Desenvolvimento de Software Inteligente Para Treinamento de Operadores e Pessoal de Manutenção

V. Leonardo Paucar, UFMA

***Resumo* — Neste trabalho é desenvolvido um protótipo de software integrado por dois módulos, o primeiro módulo voltado para o treinamento de operadores de centros de controle, e o segundo como um software interativo de realidade virtual 3-D de auxílio ao pessoal dos centros de controle envolvido com a manutenção de equipamentos das subestações como os transformadores de extra alta tensão em 500 kV da ELETRONORTE. O objetivo principal do primeiro módulo foi desenvolver uma ferramenta que simule determinadas manobras, efetuadas pelos operadores, e suas conseqüências, de forma gráfica e intuitiva, no sistema elétrico de forma off-line evitando assim, o desligamento ou ligamento de equipamentos de forma inadequada no sistema elétrico real. O segundo módulo pretende contribuir com uma melhor capacitação dos operadores e pessoal dos centros de controle mediante a utilização de simuladores e tutores de treinamento com características de sistemas inteligentes para o treinamento em manutenção de autotransformadores. O protótipo foi projetado utilizando metodologia orientada a objetos e codificado na linguagem Java. A implementação da metodologia descrita foi testada com sucesso no sistema da ELETRONORTE.**

***Palavras-chave* — Simulador de manobras, manutenção de equipamentos, processador topológico, realidade virtual 3D, programação orientada a objetos.**

I. Introdução

O processo de desregulamentação do setor elétrico brasileiro vem trazendo consigo a reestruturação e a privatização dos seus sistemas elétricos de potência (SEP). Nesse processo, geração, transmissão e distribuição são consideradas setores separados. Uma das metas do processo de reestruturação do setor elétrico brasileiro é aumentar a participação de agentes privados de maneira a reduzir a participação do Governo nas funções empresariais do setor elétrico, possibilitando investimentos para a expansão do sistema. Nesse novo contexto, as atividades de planejamento e operação dos sistemas de potência devem ser realizadas considerando várias restrições técnicas e econômicas, nas quais qualidade, economia, segurança e confiabilidade dos SEP são aspectos de grande importância. [1]

A operação e o planejamento dos sistemas de potência são realizados nos centros de controle ou EMS (Energy Management System) [2]. Os centros de controle devem realizar avançadas funções em modo estudo e em tempo real, com a finalidade de garantir a operação adequada do sistema, tanto em operação normal como diante da ocorrência de perturbações. Os operadores dos centros de controle podem enfrentar situações complexas quando o sistema é submetido a perturbações. Assim, por exemplo, a tomada de decisões por parte dos operadores no tratamento de alarmes pode ser uma tarefa difícil que geralmente requer o apoio de programas computacionais baseados em inteligência artificial.

Com o advento dos mercados elétricos competitivos, os centros de controle estão sendo modernizados para lidar com a crescente competitividade no setor elétrico. Assim, por exemplo, o centro de controle do ONS (Operador Nacional do Sistema) vem sendo equipado para realizar tarefas como: funções SCADA, funções EMS, simulador de treinamento de despachantes, além de suporte ao planejamento, pré-operação e pós-operação.

As concessionárias de eletricidade, como é o caso da ELETRONORTE, também vêm atualizando seus centros de controle. A modernização dos centros de controle em termos de hardware, software, sistemas de comunicação e equipamento computacional de um modo geral, significa um grande desafio tanto pela tecnologia do estado da arte a adotar quanto pelo alto investimento requerido.

Para garantir uma operação adequada do sistema de potência, um centro de controle precisa de um funcionamento confiável, eficiente e rápido. Para isso é necessário que os operadores estejam capacitados e disponham das ferramentas computacionais correspondentes. Uma estratégia para uma melhor capacitação dos operadores têm sido, por exemplo, os cursos de capacitação e treinamento de operadores mediante o uso de simuladores de centros de controle e de sistemas de potência. Em alguns casos, esses simuladores podem ser considerados inteligentes por possuírem características típicas de áreas da inteligência artificial como os sistemas especialistas e as redes neurais.

Diante desse panorama, os operadores e o pessoal dos centros de controle precisam de um software de treinamento para adequar-se aos novos sistemas EMS/SCADA, principalmente no que diz respeito a tratamento de alarmes, funções básicas EMS e manutenção de equipamentos de subestações, tais como os transformadores, entre outros. Esse software deve possuir características de sistemas inteligentes com interatividade gráfica 3-D e a capacidade de rodar em ambiente de computação distribuída, além das características de portabilidade, interoperabilidade, escalabilidade, etc. [3]

O envolvimento do grupo de pesquisa DEE-UFMA com

Este trabalho foi apoiado pelas Centrais Elétricas do Norte do Brasil (ELETRONORTE).

V. L. Paucar trabalha na Universidade Federal do Maranhão (e-mail: Lpaucar@ieee.org).

programação orientada a objetos, inteligência artificial, sistemas multiagentes, computação distribuída, software interativo gráfico, entre outros, também motivou este projeto. [4]

II. Paradigma de Orientação a Objetos (POO)

O ambiente competitivo do mundo dos negócios está mudando constantemente, a um passo acelerado, levando as empresas a se adaptarem a ele se quiserem continuar com sua posição de destaque no mercado.

Durante os anos 60 isso já era uma realidade, pois muitos esforços foram aplicados no desenvolvimento de software de grande porte, mas os resultados não foram bem os esperados, pois as empresas acabaram enfrentando sérias dificuldades, tais como: os cronogramas de software estavam sempre atrasados, os custos excediam em muito os orçamentos e os produtos finais não eram confiáveis. Então, se começou a perceber que o desenvolvimento de software era uma atividade muito mais complexa do que se havia imaginado e que os modelos de programação, da época, já não auxiliavam os desenvolvedores a atender a demanda de desenvolvimento de software de grande complexidade em um curto intervalo de tempo com alta confiabilidade e baixo custo.

O POO surgiu como uma solução para suprir tais problemas, pois traz as seguintes vantagens [5][6]:

- Fácil reaproveitamento de código – Reutilizar um código já testado e sem erros traz, ao desenvolvedor um ganho de tempo fenomenal, pois não tem que passar por todos os problemas encontrados durante o desenvolvimento do mesmo.
- Facilidade na manutenção do software – O POO trabalha com módulos, ou seja, o software é dividido em várias partes para um melhor reaproveitamento de cada módulo e fácil manutenção tanto no sentido de estender, como no sentido de corrigir erros.
- Menor código gerado – Os softwares desenvolvidos em POO, podem gerar códigos fontes menores, aplicando alguns conceitos de POO, como herança e polimorfismo.
- Fácil representação do mundo real – "Um aspecto interessante da teoria dos objetos é a sua característica intrínseca de analisar o mundo como ele é, permitindo organizar resultados de maneira mais fácil e natural", ou seja, O POO procura representar o mundo real como ele realmente se apresenta, pois o mundo é formado de objetos.

O modelo orientado a objetos apresenta os seguintes elementos:

- Abstração – Denota as características essenciais de um objeto que o diferencia dos outros e assim provê limites conceituais bem definidos, dependendo da perspectiva do visualizador, ou seja, descreve as características de um objeto que o torna único isso a partir da visão da pessoa que esta modelando outra pessoa, por exemplo, poderia ter uma visão diferente sobre o mesmo objeto. A abstração é uma das principais formas que os humanos usam para lidar com a complexidade.
- Encapsulamento – é o processo de compartimentalizar os elementos de uma abstração que constitui sua estrutura e o seu comportamento. Encapsulamento serve para separar a interface da abstração de sua implementação. Os conceitos de abstração e encapsulamento são conceitos complementares, sendo que a abstração foca o comportamento de um objeto e o encapsulamento foca a implementação que vai realizar tal comportamento.
- Modularidade – é a propriedade de um sistema de ser decomposto em um conjunto de módulos coesivos e fracamente acoplados. A idéia principal de modularizar um sistema é dividir o problema em pequenas partes que tenham conexões entre sim para melhor compreensão e resolução do problema.
- Hierarquia – é uma forma de classificação ou ordenamento das abstrações. Fazer hierarquias de abstrações simplifica e ajuda a entender o problema.

