[label="2"];
  Pagina2 -> Pagina2[label="1"];
  Pagina1 -> Pagina2[label="1"];
  { rank = same ; 0;Pagina1; }
  { rank = same ; 1;Pagina2; }
  { rank = same ; 2;Pagina3; }
}
```

Figura 6 - Arquivo .dot

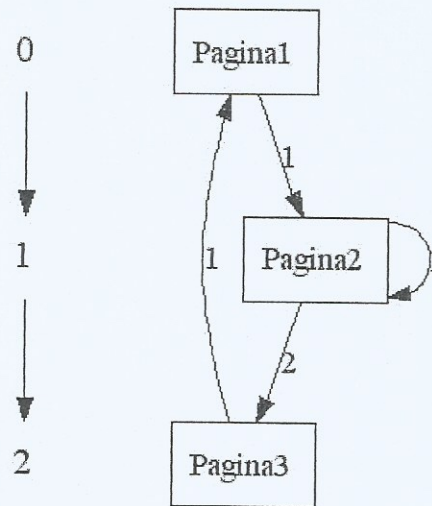


Figura 7 - Figura gerada pelo software
GrapViz

A sintaxe do arquivo.dot é muito simples. Para cada aresta, escreve-se do lado esquerdo, o nodo que aponta, logo o símbolo "->" (que simboliza uma aresta) e do lado direito, o nodo apontado. Isso é tudo que precisamos, a princípio.

- Nodos → que são as classes
- Arestas → que são os links entre os nodos.

Logicamente que o exemplo acima é ilustrativo. Existem vários detalhes que devem ser levados em conta, na hora de gerar a imagem. Outros detalhes do arquivo.DOT são dados na seção 5

5. Implementação

A resposta para a primeira parte do nosso projeto é buscar os "links" do sistema no código original do projeto, isto é, na codificação feita em Java, basicamente.

5.1.Parser

Passos a serem seguidos para realização da primeira parte (o parser)

- Varredura do Projeto
- Varredura do Código-Fonte

A implementação desses passos é explicada detalhadamente a seguir.

Varredura do Projeto

Este passo do nosso parser talvez não se difira tanto dos outros tipos de Geradores de Mapas de Site. A idéia é iterar sobre um conjunto de arquivos e armazenar aqueles que são relevantes. Arquivos relevantes são aqueles que possuem pelo menos um *link* para outra página. O que muda aqui é que: no lugar de armazenar arquivos *.html*, armazenamos arquivos com extensão *.java* (que são aqueles que contém a informação dos *links*).

A idéia informal é a seguinte:

- O programa recebe como entrada um caminho (path).
- O programa itera sobre os diretórios contidos dentro desse caminho.
- O programa armazena em uma estrutura (hash) os nomes dos arquivos *.java*

Inicialmente havia-se utilizado um simples array, com os nomes dos arquivos. Ao decorrer da implementação, além desse array, utilizou um Hashtable para guardar a relação entre os arquivos. (quem aponta para quem e quem estende de quem).

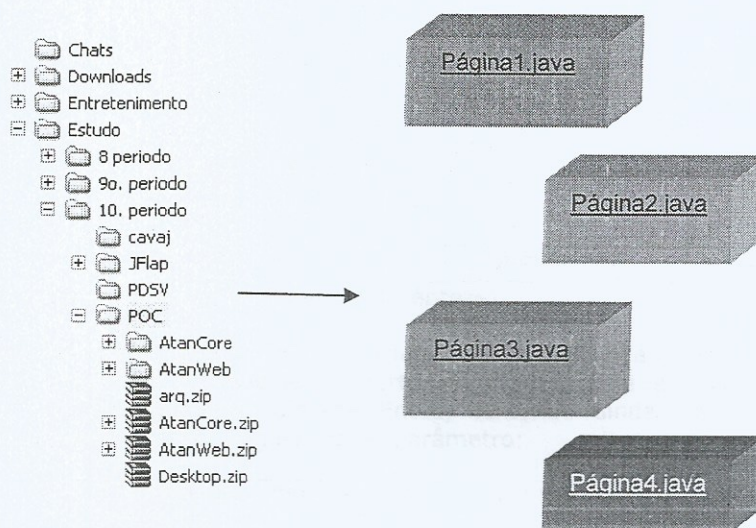


Figura 8 - 1º. Passo do Parser

Varredura do Código-Fonte

Uma vez armazenados em memória os nomes de todos os nomes dos arquivos que serão varridos, é hora de procurar os links de cada página. A idéia é simples: basta procurar o comando que realiza a transferência de fluxo de uma página para a outra no Framework^[1] da Atan.

O Comando Transfer

No Framework-Web da Atan, toda página (salvo raras exceções) de um Sistema é uma classe Java que estende da Classe WebPage.

Essa classe possui o método transfer, responsável por transferir o fluxo de uma página a outra. Para que ele funcione, basta que o desenvolvedor faça a seguinte chamada:


