

CAPÍTULO 2

JAVA E O PROJETO DO SOFTWARE

2.1 – INTRODUÇÃO

Softwares educacionais são importantes ferramentas complementares no ensino de engenharia. Isso se deve ao fato de estes *softwares* oferecerem meios convenientes para manipulações matemáticas e para visualização dos fenômenos físicos associados às matérias de engenharia.

Com o crescimento da Internet, cada vez mais negócios são realizados pela *web*. A área educacional também está se adaptando a essa tendência. Cursos à distância são cada vez mais oferecidos, inclusive cursos de nível superior e de pós-graduação[8].

O ensino à distância ou EAD é uma forma de ensino que possibilita a auto-aprendizagem com a mediação de recursos didáticos sistematicamente organizados. Além disso, estes recursos são apresentados em diferentes suportes de informação, são utilizados isoladamente ou combinados e são veiculados pelos meios de comunicação, dando destaque a autonomia do indivíduo [9]. Com a popularização da Internet, este meio tornou-se o meio mais básico de acesso às mais diversas informações, possuindo diversas vantagens sobre os outros meios como:

- É flexível – a qualquer hora ou lugar pode-se acessar o curso, desde que haja os recursos mínimos (computador ligado à Internet e etc.);
- É dinâmica – facilmente atualizável e possibilita o contato direto;
- É aberta – além do ambiente virtual criado para o curso, permite pesquisa em diferentes "lugares" na Internet;
- É sem fronteiras internacionais – pode-se atingir pessoas presentes em qualquer parte do mundo;
- É amigável – requer do aluno mínimos conhecimentos de navegação;
- É adaptável às necessidades do aluno – adequa-se à formação continuada de profissionais que não podem interromper suas atividades de trabalho e também não podem se deslocar para participar de aulas presenciais.

No Laboratório de Eletromagnetismo Aplicado (LEA) do Departamento de Engenharia Elétrica e Computação da UFPA, vem-se desenvolvendo *softwares* para

auxílio ao ensino de eletromagnetismo e de telecomunicações[3]. Para isso, utilizam-se ambientes como o Mathematica e linguagens nativas como o Delphi. Este trabalho faz parte de uma evolução natural dessa linha de pesquisa que é o desenvolvimento de *softwares* baseados em *web* para auxílio ao estudo de eletromagnetismo[5].

2.2 – SOFTWARES EDUCACIONAIS NAS ENGENHARIAS

Matlab e Mathematica são ambientes computacionais muito usados em engenharia, tanto na área de ensino quanto na área profissional. Tais ambientes possibilitam a criação de *softwares* que proporcionam facilidades em manipulações matemáticas e boas rotinas para visualização de resultados. Entretanto, esses *softwares* somente podem ser executados na presença daqueles ambientes. Para contornar esse problema, linguagens como C++, Visual Basic e Delphi são largamente utilizadas (com excelentes resultados) na criação de arquivos executáveis independentes. Porém, geralmente, limitam-se a receber dados e a fornecer os resultados correspondentes. Algumas vezes, apresentam gráficos ao usuário.

No entanto, isso não é suficiente para programas destinados à área de ensino, pois *softwares* voltados à área educacional devem ser bastante interativos e intuitivos.

A grande motivação deste trabalho é dar um primeiro passo rumo ao que virá a ser uma espécie de livro eletrônico. Neste, o usuário terá um ambiente amigável para aprendizado de eletromagnetismo baseado em texto. Mas que, além disso, permitirá a manipulação de ferramentas gráficas que possibilitarão uma compreensão mais clara dos fenômenos físicos envolvidos nessa matéria. Em vista disso, decidiu-se que o ambiente objeto deste trabalho será desenvolvido baseado no modelo de *applets* Java.

Dessa forma, tem-se a linguagem HTML como boa ferramenta para formatação de textos e, principalmente, o ambiente desenvolvido poderá ser disponibilizado diretamente na Internet. Podendo, assim, ser acessado a qualquer hora, de qualquer lugar, de computadores baseados nas mais diversas plataformas e operando com diferentes sistemas operacionais.

Assim, haverá um ambiente no qual gráficos, equações, texto e etc., juntos auxiliarão aos alunos em seus estudos na matéria de eletromagnetismo.

2.3- VANTAGENS DA LINGUAGEM JAVA

Java é uma linguagem de programação desenvolvida pela Sun Microsystems. Originalmente ela foi concebida para ser usada com a televisão interativa, porém Java tornou-se uma linguagem de grande interesse para a comunidade Internet quando a Sun lançou o HotJava, um navegador de *web* que podia executar pequenos programas Java embutidos dentro das páginas *web*, chamados *applets*. Logo depois, o suporte para *applets* Java foi acrescentado em outros navegadores.