Todo sistema que for chamado de sistema completamente orientado a objetos deve possuir no mínimo os três primeiros elementos citados acima: É importante mencionar os seguintes conceitos, já que são necessários para um entendimento do POO:

- Objeto – um objeto tem estado, comportamento e identidade, sendo que as estruturas e os comportamentos de objetos similares são definidos em classes comuns. Os termos instância e objetos têm o mesmo significado.
- Estado – é uma condição durante a vida de um objeto, ou seja, é como os seus atributos ou propriedades estão durante um determinado momento do seu ciclo de vida. O estado de um objeto representa os resultados acumulativos de seu comportamento.
- Comportamento – é como um objeto atua ou reage, quando ocorre a passagem de mensagem e mudança de seu estado.
- Identidade – é o atributo ou propriedade que o diferencia de todos os outros objetos.
- Classe – é um conjunto de objetos que compartilham uma estrutura e um comportamento comum.
- Interface de uma classe – consiste na declaração de todos os métodos que a classe conterá, ou seja, define a abstração dos objetos desta classe.
- Implementação – Consiste na implementação de todos os métodos contidos na interface da classe, ou seja, define o comportamento dos objetos desta classe.
- Herança – define um relacionamento entre classes onde uma classe compartilha a estrutura ou o comportamento definidos em uma ou mais classes. Pode ser classificada em:
    - Simples – é quando uma classe compartilha a estrutura ou o comportamento a partir de outra classe, sendo que a classe que compartilha sua estrutura é denominada super-classe e a que herda a estrutura de sub-classe, ou seja, cada sub-classe possui apenas uma super-classe.
    - Múltipla – é quando uma classe compartilha a estrutura ou o comportamento a partir de duas ou mais classes, sendo que a classe que compartilha sua estrutura é denominada super-classe e a que herda a estrutura de sub-classe, ou seja, cada sub-classe possui

mais de uma super-classe.

A. *Unified Modeling Language (uml)*

"É uma linguagem de modelagem visual que é usada para especificar, visualizar, construir e documentar os artefatos de software.", ou seja, a UML funciona como um padrão para documentação de software desenvolvidos com a tecnologia de objetos.

Esse tipo de abordagem é importante, pois evita que o desenvolvedor do software seja o único a conhecer como o mesmo funciona evitando que ele seja a única pessoa que possa manipulá-lo. A UML é uma forma de representação visual que tem o propósito de modelar, configurar, manter e controlar informações sobre um software, possibilitando que qualquer pessoa possa compreender e manipulá-lo.

Alguns dos principais objetivos da UML:

- Fornecer aos usuários uma linguagem de modelagem visual expressiva;
- Ser independente de linguagens de programação e processos de desenvolvimento;
- Prover uma base formal para entender a linguagem modelada.

B. *Diagrama de Classes*

O diagrama de classes é o principal diagrama dentro da modelagem orientada a objetos. Dentro deste diagrama é possível modelar todas as classes e seus respectivos relacionamentos. Também podem ser modelados nesse diagrama interfaces e pacotes.

Os principais elementos que constituem um diagrama de classes são suas classes e seus relacionamentos.

- Classes – O conceito de classe já foi abordado na Seção 2.1.2 deste capítulo. Sua representação dentro dos padrões da UML é: um retângulo subdividido em três compartimentos, separados por linhas horizontais que nessa ordem armazenam: o nome da classe, a lista de atributos da classe e a lista de operações desta classe, Figura 1.

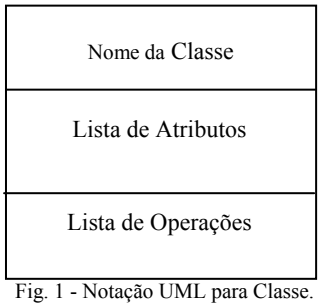

Fig. 1 - Notação UML para Classe.

- Relacionamentos – As classes dentro do contexto de um software geralmente não trabalham isoladamente, ou seja, as classes colaboram entre si por meio de relacionamentos. Os principais relacionamentos dentro de um diagrama de classes são:
  - Associação – É um relacionamento que interliga duas ou mais classes, demonstrando a colaboração entre as instâncias (objetos) da classe. É um dos relacionamentos mais comuns e sua representação é uma linha sólida entre as duas classes.
  - Generalização – É um relacionamento entre um elemento mais geral e outro mais específico, sendo que o elemento mais específico deve possuir todas características do elemento mais geral, além de suas próprias. Sua representação é uma seta fechada e vazada, como seu corpo sólido, que parte da classe mais específica para a mais geral.
  - Dependência – É o relacionamento que indica que a mudança na interface de uma classe poderá gerar mudança na outra. Sua representação é uma linha tracejada que parte da classe dependente para a outra classe terminando com uma seta aberta.
  - Agregação – É um caso particular da associação. A agregação indica um relacionamento "todo-parte", ou seja, indica que um objeto todo é representado por outros objetos que são sua parte.
  - Composição – É uma variação mais poderosa da agregação. A diferença é que na composição a classe parte pertence apenas a sua classe todo, ou seja, a classe todo não pode existir sem sua parte, já na agregação isso pode acontecer.

C. *JAVA*

Java é um ambiente de programação completamente orientado a objetos que pode ser utilizado para o desenvolvimento de excelentes aplicações comerciais ou para ser usada em universidades como objeto de estudo. Java pode ser vista como uma fusão de várias tecnologias que vêm sendo desenvolvidas na área de computação, de modo que estudantes dessa linguagem tem a oportunidade de tomar contato com vários tópicos recentes: programação concorrente, sistemas distribuídos, orientação a objetos, protocolos da internet, e uma série de outros assuntos fáceis de praticar nessa linguagem. [7]

D. *Características da Linguagem Java*

- Compilada: Um programa em Java é compilado para o chamado "byte-code", que é próximo as instruções de máquina, mas não de uma máquina real. O "byte-code" é um código de uma máquina virtual idealizada pelos criadores da linguagem. Por isso Java pode ser mais rápida do que se fosse simplesmente interpretada.
- Portável: Java foi criada para ser portável. O "byte-code" gerado pelo compilador para a sua aplicação específica pode ser transportado entre plataformas distintas que suportam Java (Solaris 2.3®, Windows-NT®, Windows-95®, Mac/Os etc). Não é necessário recompilar um programa, para que ele rode em uma máquina ou sistema operacional diferente, ao contrário do que acontece, por exemplo, com programas escritos em C e outras linguagens. Esta portabilidade é importante para a criação de aplicações para a heterogênea internet. Em Java um inteiro, por exemplo, tem sempre 32 bits, independentemente da arquitetura. O próprio compilador Java é escrito em Java, de modo que ele é portável para qualquer sistema que possua o interpretador de "byte-codes".
- Orientada a Objetos: A portabilidade é uma das características que se inclui nos objetivos almejados por uma

linguagem orientada a objetos. Em Java ela foi obtida de maneira inovadora com relação ao grupo atual de linguagens orientadas a objetos. Java suporta herança, mas não herança múltipla. A ausência de herança múltipla pode ser compensada pelo uso de herança e interfaces, onde uma classe herda o comportamento de sua superclasse além de oferecer uma implementação para uma ou mais interfaces. Java permite a criação de classes abstratas. Outra característica importante em linguagens orientadas a objetos é a segurança. Dada a sua importância o tópico foi escrito à parte.