```
    //código
    .
    .
    transfer(p_parametro);
    .
    .
    //código
```

Sendo que p_parametro é do tipo WebPage (uma página também).

A assinatura do método é a seguinte:

```
public void transfer(Webpage p_Page) throws Exception
```

O problema parece simples. Resume-se a encontrar uma string dentro do código fonte - a string "transfer". Para fazer essa busca, basta abrir o arquivo com o código fonte, colocá-lo dentro de uma String e chamar o método "matches" da classe String, passando como parâmetro "transfer".

Infelizmente, de cara, tal algoritmo falha, quando, por exemplo, no código fonte existe uma método chamado: "**transferePagina()**". O match retornaria true para um resultado incorreto.

A primeira correção desse algoritmo seria fazer um match com a seguinte string: "**transfer**". Tal contorno, no entanto, também falha. Basta lembrar que o programador pode decidir fazer a seguinte chamada ao método transfer:

```
transfer      (parâmetro);
```

(note os espaços entre o comando transfer e o parêntese).

Como a linguagem Java permite várias construções diferentes para um mesmo significado sintático, o parser tem que ser "esperto" o suficiente para lembrar de todos os casos de ambigüidade que a linguagem "oferece". Então, tentando ainda contornar o problema, poderíamos passar uma expressão regular como parâmetro:

```
Matches("transfer[ ]*[/t]*[/n]*")
```

Traduzindo a expressão regular: procure a palavra "transfer", seguida por uma combinação de enter's, tabs e espaços, seguida de um abre-parêntese. Isso resolve o nosso problema - claro, se não existissem mais casos a serem tratados.

Suponha que na classe Página do desenvolvedor exista a seguinte linha de código:

```
System.out.println("chamando o método transfer()");
```

Isso daria um crash na expressão regular passada anteriormente, afinal, ela iria casar com a string "transfer(" que está dentro da string impressa. Poderíamos pensar em uma expressão regular que, aliada a anterior, tapasse esse buraco - isto é, ignore strings que estejam dentro de strings. O problema é que existem vários outros buracos ainda a serem tratados, como por exemplo:


```
//chamando o método transfer();

/*
//
chamando o método "transfer()";
*/

String v = "teste" +
"chamando o transfer()"
```

etc.

Sendo assim, o gasto em conseguir tal expressão regular poderia ser uma aposta arriscada. Garantir que ela funcione para todos esses casos de ambigüidade da linguagem é difícil. Entre uma expressão regular ou um autômato feito à mão, optamos por um autômato feito à mão. Trabalhando com um autômato feita a mão, além do fato de saber exatamente em qual string "estamos pisando", temos o argumento de que implementado-o de forma genérica ele será reutilizado em outras partes do projeto - como a busca pela palavra `abstract` e `extends` - explicada mais adiante.

Feita a implementação do autômato....

Armazenados os nomes dos arquivos em uma hashtable, percorremos cada código fonte de cada arquivo, um a um, a procura de chamadas ao método `transfer`. Se a procura retorna sucesso, guardamos a informação de qual página está recebendo o fluxo no sistema.

Para saber qual página está recebendo o fluxo do sistema, inicialmente, temos um problema relativamente simples, que é o de pegar, diretamente, o que há dentro do parêntese na chamada do método `transfer()`.

```
transfer(new Pagina2())
```

Nesse caso (realmente, o de mais comum utilização nas implementações dentro do framework^[1]), o parser construído consegue recuperar tudo que está dentro dos parênteses "new `Pagina2()`".

Finalmente faz-se um tratamento simples, ignorando a palavra `new` e os parênteses. Porém, ainda não estamos num mar de rosas.

Dificuldades do Parser

Além do conseguido até o momento, precisamos tapar buracos de outra natureza. Suponha que o desenvolvedor resolva fazer a seguinte chamada ao método `transfer`:

```
Pagina2 v_Pagina = new Pagina2()
transfer(v_Pagina)
```

Isso é perfeitamente possível e possui o mesmo valor semântico que o primeiro exemplo dado anteriormente. Note que existem vários outros tipos possíveis de se chamar o método `transfer`. Listamos alguns:


```

public class Pagina1 extends WebPage
{
    Pagina2 m_Pagina2 = new Pagina2();
    WebPage m_PaginaWeb;

    public void transfere2()
    {
        transfer(m_Pagina2);
    }

    public void transfereWeb()
    {
        transfer(m_PaginaWeb);
    }

    public void transfereP(Pagina3 p_Pagina3)
    {
        transfer(p_Pagina3);
    }

    public void transfereF()
    {
        transfer(transfereFuncao());
    }

    private WebPage transfereFuncao()
    {
        return m_Pagina2;
    }
}

```

Por membro simples

Por membro definido em classe herdeira

Por página passada como parâmetro

Por chamada de função que retorna uma página

Figura 9 - Exemplos de submits

Ainda podem ser citados vários outros: Transfer por chamada de uma função que corresponde a outro membro da classe, transfer por chamada de função que está na classe herdeira (ou classe pai) (ou em ambas) e transfer via JavaScript. É possível (apesar de não recomendado) dentro do Framework^[1] inserir código JavaScript dentro das páginas Java implementadas.

Dessa forma, observa-se que o programador tem muita liberdade no modo em que ele transfere o fluxo de uma página para outra. Essa liberdade significa mais trabalho na implementação do gerador, logicamente.

O que foi implementado:

Foi preciso setar prioridades sobre as tarefas. Implementou-se a solução para os seguintes casos, apenas:

- ➊ transfer(new Pagina())
- ➋ transfer(v_Variavel)
- ➌ transfer(m_MembroDeclaradoNoProprioArquivo)
- ➍ transfer(p_Parametro)

A razão dessa decisão é que, para a maioria dos casos, o parser irá gerar o mapa de site corretamente. Para os casos não cobertos (inclusive o caso do JavaScript), o parser do gerador simplesmente não exibirá transferências de fluxos no resultado final (a figura).



Figura 10 - Resultado do Parser (em memória)

5.2. Desenho do Mapa

A 2ª. Parte do projeto (gráfica) pode, também, ser dividida em 2 passos:

- Criação de um arquivo de entrada para o software externo: DOT.
- Geração dinâmica da figura final, numa página HTML.

Criação do arquivo .DOT

No primeiro passo desta segunda parte precisamos descrever a estrutura de classes e links (feitos na primeira parte) em forma de um arquivo.DOT – arquivo de entrada para o software GraphViz^[5].

Para isso, percorremos a hashtable com as ClassesPaginas do projeto e começamos a gerar o arquivo de saída "Mapa.dot". A classe principal desta parte do projeto é a classe: Gerador.java. Nessa classe, o método mais importante é o método "gerarArquivoDoMapa". Ele é responsável por iterar sobre o hashtable, pegar todas as informações que precisa sobre cada classe, e jogar num arquivo intermediário (o .DOT).