Java é notável não apenas porque *applets* do Java podem ser executados dentro de páginas *web*, mas também porque ele é uma linguagem orientada a objetos com muitos recursos associados. A linguagem Java lida com muitos dos problemas comuns porém complexos que os programadores geralmente encontram ao desenvolver aplicações robustas. Java suporta multiprocessamento com sua classe *thread* (linhas de execução) e executa automaticamente a coleta do lixo, liberando memória que não está mais sendo usada em segundo plano. A Java *Application Programming Interface* (API), incluída no Java *Developers Kit* fornecido pela Sun, dá aos programadores acesso independente de plataforma para ferramentas essenciais na programação de aplicações Internet complexas, como *sockets* de rede e um sistema de janelas gráficas [10].

Java torna realidade o ideal de independência de plataforma porque ela é uma linguagem pseudo-compilada. Os *applets* Java podem ser executados em qualquer máquina que possa executar um navegador *web* com recursos Java. Java é a primeira linguagem de programação de alto nível conhecida que é verdadeiramente independente de plataforma.

A independência de plataforma é a capacidade de o mesmo programa ser executado em diferentes plataformas e sistemas operacionais. Quando um programa escrito em C ou na maioria das outras linguagens é compilado, o compilador transforma o arquivo fonte em código de máquina. Se o código do programa for compilado em um sistema Intel, o programa resultante poderá ser executado em outros sistemas baseados em Intel, mas não funcionará em computadores com outra arquitetura. Se for necessário a utilização do mesmo programa em outras plataformas, deve-se transferir seu código fonte para a nova plataforma e recompilá-lo para produzir o código de máquina específico para esse sistema.

Os programas Java atingem essa independência através da utilização de uma máquina virtual. A máquina virtual pega os programas Java compilados e converte

suas instruções em comandos que um sistema operacional possa manipular. O mesmo programa compilado pode ser executado em qualquer plataforma e sistema operacional que possua uma máquina virtual Java.

2.4- PROJETO DO SOFTWARE

Java é particularmente conveniente para o uso em aplicativos para a Internet, mas também é uma linguagem de propósito geral [11]. Pode ser usada com bons resultados para gerar arquivos independentes, assim como C, Pascal e etc.

Um dos motivos do grande sucesso do Java é que com ela pode ser criado o que se chama de *applet*. Um *applet* é uma aplicação que é executada quando se acessa uma página *web*. Assim como toda aplicação, um *applet* também tem uma área de trabalho. Em se tratando de um *applet*, essa área fica dentro da janela do *browser* ou navegador.

Os *applets* são pequenos aplicativos que são acessados num servidor da Internet, transportados pela rede, instalados automaticamente no seu computador e executados como parte de um documento *web*. Depois que um *applet* chega ao cliente, ele pode produzir uma interface de usuário multimídia e executar cálculos matemáticos complexos.

Existem dois modos diferentes de se executar um arquivo executável de Java:

- Como um programa independente que pode ser chamado a partir de uma linha de comando. Isso é chamado de “aplicativo”;
- Como um programa embutido em uma página *web*, a ser executado quando a página é navegada. Isso é chamado de “*applet*”.

Os aplicativos e *applets* diferem quanto aos privilégios de execução que eles têm e também quanto ao modo que indicam onde iniciar a execução.

2.4.1 – Programação orientada a objetos

A programação orientada a objetos é um modelo poderoso na criação de programas que sobrevivem a mudanças inevitáveis, acompanhando o crescimento e o amadurecimento de qualquer sistema. Quando se entende a função de cada objeto e se tem interfaces bem definidas entre eles, pode-se desativar com segurança partes de um sistema mais antigo. Os mecanismos fundamentais da programação orientada a objetos são conhecidos como encapsulamento, herança e polimorfismo.

A programação orientada a objetos é um modo de conceitualizar um programa de computador como um conjunto de objetos que se interagem. Sua idéia central é organizar os programas de forma que eles espelhem o modo como os objetos são organizados no mundo real.