- Segura: A presença de coleta automática de lixo, evita erros comuns que os programadores cometem quando são obrigados a gerenciar diretamente a memória (C , C++, Pascal). A eliminação do uso de ponteiros, em favor do uso de vetores, objetos e outras estruturas substitutivas traz benefícios em termos de segurança. O programador é proibido de obter acesso à memória que não pertence ao seu programa, além de não ter chance de cometer erros comuns como, por exemplo, uso indevido de aritmética de ponteiros. Estas medidas são particularmente úteis quando pensarmos em aplicações comerciais desenvolvidas para a internet. A presença de mecanismos de tratamento de exceções torna as aplicações mais robustas, não permitindo que elas abortem, mesmo quando rodando sob condições anormais.
- Suporta concorrência: A linguagem permite a criação de maneira fácil, de vários "threads" de execução. Este tópico será útil quando você estudar animações, e é particularmente poderoso nos ambientes em que aplicações Java são suportadas, ambientes estes que geralmente podem mapear os threads da linguagem em processamento paralelo real.
- Eficiente: Como Java foi criada para ser usada em computadores pequenos, ela exige pouco espaço, pouca memória. Java é muito mais eficiente que grande parte das linguagens de "scripting" existentes, embora seja cerca de 20 vezes mais lenta que C, o que não é um marco definitivo. Com a evolução da linguagem, serão criados geradores de "byte-codes" cada vez mais otimizados que trarão as marcas de performance da linguagem mais próximas das de C++ e C. Além disso, um dia Java permitirá a possibilidade de gerar código executável de uma particular arquitetura "on the fly", tudo a partir do "byte-code".

Suporte para programação de sistemas distribuídos: Java fornece facilidades para programação com: Sockets, RMI - Remote Method Invocation, TCP-IP, etc.

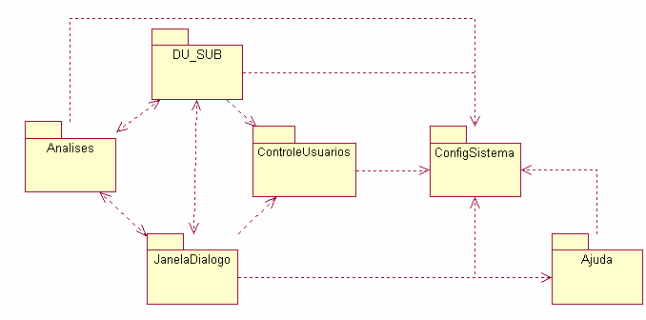

Fig. 2 - Arquitetura do Sistema

Fig. 3 - Pacote "Analises"

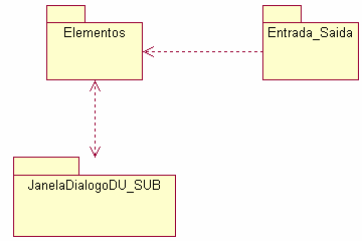

Fig. 4 - Pacote "DU_SUB"

III. Análise do Sistema

A análise do sistema foi feita usando o paradigma de orientação a objetos, sendo documentado através de uma linguagem de modelagem denominada UML – Unified Modeling Language.

A. Arquitetura do Sistema

Os pacotes descritos abaixo constituem a arquitetura do sistema que foi desenvolvido.

B. Pacote "Analises".

Este pacote é responsável por agrupar as classes ou outros pacotes que tenham características de análises elétricas que possam ser aplicadas usando os dados disponíveis pela aplicação. Dentro deste pacote existem outros dois que são Simulação e Processador de Topologia. O primeiro contém as classes responsáveis por gerenciar a avaliação de um operador e o segundo contém as classes que gerenciam a análise de rede.

C. Pacote "DU_SUB"

Este é o pacote mais importante da arquitetura, pois nele são modeladas as classes que contém as informações necessárias para modelar um SEP. Ele é dividido em outros três pacotes que serão detalhados em nível de classes: Elementos, Entrada_Saida e JanelaDialogoDU_SUB.

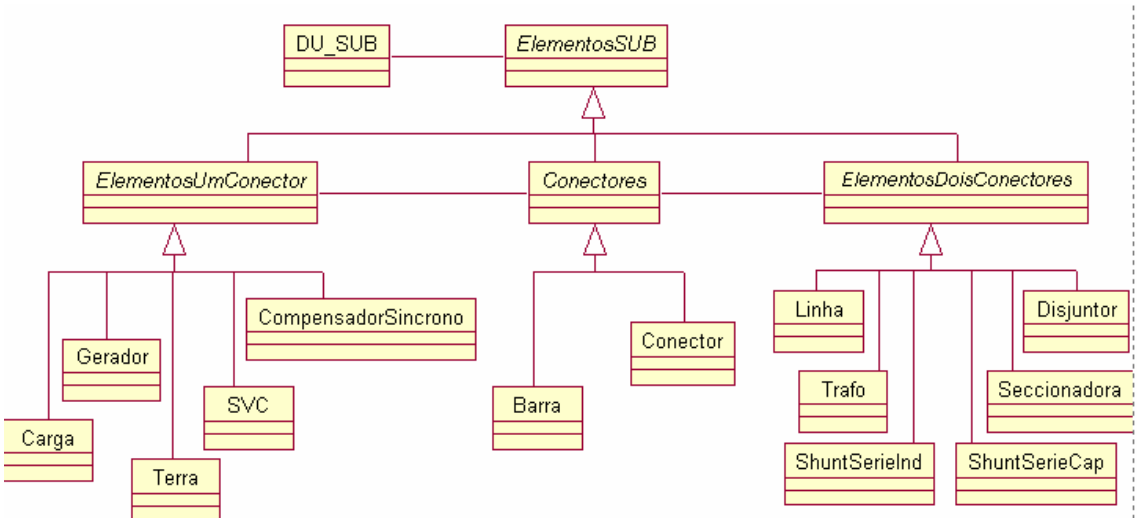

Fig. 5 - Pacote "ELEMENTOS"

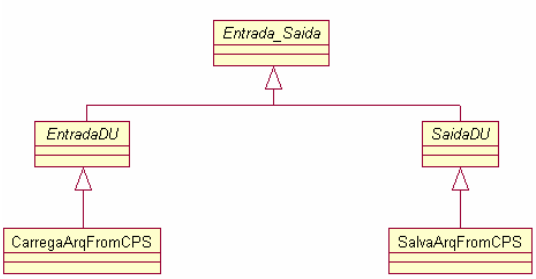

Fig. 6 - Pacote "Entrada_Saida"

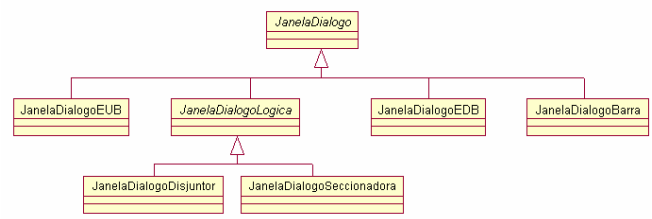

Fig. 7 - Pacote "JanelaDialogoDU_SUB"

D. Pacote "Elementos"

Este pacote é responsável pelo agrupamento dos elementos que constituem o Sistema Elétrico de Potência, ou seja, neste pacote estão considerados representações lógicas de elementos que constituem ou formam um SEP. Para a modelagem deste sistema computacional foram considerados os seguintes elementos: Barra, Conector, Gerador, Compensador Síncrono, Carga, Terra, SVC, Linha, Transformador, Shunt Série Capacitivo, Shunt Série Indutivo, Seccionadora e Disjuntor, sendo que outros elementos podem ser incluídos facilmente neste pacote.

E. Pacote "Entrada_Saida"

Este pacote é responsável por garantir a persistência dos dados de um SEP, ou seja, este pacote é responsável por armazenar em disco um sistema elétrico modelado e por recuperar tais informações.

F. Pacote "JanelaDialogoDU_SUB"

Este pacote tem por finalidade disponibilizar janelas de diálogos para a entrada dos dados pertinentes ao elementos elétricos modelados no pacote Elementos.