Nessa parte do gerador, para cada classe, é impresso no arquivo de saída as seguintes informações:

- Nome da classe sendo tratada
- Lista de classes apontadas (links)
- Tipo da classe (abstrata ou não)
- Hierarquia da Classe

Hierarquia da Classe

Aqui, introduzimos o conceito que chamamos de Hierarquia da Classe.

O GraphViz^[5] resolveu uma boa parte do problema ao "ler" a estrutura hashtable e transformá-la numa figura. No entanto, se não especificamos como queremos desenhar essa estrutura, ele simplesmente "cuspirá" as classes na figura de qualquer forma.

Desejariamos que houvesse uma certa organização nessa figura, isto é: Não queremos que as classes sejam desenhadas de qualquer forma. Queremos que a classe base do projeto (por exemplo: a página de menu) apareça no topo da figura, e que as classes chamadas por essa página base (os menus) apareçam logo abaixo da classe base. Gostaríamos, em seguida, que as classes chamadas pelas páginas do segundo nível aparecessem no terceiro nível da figura e assim sucessivamente.

Mostramos a seguir um pequeno exemplo da evolução nesse conceito, dentro do nosso projeto.

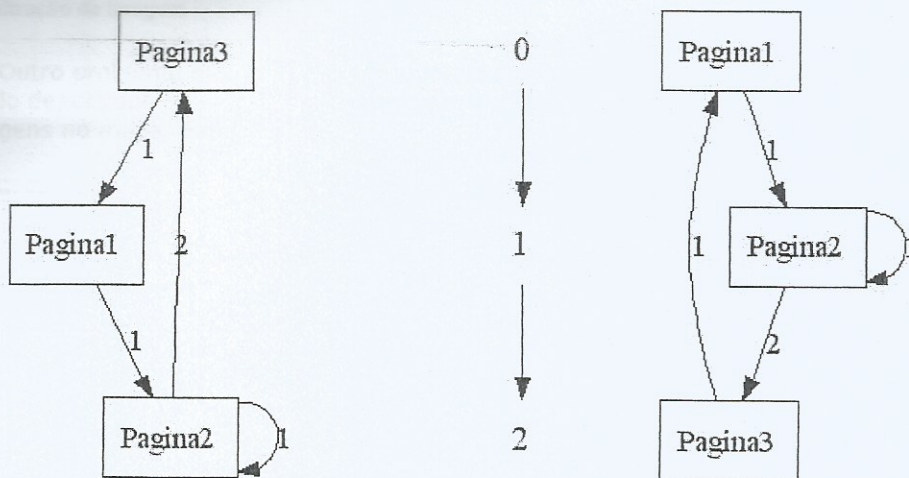


Figura 11 - (a) Grafo Hierarquizado (b) Grafo Não-Hierarquizado

A primeira figura é um exemplo de um mapa de site de um projeto simples (de 3 paginas apenas) ainda não hierarquizado. Isto é, note que o GraphViz^[5] não tinha nenhuma informação de qual era a página principal. Ele apenas desenhou o grafo da forma que "achou" mais conveniente.

Já, na figura seguinte, temos um exemplo melhorado. Para obter tal resultado, "dissemos" para o GraphViz^[5] que a página principal era a Página1, e a partir daí, podemos determinar uma hierarquia de importância de páginas.

O problema da hierarquização – assim como no parser – a princípio parece bem simples. Porém, começamos a tropeçar em alguns casos à parte. Suponha que no projeto acima, houvesse mais um link da página 1 para a pagina 3. Nesse caso, em que nível hierárquico a Página 3 deveria ficar? (nível 3 ou nível 2?). Tomar a decisão é fácil: digamos que quando esse caso ocorre, assumimos que "queremos" o seguinte resultado:

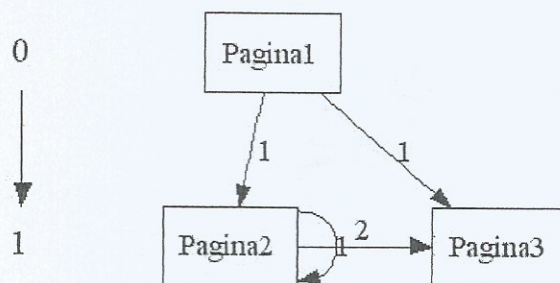


Figura 12 - Exemplo de Projeto com relacionamentos de hierarquias ambíguas

A decisão, sim, é fácil, mas a codificação fica um pouco mais complexa, na medida em que temos que considerar todos os relacionamentos de uma classe, e não apenas 1, para poder decidir a qual nível hierárquico ela pertence. Além dessa complexidade adicional no código, temos que enfrentar outros problemas, do tipo:

Suponha que Pagina2 seja uma classe abstrata e que Pagina3 herde de Pagina2. Nesse caso, seria interessante mostrar as classes abstratas no grafo, e caso sim, devemos mostrar seus relacionamentos?

O caso das classes abstratas foi levado em conta e colocado como uma opção no gerador. O desenvolvedor que utiliza o componente pode decidir imprimir ou não as classes abstratas.

Outro problema que precisou ser enfrentado – e esse foi um problema que só foi percebido depois que os resultados começaram a aparecer – foi o problema da horizontalização das imagens no mapa. Explica-se:

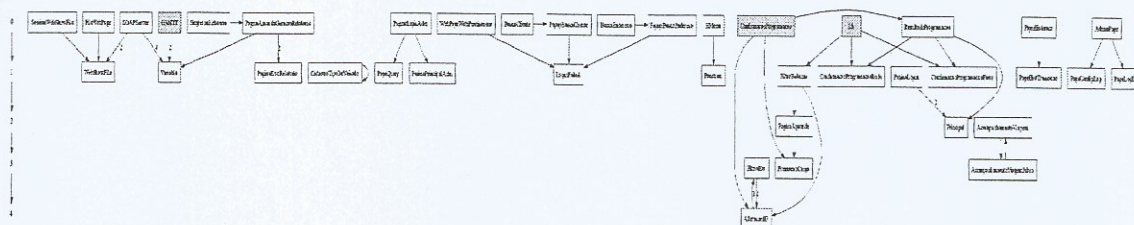


Figura 13 - Exemplo de Mapa de Site com horizontalização excessiva

A figura acima é um resultado parcial de um sistema feito na ATAN. Note que ele tem um problema: a figura está muito extensa, isto é, sua largura prejudica a visualização da imagem. O ideal seria que houvesse um limite máximo de classes que devem aparecer na mesma linha, fazendo com que a visualização fique melhor para o usuário final.

Consequimos amenizar esse problema de forma parcial, aproveitando-se do fato de grafos desconexos em um mesmo projeto.

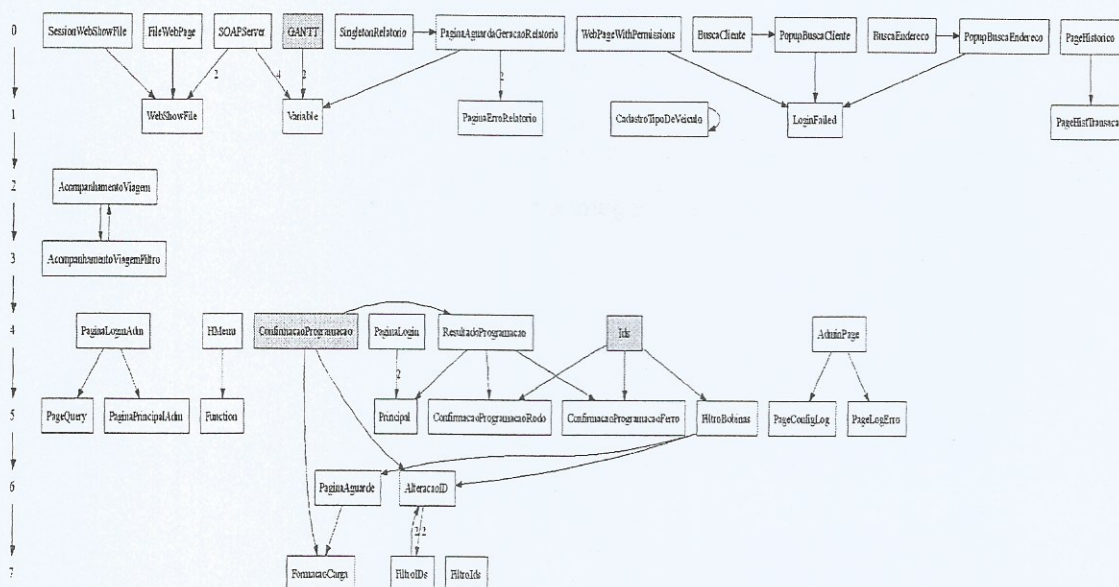


Figura 14 - O mesmo exemplo da Figura 9 com a horizontalização excessiva eliminada

A figura acima corresponde à mesma figura anterior, com uma melhoria implementada. Os vários grafos foram reposicionados para uma melhor visualização. Veja que o projeto possuía 3 fluxos principais – 3 grafos desconexos – e por essa razão conseguimos alterar a hierarquia de classes.

Instanciação do Mapa De Site

Finalmente, chegamos ao último passo do gerador, que é mostrar o mapa gerado para o usuário. A ideia inicial do projeto era fazer toda a parte gráfica utilizando-se SVG^[9] (Scalable Vector Graphics), porém, devido à insuficiência no tempo, foi decidido utilizar o software GraphViz^[5].

Para poder realizar isso, foi preciso um pequeno dispêndio de tempo para integrar o gerador implementado com o software externo. Como se trata de um software desenvolvido em C++, foi preciso instanciar um Web-Service configurado no ambiente de trabalho da ATAN.

Web-Service para GraphViz^[5]

Obs: Para maiores detalhes sobre Web-Service, vide apêndice B.

Não se perdeu muito tempo com a configuração do ambiente, já que é comum utilizar programas externos, na ATAN, por meio de web-services. Tal Web-service foi instanciado em um dos servidores da ATAN com a disponibilização do método principal:

```
Public String gerarFigura(String p_ArquivoDot String p_TipoArquivo)
```

Tal método é responsável por executar o GraphViz^[5] através da linha de comando. O Web-service recebe a requisição tendo como parâmetros o fonte para o arquivo.DOT e o tipo de arquivo que ela irá gerar. Do lado do servidor (onde o GraphViz^[5] está instalado), é gravado um arquivo com a descrição do grafo em memória secundária e o software é executado via linha de comando:

```
>dot -Tj<p_TipoArquivo> Dot.txt -o Mapa.<p_TipoArquivo>
```

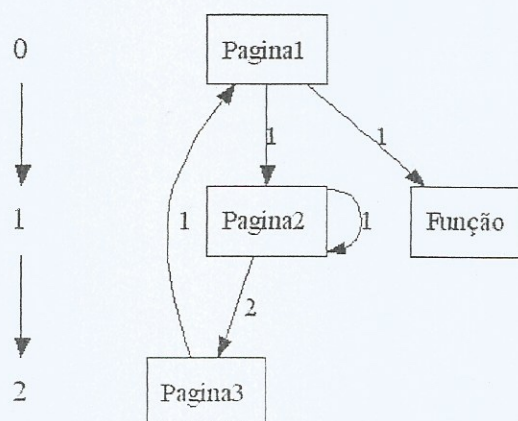
Onde p_ArquivoDot é o um código fonte na linguagem DOT. Para o desenvolvedor Java, a chamada via Web-Service fica transparente. Ele simplesmente recebe como retorno o caminho da imagem onde a figura foi gravada.

Isso já é o suficiente para instanciar uma imagem no Framework^[1] da ATAN.

6. Problemas Enfrentados

No projeto desenvolvido, nem todos os problemas foram solucionados. Alguns dos problemas que tomaram mais tempo no projeto foi à busca pelos *links* do sistema (casos excepcionais) e a forma da representação gráfica do mapa.

6.1.Problema 1 – Submits Excepcionais



```
public void transfere2()
{
    transfer(recuperaPagina());
}

private WebPage recuperaPagina()
{
    return new Pagina1();
}
```


Figura 15 - Exemplo de Submit-Excepcional no código-fonte, e o seu resultado no mapa de site

A figura mostra o mapa de site contendo uma transferência de fluxo mais complexa já dada como exemplo em outras seções. Trata-se da chamada a uma função, dentro da chamada ao método transfer. Esse é um caso de chamada que não conseguimos tratar no escopo deste projeto. Por certo é uma demanda para "releases" seguintes.

6.2.Problema 2 – Horizontalização excessiva:



Figura 16 - Mapa de Site com Horizontalização Excessiva sem grafos desconexos

Na seção 5.2 falamos um pouco sobre o problema da horizontalização excessiva e como ele foi abordado para ter seus efeitos amenizados. Porém, o problema não foi resolvido por completo. A figura acima mostra outro Mapa de site com horizontalização excessiva, com um agravante: Existem muitas classes na mesma hierarquia que possuem alguma relação. Assim, o algoritmo desenvolvido não encontra grafos desconexos e é "obrigado" a desenhar tudo na mesma hierarquia.

Esse tipo de problema é mais complicado porque interfere no algoritmo que seta a hierarquia das páginas. O algoritmo precisa ser inteligente o suficiente para modificar a hierarquia de alguma classe quando necessário, isto é: deve existir um algoritmo de decisão que decide se é melhor abrir mão da hierarquia da classe ou melhor abrir mão da posição da classe na figura.

A solução não entrou no escopo deste projeto e é uma idéia para releases futuras.

7. Utilização

A utilização do gerador desenvolvido é bastante simples. Como se trata de um componente do framework^[1], basta que o usuário o encare como qualquer outro componente. Por exemplo, numa pagina normal, o usuário pode declara um componente do tipo Button (um botão HTML), assim:

```
Private Button m_BotaoOk = new Button(this, "Botão Ok")
```

Para o gerador, a idéia é a mesma. Basta que ele declare o gerador, assim:

```
Private Gerador m_GeradorDeMapa = new Gerador(this, new Class[]  
{PaginaBase.class})
```

O gerador precisa de duas informações principais para rodar: o caminho do projeto (para poder iterar sobre o código fonte e a classe base do projeto). O caminho do projeto fica normalmente num arquivo de configurações (config.properties) padrão utilizado pela Atan Sistemas. Quanto à classe base, o usuário passa um vetor de objetos do tipo Class que é o conjunto de classes base do sistema (pode existir mais de uma classe base).

Posteriormente no código, o usuário deve chamar a função "gerarMapaDeSite" assim:


```

String p_CaminhoDaImagem = m_GeradorDeMapa.gerarMapaDeSite();
Img v_Imagem = new Img(this, p_CaminhoDaImagem);
Table v_Table = new Table(this);
V_Table.add(v_Imagem);
V_Table.flush();

```

O método é chamado e retorna a string com o caminho da imagem que ele gerou. Assim, uma sugestão de uso dessa imagem é feita no código acima, onde ele utiliza o caminho para instanciar uma imagem (objeto `Img` já existente no Framework^[1]) para instanciar a imagem através de um caminho presente na hierarquia de diretórios do projeto. Finalmente ele pode incluir tal imagem numa tabela (objeto `Table` do FW), por exemplo, e apresentar tal tabela na página.

8. Resultados

A seguir apresentamos alguns resultados obtidos pela ferramenta desenvolvida:

Sistema Despacho de Bobinas - CST

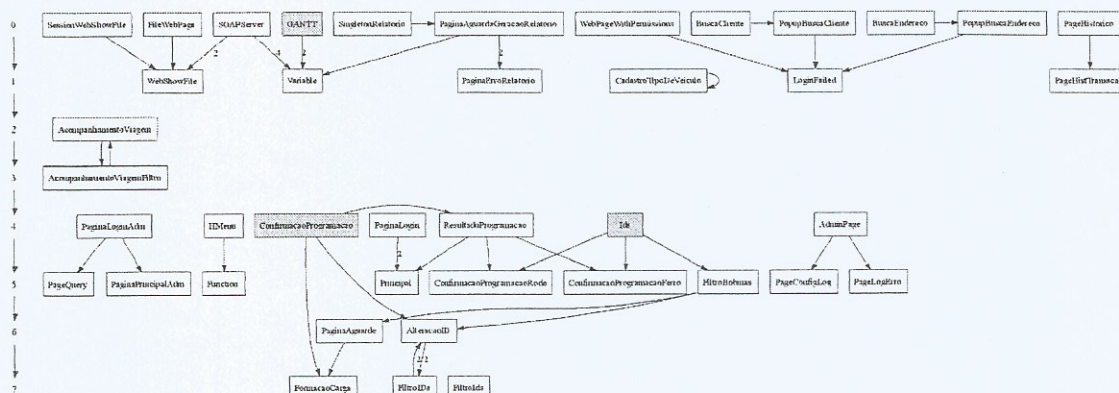


Figura 17 - Mapa de Site do Sistema Despacho de Bobinas Terrestres - CST

Nem todos os resultados puderam ser apresentados dentro do respectivo projeto por não haver permissão de executar o projeto dentro da Empresa.

Por questões de estrutura, não é possível disponibilizar uma versão completa do gerador implementado na Internet. Porém, o código de uma versão demo está disponibilizado em <http://www.dcc.ufmg.br/~diegobs/gerador/src>. Essa versão tem como saída o arquivo.dot, pronto para ser utilizado pelo software GraphViz^[5].

Também é disponibilizado um projeto teste em: <http://www.dcc.ufmg.br/~diegobs/gerador/projetoWeb>, como exemplo de funcionamento do gerador.

9. Apêndices

No projeto inicial, a parte gráfica do Mapa seria desenvolvida em SVG^[9]. Houve um esforço para alcançar esse objetivo, porém, ele foi interrompido no meio do desenvolvimento por falta de tempo. Ainda assim, por ser um objetivo futuro transferir a parte gráfica do GraphViz^[5] para SCG, o que foi desenvolvido em SVG^[9] não é ignorado.

SVG^[9]

SVG^[9] - sigla para Scalable Vector Graphics – é uma linguagem XML que serve para descrever gráficos bi-dimensionais, estáticos ou animados. É um padrão aberto criado pela W3 (World Wide Web Consortium).

Pelo próprio site da W3, temos a definição:

"SVG^[9] é uma linguagem que descreve gráficos bi-dimensionais e aplicações gráficas em XML. É uma recomendação da W3C."

Basicamente, SVG^[9] é uma linguagem que aceita três tipos de objetos gráficos:

- Formas gráficas vetoriais (linhas, curvas, áreas).
- Imagens Digitais (Raster Graphics)
- Texto

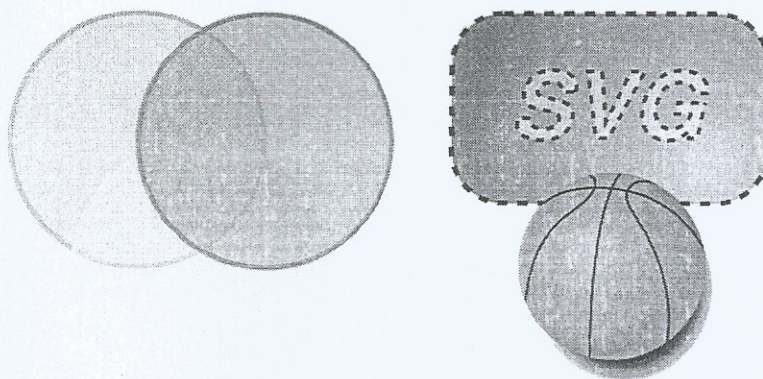


Figura 21 - Exemplo de Figuras desenvolvidas na linguagem SVG

Durante o projeto, foi desenvolvida uma infra-estrutura em SVG^[9] dentro do Framework-Web da Atan para instanciamento de objetos do tipo SVG^[9]. Correspondem também a componentes dentro do Framework^[1]. É possível, por exemplo, instanciar um documento SVG^[9] (dentro de uma página) e inserir figuras como, círculos, quadrados, polígonos diversos, linhas, pontos, etc.

Uma das razões que determinou a desistência momentânea da utilização dessa infraestrutura neste projeto é que o algoritmo para cálculo da posição dentro da figura SVG^[9] demandaria um tempo bem maior do que o se tinha disponível.

Ainda assim, o código desenvolvido (que será utilizado posteriormente) está disponível em: <http://www.dcc.ufmg.br/~diegobs/gerador/svg>