Para ser considerada como sendo “orientada a objetos”, uma linguagem precisa trazer suporte a um conjunto mínimo de três características [9][10][12]:

- Encapsulamento – Modulação é a capacidade de esconder detalhes de implementação. Deve-se pensar no encapsulamento como uma embalagem de proteção ao redor do código e dos dados que estão sendo manipulados. Essa embalagem define o comportamento e protege o código e os dados contra o acesso arbitrário de outro código. Na linguagem Java, a base do encapsulamento é a classe. Cria-se uma classe que representa uma abstração de um conjunto de objetos que compartilham da mesma estrutura e do comportamento definidos pela classe;
- Polimorfismo – Uma mesma mensagem enviada a diferentes objetos resulta em diferentes comportamentos dependendo da natureza do objeto que recebe a mensagem;
- Herança – Definição de classes e comportamentos baseados em classes já existentes para se conseguir uma reutilização de código segura.

2.4.2 – Embutindo programas Java em uma página web

Como os *applets* são executados a partir de um *browser* é a primeira coisa que deve ser entendida, porque os diversos métodos que os *applets* podem sobrepor derivam disso.

Os *browsers* lidam com HTML. Existem *tags* HTML que dizem “coloque esse texto em negrito”, “vá para um novo parágrafo” e “inclua essa imagem aqui”. Existe também uma *tag* HTML que diz “execute o *applet* Java encontrado neste arquivo **.class**”. Assim, como um arquivo de imagem(.gif ou .jpg, por exemplo) será exibido no lugar onde sua *tag* estiver no código fonte HTML, também um *applet* será executado quando sua *tag* for encontrada e os resultados serão exibidos naquele lugar na página web.

Um exemplo de código HTML que chama um *applet* é exibido na Fig. 2.1.

```

<title>
Um exemplo de applet
</title>
<applet code="teste.class" width=300 height=400>
</applet>

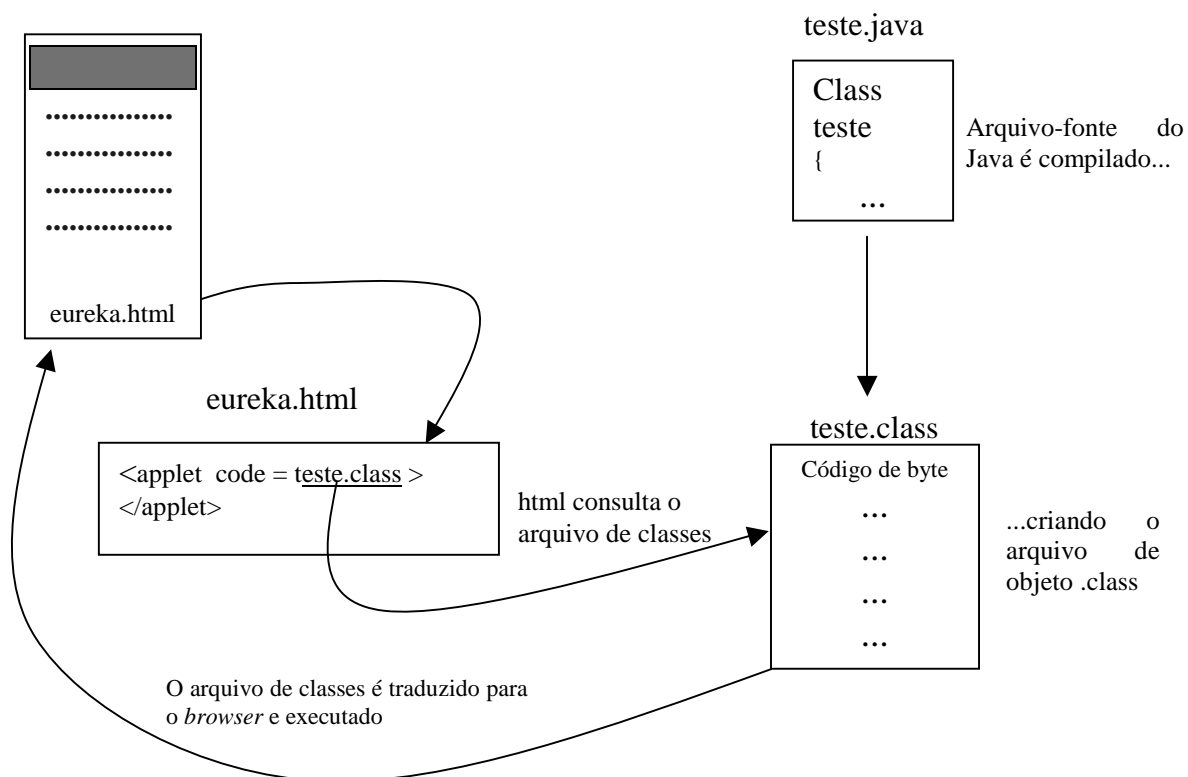
```

Fig. 2.1 -Exemplo de código HTML.