IV. Simulador de Manobras

O simulador de manobras tem como objetivo identificar os elementos da rede que se encontram energizados, desenergizados e com perda de função, levando em conta o estado de operação atual dos componentes do sistema.

O simulador de manobras realiza um processamento topológico e determina ilha elétrica a partir da conectividade da rede, sendo que a mesma pode estar: energizada quando nela encontram-se geradores e demandas, desenergizada quando na ilha encontra-se somente demanda e com perda de função quando na ilha encontram-se só geradores. [8]

De forma pré-definida os componentes energizados serão visualizados pela cor vermelha, os componentes desenergizados serão visualizados pela cor verde e os componentes com perda de função serão visualizados com a cor azul. Na Figura 8 é mostrada a S.E. Presidente Dutra como um exemplo da visualização dos componentes energizados, desenergizados e com perda de função. A Figura 8 pode ser dividida em quatro partes: Barra de Título, Barra de Desenho, Área de Desenho e Barra de Status.

Barra de Título

Esta parte da janela é responsável por indicar qual o sistema elétrico esta aberto para edição ou manobra, além de ter as funções para minimizar, maximizar e fechar.

Barra de Desenho

Esta parte da janela é responsável por disponibilizar representações geométricas dos elementos reais de um sistema elétrico de potência, sendo eles, barra, conector, gerador, compensador síncrono, carga, linha, transformador, terra, SVC, shunt em série capacitivo, shunt em série indutivo, seccionadora e disjuntor , como também opções para manipulação dos mesmos, como por exemplo, selecionar, mover, apagar, etc.

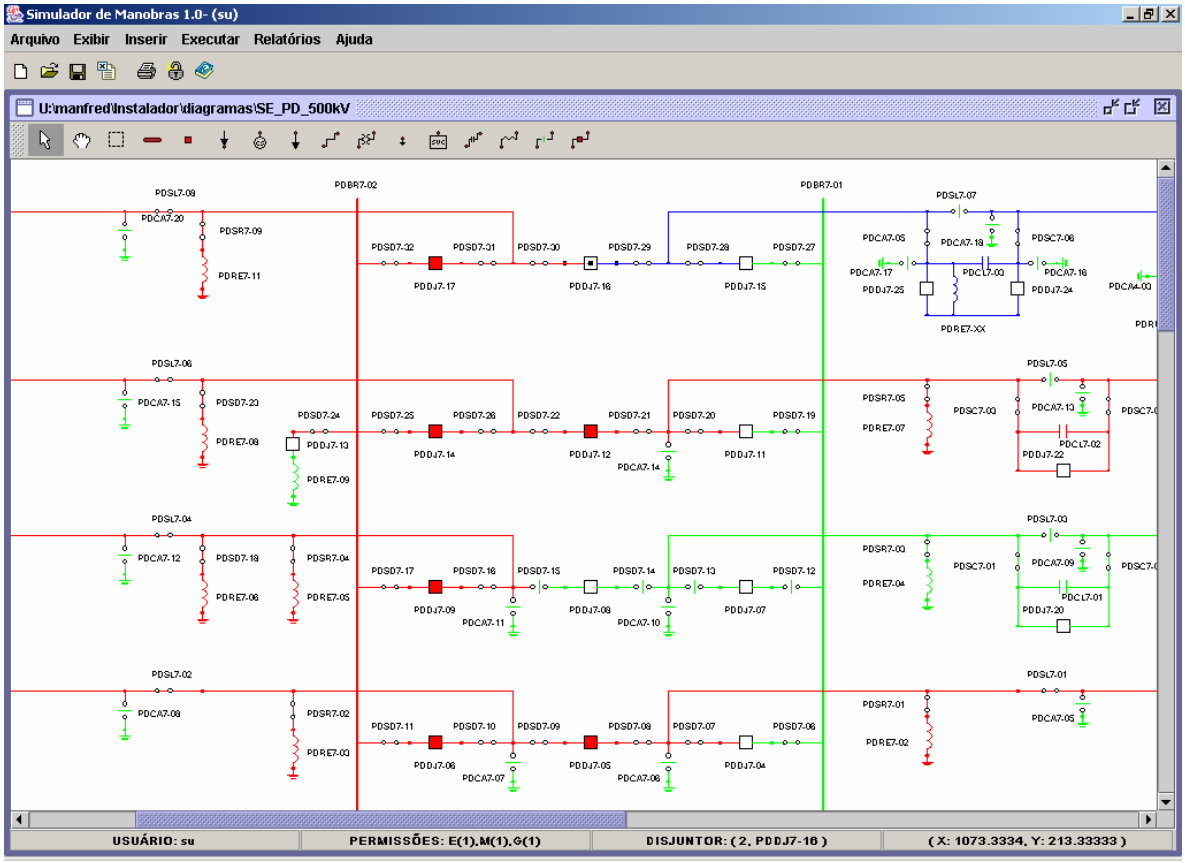

Fig. 8 – Janela de visualização dos componentes energizados, desenergizados e com perda de função.

Área de Desenho

Nesta parte da janela pode ser modelado um sistema de potência onde os elementos da barra de desenho podem ser utilizados. Todas as manipulações dentro de um sistema elétrico modelado são feitas por intermédio desta área, como por exemplo, à edição dos dados de um elemento elétrico é feita através de uma caixa de diálogo que é exibida ao se efetuar o duplo clique no elemento que o representa. Essa janela possui uma área considerável para se modelar sistemas elétricos, por isso a mesma é provida de barras de rolagem tanto na horizontal como na vertical.

Barra de Status

A Barra de Status tem como objetivo disponibilizar algumas informações, respectivamente, o usuário logado no sistema, as permissões deste usuário, elemento que está conectado e as coordenadas da posição em que o cursor se encontra na tela.

A. Resumo de manobras

O resumo de manobras é uma janela de ajuda ao operador onde é mostrado de forma resumida os componentes do sistema que mudarem seu estado de operação depois de ser feita uma manobra. Cores e sons também foram incluídos de forma a melhorar o desempenho na elaboração da seqüência de manobras.

Um conjunto de opções foram consideradas no resumo de manobras e podem ser visualizadas fazendo um clique no botão direito do mouse. A opção de Salvar salva a seqüência de manobras em um formato *.xml para o visualizador do relatórios. A opção Voltar retrocede um passo na seqüência de manobras eliminando a última manobra no Resumo de Manobras e na Janela de visualização dos componentes. A opção de Adicionar Comentários adiciona no relatório de manobras um comentário e é armazenado para uma possível utilização futura.

B. Visualizador de manobras

A visualização da seqüência de manobras se dá através de um relatório que é gerado por meio de um arquivo XML, onde este arquivo é carregado através de uma janela de diálogo ou diretamente do sistema no momento de salvar uma nova seqüência de manobras. Depois que o arquivo é carregado outra janela de diálogo será apresentada, onde a mesma é dividida em quatro partes: Título, Cabeçalho, Dados e Botões de ações..

A função dessa janela é disponibilizar ao usuário uma forma simples de entrar com os dados necessários para a elaboração de um relatório de seqüência de manobras. Como citado anteriormente, esta janela é dividida em quatro partes: Título, Cabeçalho, Dados e Botões de ações.

- Título – Compõem os dados necessários para a construção do título do relatório.