Web-Services

Web-Service não é um software, nem está atribuído a uma única linguagem. Um Web-Service pode ser visto como uma solução utilizada na integração de sistemas e aplicações diferentes. Com essa tecnologia é possível "conectar" novas tecnologias com tecnologias já existentes. A grande sacada desta solução é utilizar o XML como comunicação entre as partes. O XML é como se fosse a língua universal em que todas as partes envolvidas conversam.

Para as empresas, os Web services podem trazer agilidade para os processos e eficiência na comunicação entre cadeias de produção ou de logística. Toda e qualquer comunicação entre sistemas passa a ser dinâmica e principalmente segura, pois não há intervenção humana.

Apesar dessa visão ideal que demos acima e do objetivo principal de web-services ser a padronização, pode haver confusão na definição de um web-service. Se olharmos a definição de um web-service no próprio site da W3, nos diz:

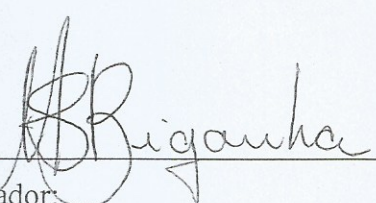
"Definição: Web-service é um software implementado para suportar interoperabilidade e interação entre máquinas em uma rede. Ele possui uma interface descrita em uma linguagem chamada WSDL. Outros sistemas interagem com Web-Service utilizando mensagens descritas em SOAP (um tipo especial de XML), utilizando protocolo http em conjunto com outros padrões da Web".

Como dito acima, a tecnologia envolve várias outras tecnologias. Para maiores detalhes – já que o foco aqui não é entrar em detalhes – consulte as referências.

Neste projeto, o Web-Service foi importante para realizar a comunicação de um projeto escrito em Java com um software feito em C++. Para isso foi necessário configurar uma DLL em um componente COM+ para que fosse disponibilizado como um web-service. Para a "parte Java" do projeto, fica transparente a chamada ao método, assim como em qualquer web-service.

10. Referências

- [1] Framework - Conceitos Gerais , em:
<http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/oque.htm> ,último acesso:
10/12/2006
- [2] Neto, B., I., Framework - Treinamento Interno - Atan Sistemas, 2006. 30p.
- [3] Neto, B. I., Salles, G. - Framework - Introdução - Atan, 2006.38p.
- [4] Gama, E., Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional, 1st edition 1995.395p.
- [5] Graph Visualization Software, em: <http://www.graphviz.org> - último acesso em
10/12/2006
- [6] Graph Viz Java API, em
<http://www.loria.fr/~szathmar/off/projects/java/GraphVizAPI/index.php> - último acesso em
10/12/2006
- [7] Gansner, E., Drawing Graphs with dot, 2006, 12p.
- [8] Google Site Map Generator, em <http://www.auditmypc.com/free-sitemap-generator.asp>,
último acesso em 10/12/2006
- [9] SVG Tutorial, em: <http://www.w3schools.com/svg/default.asp>, último acesso em
10/12/2006
- [10] World Wide Web Consortium - SVG Oficcial Site, em <http://www.w3.org/Graphics/SVG>,
último acesso: 10/12/2006
- [11] World Wide Web Consortium - Web Service Oficcial Site - em
<http://www.w3.org/2002/ws/>, ultimo acesso: 10/12/2006
- [12] The Java Sun Web-Service Page, em <http://java.sun.com/webservices>, ultimo acesso:
10/12/2006


Orientador:

MarizaAndrade da Silva Bigonha