Os campos *width*(largura) e *height*(altura) são obrigatórios e são medidos em unidades de pixels(pontos de resolução no monitor do computador). Os *applets* são executados em um objeto de janela chamado painel e é necessário dar ao *browser* uma dica do tamanho do painel com o qual o *applet* deve começar.

Com o exemplo anterior colocado em um arquivo chamado **eureka.html**, a Fig. 2.2 ilustra como o *applet* **teste.class** será executado.

O *browser* visita o arquivo html

Fig. 2.2 – Execução de um arquivo **.html** contendo referência a um arquivo **.class**.

2.4.3 – Implementação do software

Neste trabalho, foram implementadas cinco classes: `Labwlt`, `Carta`, `Sistema`, `Grafico` e `Mat`. Estas classes se relacionam formando o ambiente de acordo com a estrutura do diagrama de classes modelado em UML [13] visto em Fig. 2.3.

O diagrama de classes de Fig. 2.3 foi estruturado com a utilização dos conceitos de agregação (representado pelo losango) e herança (representado pelo triângulo).

Nesse diagrama, pode ser visto que `Labwlt` agrega as demais classes do ambiente. Na agregação, um objeto complexo é composto por vários outros objetos mais simples. Nota-se também que a classe `Carta` é herdeira da classe `Canvas` do Java; `Sistema` e `Grafico` são herdeiras da classe `Panel`; e `Mat`, da classe `Object`.

Com `Carta`, objetiva-se uma ferramenta computacional didática capaz de reproduzir a carta de Smith; mostrando detalhadamente suas aplicações na solução de problemas clássicos de linhas de transmissão.

Já com `Sistema`, tem-se um modelo básico de linha de transmissão.

Em `Grafico` são mostrados gráficos da variação do módulo da tensão ao longo da linha em função de alguns parâmetros estabelecidos pelo usuário.

`Mat` é a classe onde se encontram todas as rotinas matemáticas para os cálculos envolvidos em linhas de transmissão.

Para que as rotinas gráficas das classes `Carta`, `Sistema` e `Grafico` possam ser exibidas num arquivo *web*, primeiro precisam ser adicionadas a uma classe do tipo *applet*. Em vista disso, a classe `Labwlt` agrega as classes `Carta`, `Sistema` e `Grafico`.

`Labwlt` é um *applet* que faz a interface com o usuário. É nesta classe que são obtidos os parâmetros a serem analisados no estudo de linhas de transmissão desenvolvido no ambiente.

Atente-se para o fato que `Carta`, `Sistema` e `Grafico` sozinhos não fazem nada. Estes componentes funcionam como um botão, um campo para a entrada de dados, ou qualquer outro componente de interface que se vê em *softwares*. É preciso uni-los e relacioná-los com eventos, como, por exemplo, o desenho da carta de Smith. Dessa forma, esse componentes podem ser manipulados de tal maneira

que juntos formem um ambiente no qual gráficos, equações, texto e etc. auxiliem ao aluno em seus estudos no que diz respeito à disciplina de eletromagnetismo, em especial o tópico de linhas de transmissão.

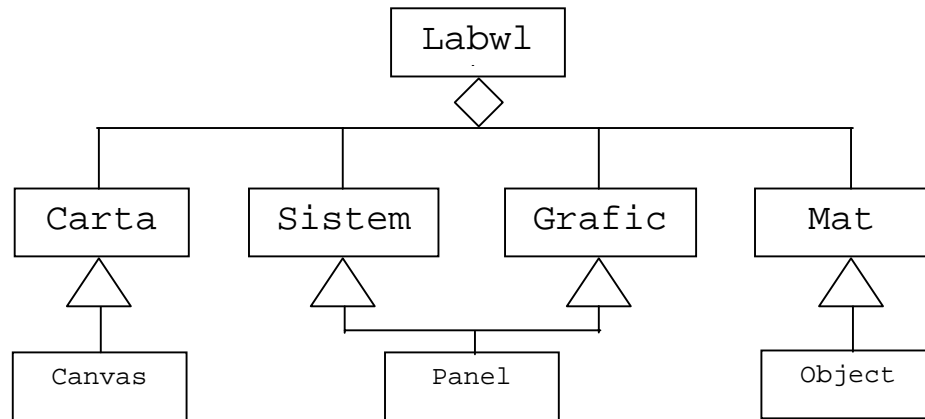


Fig. 2.3 – Diagrama de classes do ambiente em linguagem UML.