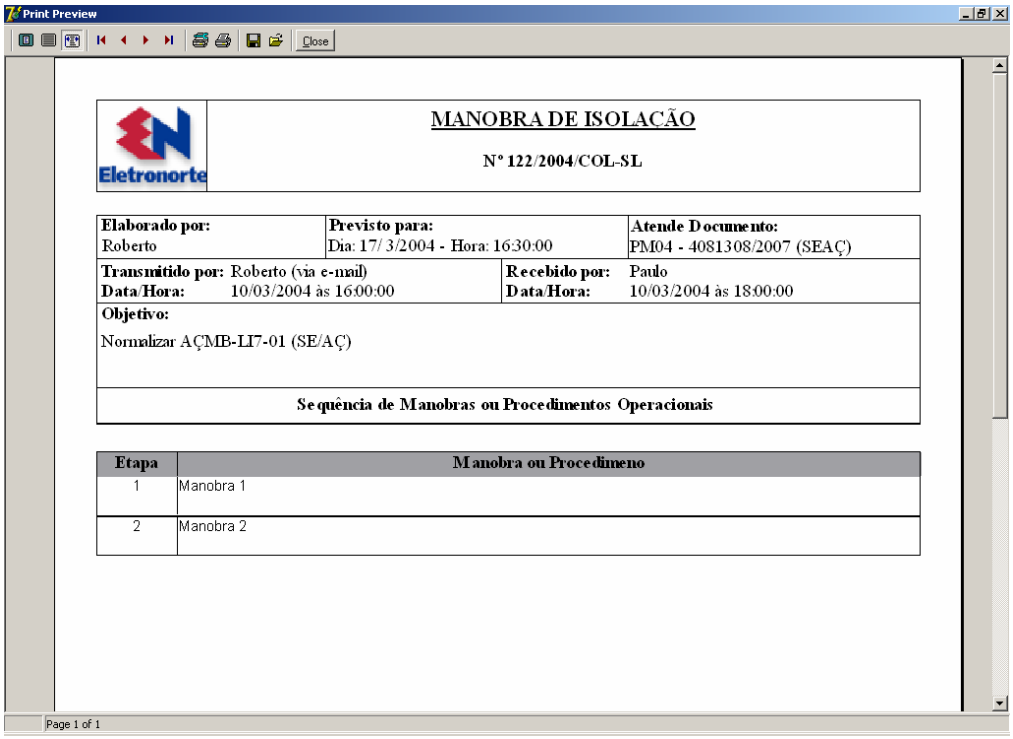

Figura 9 – Visualização de relatório de seqüência de manobras.

- Cabeçalho – Compõem os dados necessário para a construção do cabeçalho do relatório.
- Dados – Compõem a seqüência de manobras que será exibida no relatório. Esta parte da janela de diálogo se torna interessante, pois o usuário pode manipular as manobras que foram carregadas através da barra de ferramenta que é disponibilizada nesta parte da janela. Esta barra de ferramenta é composta das seguintes opções:
  - Novo – Insere uma nova manobra na lista;
  - Editar – Edita uma manobra existente na lista;
  - Salvar – Salva uma manobra que foi inserida ou editada;
  - Cancelar – Cancela uma ação de inserção ou edição;
  - Excluir – Exclui uma manobra existente;
  - Primeiro – Vai para a primeira manobra da lista;
  - Anterior – Volta para uma manobra antes da atual;
  - Próximo – Vai para a manobra seguinte;
- Último – Vai para a última manobra.
- Botões de ação – Estes botões tem como objetivo executar alguma ação pré-definida, entre elas pode citar:
  - Visualizar o relatório – Onde o responsável por tal ação é o botão Visualizar. O resultado da ação deste botão pode ser visualizado na Figura 9;
  - Finalizar a janela – Onde o responsável por tal ação é o botão Cancelar;
  - Importar uma nova lista de manobras – Onde o responsável é o botão Importar.

C. Gerenciamento de Usuários e Seus Privilégios

O gerenciamento de usuários é feito através de uma janela de dialogo. Esta janela é composta de duas partes: uma lista de usuários que contém todos os usuários cadastrados no sistema e uma barra de opções onde ficam as funcionalidades para o gerenciamento de usuários, tais como: inserir, editar e excluir.

1) Inserindo um Usuário

Esta opção tem como intuito adicionar um novo usuário no sistema, sendo que para executá-la o usuário deve clicar na opção Novo

- A janela para inserção de usuários é dividida em três partes: dados dos usuários, privilégios dos usuários e botões de ação.
- Dados dos Usuários: Possui os seguintes campos que devem ser preenchidos: Login – Representa o nome que o usuário vai usar para se identificar no momento da autenticação; Senha – Representa a senha que o usuário vai utilizar para se identificar no sistema; Confirmação de senha – É uma medida de segurança, para que o usuário tenha certeza da senha que foi digitada.
- Privilégios dos Usuários: Possui os seguintes privilégios que devem ser atribuídos ao usuário que vai ser inserido: Edição – O usuário que possui apenas este privilégio poderá criar, abrir e editar um modelo de diagrama unifilar, mas não poderá visualizar relatórios nem executar manobras; Manobra – O usuário que possui apenas este privilégio poderá apenas abrir um diagrama unifilar, fazer manobras, visualizar relatórios, executar a funcionalidade análise de rede e avaliação de manobras, mas não poderá criar ou editar um diagrama unifilar; Gerenciar de Usuários – O usuário que possuir apenas este privilégio poderá apenas gerenciar usuários; Super Usuário – Um super usuário será aquele que possui todos os outros privilégios citados anteriormente; Outros privilégios – Poderão ser feitas permutações das permissões citadas para criação de uma nova.

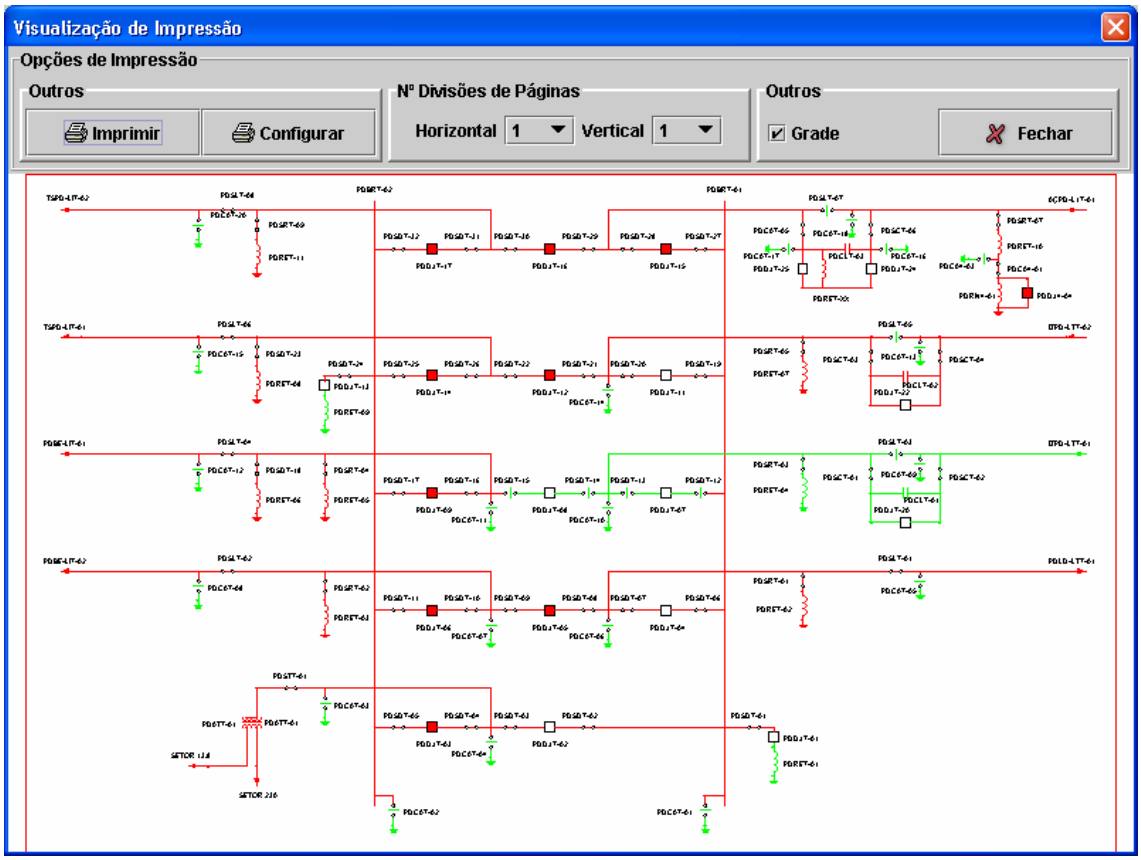

Figura 10 – Janela de pré-visualização de impressão.

2) Editando um Usuário

Esta opção tem como intuito editar um usuário que já existe no sistema, sendo que para executá-la o usuário deve selecionar o usuário que deseja editar na lista de usuários e depois clicar na opção Editar. Esta janela para edição de usuário tem a mesma estrutura e funciona de forma semelhante à janela de inserção de usuário.

3) Excluíndo um Usuário

Esta opção tem como intuito excluir um usuário que já exista no sistema, sendo que para executá-la o usuário deve selecionar o usuário que deseja excluir na lista de usuários e depois clicar na opção Excluir. Depois de efetuado o clique será exibida uma janela de confirmação, para que se tenha certeza se a exclusão é realmente desejada. Caso a resposta da confirmação seja positiva o usuário será excluído do sistema e será exibida uma informação de sucesso.

D. Imprimir

Para impressão o usuário deve selecionar a área que deseja imprimir através da opção cursor da barra de desenho, depois deve clicar na opção imprimir, onde será exibida uma janela de pré-visualização como pode ser visualizado na Figura 10.

Através desta janela pode-se configurar a impressora, o número de páginas em que a área pode ser impressa com o máximo de nove páginas, três na horizontal e três na vertical, além de se poder selecionar a grade, ou seja, o contorno no desenho

E. Avaliação de Manobras

A avaliação de manobras é feita praticamente em duas fases. A primeira fase tem o intuito de carregar um arquivo com as manobras que o operador deve simular, já a segunda fase tem o objetivo de iniciar avaliação. A seguir são detalhados os passos que devem ser realizados para executar cada fase do processo de avaliação:

1) Primeira Fase

Como já mencionado anteriormente, o objetivo desta fase é carregar um arquivo que possua uma seqüência de manobras corretas, para que o operador seja avaliado. A primeira coisa que deve ser feita para carregar o arquivo é clicar na opção de menu Executar – Avaliação de Manobras – Carregar, onde será exibida a janela de diálogo para escolha do arquivo de manobras a ser carregado. Depois que a carga do arquivo é realizada o sistema envia uma mensagem de confirmação de carregamento, caso o processo tenha sido realizado com sucesso. Com base nesse arquivo que será feita a avaliação do operador.

2) Segunda Fase

A segunda fase consiste na avaliação do usuário, onde a mesma só pode ser realizada depois do carregamento do arquivo de seqüência de manobras, referido no item da primeira fase desta seção.

Para realizar a avaliação deve-se clicar na opção de menu Executar – Avaliação de Manobras – Avaliar, onde será exibida uma janela de diálogo para se entrar com o nome do operador que será avaliado.

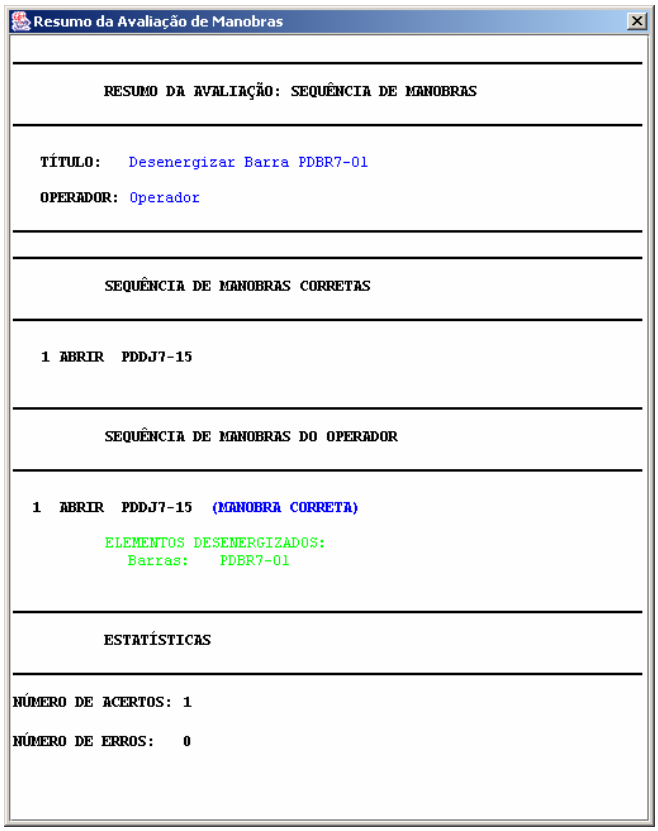

Figura 11 – Janela de resultado da avaliação de manobras.

Depois será exibida uma janela, onde todas as manobras feitas durante o processo de avaliação poderão ser visualizadas.

Para encerrar a avaliação basta o operador clicar com o botão direito do mouse na janela de visualização de manobras que será apresentada uma janela que possui a opção de encerrá-la.

Quando o operador clica na opção "encerrar avaliação" será exibida uma janela de confirmação para averiguar se o operador realmente deseja encerrar a mesma. Caso a resposta seja negativa o mesmo pode continuar as manobras, normalmente, caso a resposta seja positiva a avaliação será dada como encerrada e automaticamente será exibida uma janela com os resultados da avaliação do operador, como pode ser visto na Figura 11. Esta janela descreverá o título da avaliação, o nome do operador que foi avaliado, a seqüência de manobras corretas, a seqüência de manobras realizadas pelo operador e o número de acertos e erros.

V. Manutenção de Autotransformadores

O propósito desta aplicação é criar um sistema para treinamento de operadores e do pessoal de manutenção, na prevenção e solução de falhas no equipamento transformador. O software deve ser capaz de:

Simular o funcionamento de um autotransformador utilizando realidade virtual não-imersiva;

Simular uma falha quando o usuário selecionar uma das disponíveis no sistema;

Carregar informações dos componentes que formam o autotransformador para que o usuário possa ver as propriedades dos mesmos;

Gerar uma falha aleatoriamente quando for o usuário for simular uma falha;

Modificar o estado da cena pra simular a falha.

Além disso, o software deve apresentar as seguintes características:

Janelas e caixas de diálogo baseadas na metáfora de formulários;

Maximizar facilidade de navegação via teclado ao invés de via mouse.

A. Levantamento de requisitos

Os requisitos de software foram obtidos através de entrevistas com operadores e o pessoal de manutenção da ELETRONORTE-MA e visitas ao campo de atuação deste pessoal. Estas visitas incluíram o reconhecimento das técnicas de que os operadores e pessoal da manutenção utilizam para solucionar possíveis falhas nos autotransformadores existentes no campo de trabalho.

A principal funcionalidade que o sistema deve apresentar é capacidade de simular falhas, escolhidas aleatoriamente pela aplicação. O autotransformador possui um esquema de falhas, organizadas por cada sistema (resfriamento, proteção, etc). Cada sistema possui seu componentes. Por sua vez, cada componente do sistema possui modos de falha que agrupam as falhas existentes nele. Quando é gerada uma falha em um componente, outros componentes podem ser afetados. As informações presentes na Tabela 1, foram utilizadas para a construção do banco de dados.

O processo de geração de falhas acontece da seguinte forma:

- O Operador de Manutenção informa o sistema que irá fazer uma simulação;
- O sistema gera uma falha aleatória, dentre as existentes;
- Cada falha gera danos em componentes conforme a Tabela 1. O sistema deve modificar o atributo "status", na tabela de equipamento do banco de dados, de todos os componentes relacionados com a falha gerada para "DESLIGADO";
- O sistema então mostra ao usuário um alarme informando o modo de falha (mas não a falha). Isto serve para que o operador de manutenção tenha uma idéia de qual foi a falha gerada, diminuindo o esforço para descobrila.

Estes passos constituem o cenário principal do caso de uso "Gerar Falha".

B. Grafo de cena

A representação do mundo virtual foi estruturada de acordo com Figura 12. De acordo com este grafo, o nó "cena" representa o nó raiz, contendo todos os objetos que irão compor o mundo virtual. Dentre os objetos que irão compor a cena estão:

- O Autotransformador, representado pelo nó interno "Autotransformador";
- A Subestação, ambiente onde está localizado o Autotransformador, é representada pelo nó interno "Subestação".

Tabela 1: Esquema de falhas de um autotransformador.

Sistema	Componentes	Modo de Falha	Falhas	Componentes Afetados
Resfriamento	Motobombas	Falta de fluxo de óleo	Válvulas fechadas entre a motobomba e o radiador	Válvulas
			Palheta do fluxômetro com avaria	Fluxômetro
			Curto-Circuito elétrico na motobomba	Motobomba MB1 e MB2
		Redução da capacidade de transformação	Contato do termômetro com defeito	Contatos do termômetro
			Falta das fontes de alimentação A e B	Disjuntor termomagnético; Transformador de serviço; réguas bornes;
	Motoventiladores	Aquecimento excessivo do óleo	Quebra da palheta do ventilador	Palheta
			Quebra do eixo do ventilador	Motoventilador
			Alto nível de ruído do ventilador	Motoventilador
			Falta de fase na alimentação da força	Relé de supervisão trifásico e circuito
Proteção	Relé de gás	Bóia desregulada	Desalinhamento da base de sustentação das bóias	Bóia e base de sustentação das bóias.
		Bóia furada	Ruptura na solda de junção da bóia	Bóia

O Autotransformador ou simplesmente autotrafo é um equipamento que é composto de vários subcomponentes. Para a representação destes objetos, foram criados vários outros nós internos, com objetivo de facilitar a manipulação deles dentro da cena. Representando a base central do autotransformador está o nó "Base". Ainda têm-se os nós:

- "Motoventiladores", agrupando todos os motoventiladores do grupo 1 (constituem os motoventiladores do lado esquerdo do autotrafo) e do grupo2 (motoventiladores do lado direito);
- "Radiadores", agrupando os radiadores do lado esquerdo (nó "radiadores do grupo 1") e do lado direito (nó "radiadores do grupo 2");
- "Motobombas", contendo as motobombas;
- Representar todos os objetos do autotrafo em um grafo de cena, seria uma tarefa bastante difícil, pois a quantidade de objetos existentes é muito grande. O grafo da Figura 12, só dá uma pequena idéia de como a cena deve ser representada. Poderiam ser adicionados mais nós, como as hélices de cada motoventilador, mas isto tornaria a leitura do grafo muito complexa.

C. Apresentação do ambiente virtual tridimensional

Na Figura 13, pode ser visualizado o resultado obtido. O software possui uma interface amigável, interativa e baseada em menus. A cena, representada na Figura 12, foi colocada pra ser renderizada em um canvas estendido da classe wxGLCanvas da biblioteca wxWindows.

Quando é gerada uma falha, o operador de manutenção entra em ação, fazendo a inspeção do equipamento. Ele faz a verificação em cada componente do autotrafo que possivelmente estaria envolvido na falha gerada, pois quando a falha é gerada, a única coisa que ele sabe é o modo de falha. Esta inspeção é feita clicando-se em cada componente do autotrafo, fazendo com que o sistema, através de uma janela, informe as suas propriedades, como pode ser visto na Figura 14.

Para solucionar a falha, o operador deve selecionar corretamente, dentre uma lista de falhas predefinidas (Figura 15), qual é a aquela que ocasionou avaria no equipamento. Se for selecionada a falha correta, o sistema exibe os procedimentos de manutenção (Figura 16) utilizados pra solucionar o problema gerado.

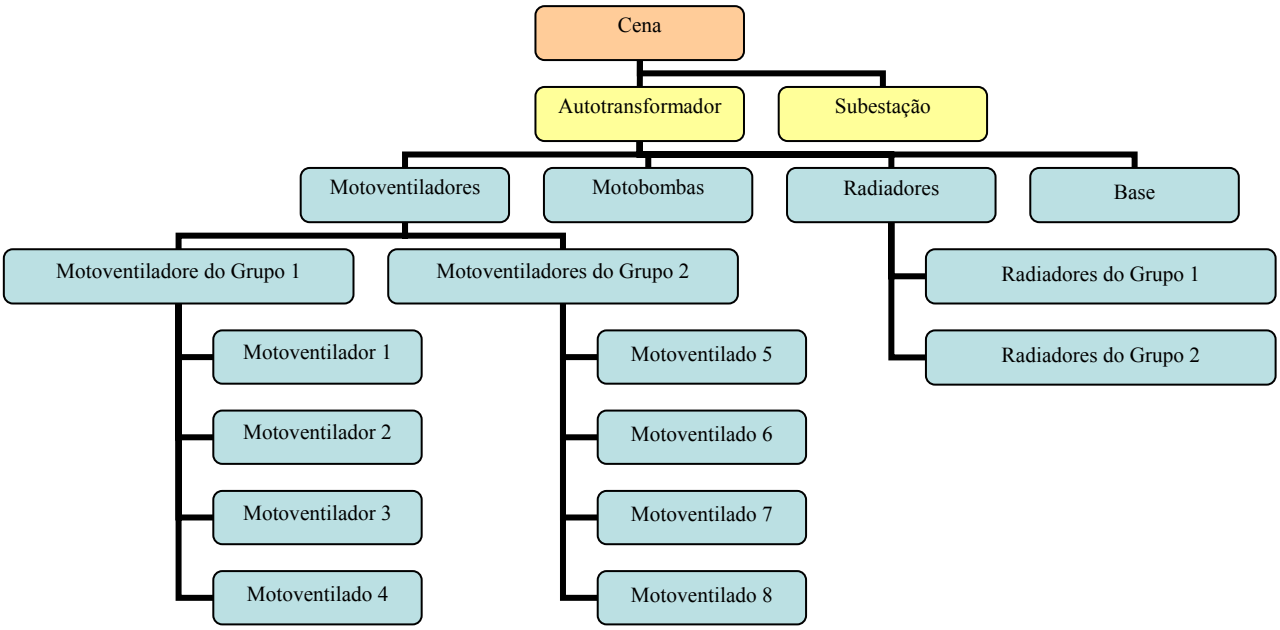

Figura 12 - Grafo de cena para representação do mundo virtual

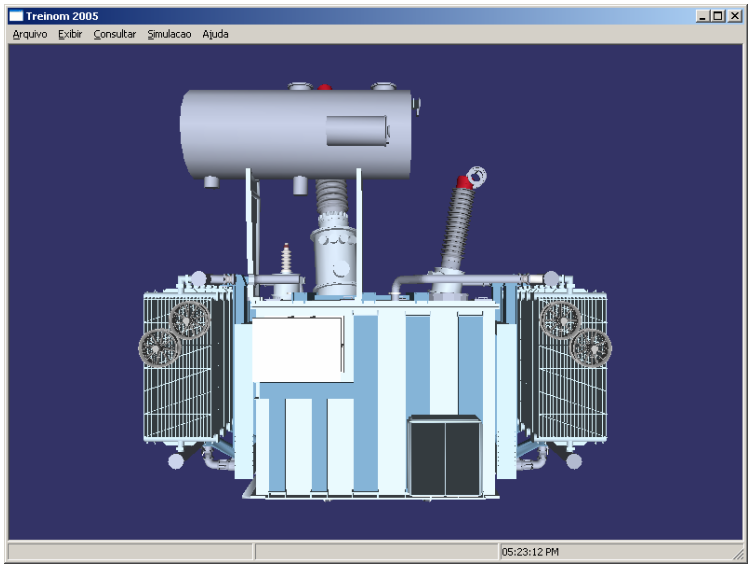

Figura 13 - Janela principal com o autotransformador.

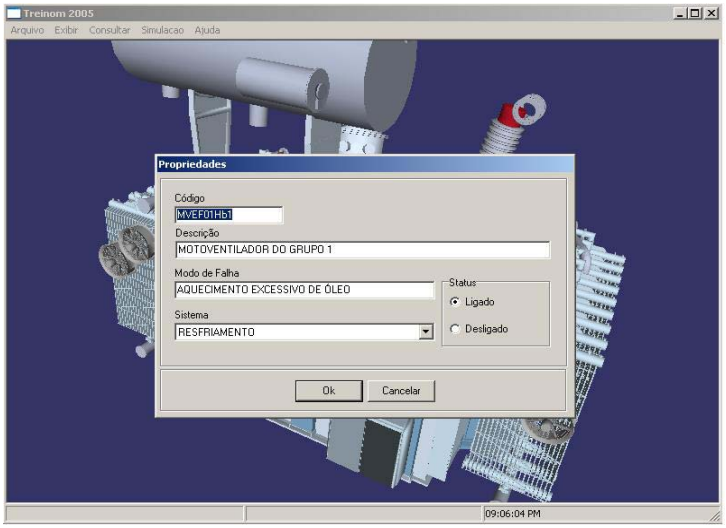

Figura 14 - Visualização das propriedades de um componente, durante a inspeção

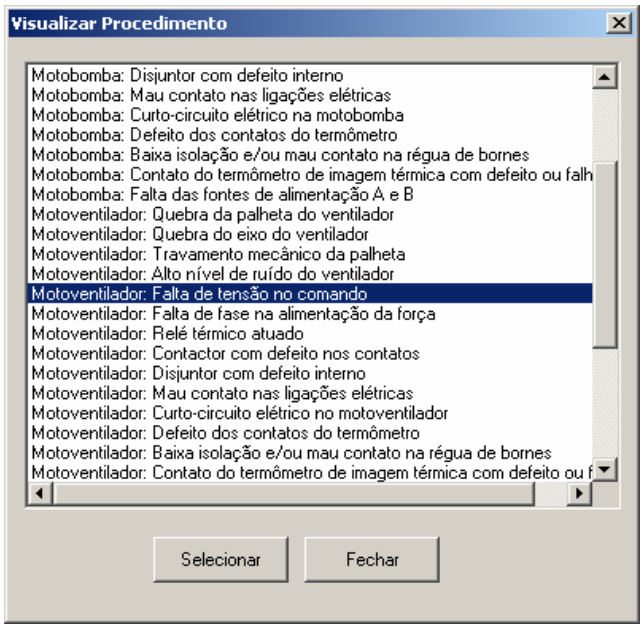

Figura 15 - Janela de seleção de falha

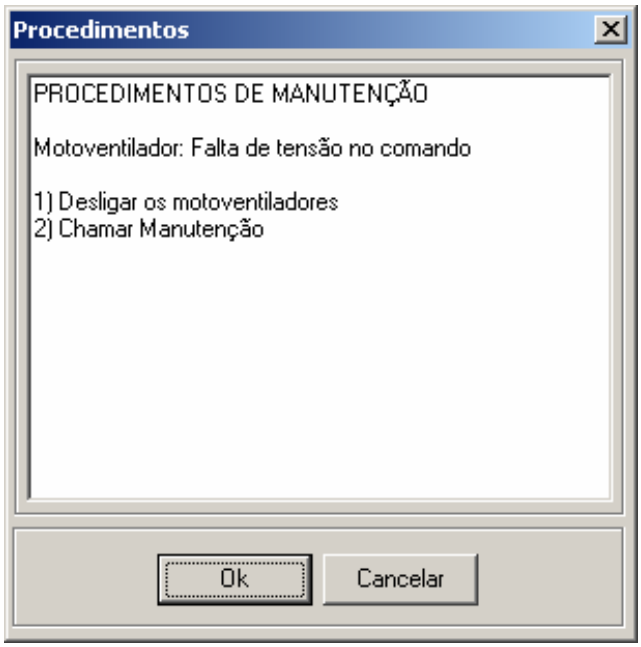

Figura 16 - Listagem dos procedimentos

Após todo este processo de construção de um ambiente virtual tridimensional interativo utilizando o OpenSceneGraph, pode-se concluir que a ferramenta até o momento apresentou bons resultados no que diz respeito a produtividade e performance. Obteve-se produtividade a partir do momento que se fez um estudo da ferramenta. Entretanto, a falta de documentação prejudica muito a produção de aplicações, inclusive no processo de instalação da biblioteca. Após esta etapa de estudo, foi verificado que o desenvolvimento do software foi feito em um curto período. Quando o software foi executado em PC´s, que não possuíam placas gráficas, a performance do OSG foi bem melhor que outras ferramentas, como o Java3D.

Outro fator que pode ser considerado é a facilidade de implementação que a biblioteca possui. Por ser orientada a objetos e utilizar padrões de software, constatou-se que é possível estender a API, sem precisar recompilá-la.

A integração com a biblioteca wxWindows também foi feita sem problemas de incompatibilidade. No entanto, nesta etapa, houve atraso no cronograma de desenvolvimento, também causado pela falta de documentação sobre como deveria ser feita esta integração.

A escolha do OSG como ferramenta para o desenvolvimento de aplicações tridimensionais é aceitável, desde que se tenha um tempo a mais pra dedicação ao aprendizado da ferramenta.

VI. Conclusões

A objetivo principal desse projeto foi desenvolver ferramentas que faca o treinamento de operadores de centros de controle, e um auxílio ao pessoal de manutenção de equipamentos como transformadores, interruptores, etc. das subestações.

O primeiro módulo evita o desligamento ou ligamento de equipamentos de forma inadequada no sistema elétrico real quando era feita uma seqüência de manobras, agora pode ser simulado antes de efetuar uma seqüência de manobra.

O segundo módulo pretende contribuir com uma melhor capacitação dos operadores e pessoal de manutenção com a utilização de simuladores e tutores de treinamento com características de sistemas inteligentes para o treinamento em manutenção de autotransformadores.

O protótipo foi projetado utilizando metodologia orientada a objetos e codificado na linguagem Java, Java3D e OpenSceneGraph.

A implementação da metodologia descrita foi testada com sucesso no sistema da ELETRONORTE.

VII. Referências bibliográficas

[1] M. D. Ilic, *Power Systems Restructuring Engineering and Economics*, Kluwer International Series in Engineering & Computer Science, 1998.

[2] G. P. Azevedo, B. Feijó and M. Costa, “Control centers evolve with agent technology”, IEEE *Computer Applications in Power*, vol.13, pp.48-53, Jul. 2000.

[3] S. Harrington, Computer Graphics: A Programming Approach, McGraw-Hill, 2000.

[4] V. L. Paucar, O. S. de Sousa Jr., I. O. Almeida, M. J. Rider, M. F. Bedriñana and J. H. Santos, “Software Development with Compute Graphics, Distributed Data Base and OOP for Deregulated Power Systems Analysis”, *Proceedings of the IEEE 2004 Large Engineering Systems Conference on Power Engineering (LESCOPE 2004)*, pp. 198–202, Nova Scotian, Canada, Jul. 28–31, 2004.

[5] J. Hollingworth, C++ Builder 5 Developer's Guide, SAMS, 2001.

[6] H. Schildt, G. Guntle, H. Schildt and G.L. Guntle, *Borland C++ Builder: The Complete Reference*, McGraw-Hill, 2001.

[7] H.M. Deitel and P.J. Deitel, *Java How to Program*, Prentice Hall, 6th Edition, 2004.

[8] A. Monticelli, *State Estimation in Electric Power Systems : A Generalized Approach (Power Electronics and Power Systems)*, Springer; 1 edition, 1999.