Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Unidade Acadêmica de Engenharia Elétrica
Programa de Pós-Graduação em Engenharia Elétrica

Dissertação de Mestrado

Módulo tutor para apoiar o treinamento de operadores em um ambiente simulado

Raffael Carvalho da Costa

Orientadora: Maria de Fátima Queiroz Vieira, Ph.D.

Campina Grande, Março de 2011.

Módulo tutor para apoiar o treinamento de operadores em um ambiente simulado

Raffael Carvalho da Costa

Dissertação de mestrado submetida à Coordenação dos Cursos de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande – Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciências no domínio da Engenharia Elétrica.

Área de Concentração

Engenharia da Computação.

Orientadora

Maria de Fátima Queiroz Vieira, PhD.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

C837m Costa, Raffael Carvalho da.
Módulo tutor para apoiar o treinamento de operadores em um ambiente simulado / Raffael Carvalho da Costa. ─ Campina Grande, 2011.
84 f.: il. color.

Dissertação (Mestrado em Engenharia Elétrica) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.
Orientadora: Profª. PhD. Maria de Fátima Queiroz Vieira.
Referências.

1. Treinamento Simulado. 2. Módulo Tutor. 3. Realidade Virtual. 4. Treinamento de Operadores de Sistemas Elétricos. I. Título.

CDU – 681.5.017(043)

MÓDULO TUTOR PARA APOIAR O TREINAMENTO DE OPERADORES EM UM AMBIENTE SIMULADO

RAFFAEL CARVALHO DA COSTA

Dissertação Aprovada em 28.03.2011

Maria de Fatima Q. Vieira

MARIA DE FÁTIMA QUEIROZ VIEIRA, Ph.D., UFCG
Orientador

EVANDRO DE BARROS COSTA, D.Sc., UFAL
Componente da Banca

Edmar Candeia Gu

EDMAR CANDEIA GURJÃO, D.Sc., UFCG
Componente da Banca

CAMPINA GRANDE - PB
MARÇO -2011

Dedicatória

Aos meus pais, minhas irmãs e a
minha namorada.

Agradecimentos

Agradeço primeiramente a Deus, por guiar meus passos e iluminar meu caminho.

Aos meus Pais, por toda dedicação, amor e sacrifício, sempre acreditando no meu potencial e incentivando meus estudos.

As minhas tão amadas irmãs, Dayane, Rayssa e Izabelly, por todo apoio, conselhos e pelos inúmeros momentos de alegrias partilhados.

A minha namorada, Elisabeth, pelo enorme amor e compreensão me dedicado, além da cobrança incansável pelo texto de minha dissertação.

Aos meus professores, em especial a professora e orientadora Maria de Fátima, por sua amizade, dedicação e paciência, compartilhando seus conhecimentos e orientando na execução deste trabalho.

Aos meus amigos e parceiros de laboratório (Ademar, Alves, Daniel's, Diego, Flávio, Gerson, Pedro, Yuska e nossa "agregada" Thiciany) que são as pessoas com quem mais convivi nos últimos anos, pelas sugestões, críticas, amizade e pelo ótimo ambiente de trabalho.

A todos os meus amigos que me acompanharam e participaram comigo de momentos de angústia e lazer. Em especial a André, Bruno, Sérgio e Reginardo, que são como irmãos para mim.

Um agradecimento muito especial também a galera "trash" do racha, vulgo "clube dos caminhoneiros", Gia, Lipan, Jhésus, Pumba, Manéu (leite de coco), Piru, Erick, Carlin, Daniel, João Luis, João Paulo, Leo e Vitória, amigos para todas as horas.

Resumo

A demanda por profissionais capacitados em períodos de tempo cada vez mais curtos tem levado à busca por estratégias mais eficazes e eficientes de treinamento. Neste contexto o uso de simuladores em Realidade Virtual (RV) apresenta-se como uma alternativa viável e eficaz. Através da RV é possível propiciar aos treinandos a vivência em situações reais, que de outra forma seria inacessível. Este trabalho consistiu na concepção e construção de um sistema tutor para integrar o projeto de um simulador para treinamento na operação de uma subestação elétrica. O módulo desenvolvido oferece recursos que apóiam a gestão de treinamentos e a avaliação do seu impacto. Na concepção do módulo tutor foi adotada uma arquitetura capaz de atender aos requisitos: modularidade, interoperabilidade e persistência. Ele também possibilita integrar o simulador a sistemas supervisórios comerciais, estendendo assim suas funcionalidades. As escolhas de projeto foram validadas a partir da realização de um experimento durante o qual foram avaliados aspectos de usabilidade e a adequação das soluções adotadas no projeto do tutor aos requisitos do projeto.

Palavras chave: Treinamento Simulado, Módulo tutor, Realidade Virtual, treinamento de operadores de sistemas elétricos.

Abstract

The demand for highly trained professionals in shorter periods of time has boosted the search for more effective and efficient ways of operator training strategies. In this context, the use of simulators using Virtual Reality (VR) presents itself as a viable and effective alternative. Trough VR it is possible to provide the trainees with real experiences otherwise inaccessible. This work consisted in conceiving and building a module tutor to be integrated the simulator developed to support training in the electric substation operating environment. The resulting module offers support to manage training programs and the evaluation of their impact. This module's architecture complies with the project requirements: modularity, interoperability and persistence. It also allows integrating the tutor to commercially available supervisory systems extending the simulator functionalities. The design choices were validated during an experiment that explored usability aspects and the adequacy of the choices to the simulator requirements.

Keywords: Virtual Reality, Simulated Training, Tutoring Module, electrical systems operator training.

Lista de abreviaturas

3D: Três dimensões

API: *Application Programming Interface*

AV: Ambiente Virtual

BD: Banco de Dados

CPN: *Colored Petri Net*

DTD: *Document Type Definition*

EAI: *External Authoring Interface*

GIHM: Grupo de Interface Homem Máquina

HTML: *HyperText Markup Language*

HTTP: *Hypertext Transfer Protocol*

IP: *Internet Protocol*

ISO: *International Organization for Standardization*

JDBC: *Java DataBase Connectivity*

LIHM: Laboratório de Interface Homem-Máquina

MSC: *Message Sequence Charts*

PDA: *Personal digital assistants*

RV: Realidade Virtual

SAI: *Scene Access Interface*

SGBD: Sistema de Gerenciamento de Banco de Dados

SML: *Standard ML*

TCP: *Transmission Control Protocol*

UML: *Unified Modeling Language*

VRML: *Virtual Reality Modeling Language*

X3D: *eXtensible 3D*

XML: *eXtensible Markup Language*

Lista de Figuras

Lista de Tabelas

SUMÁRIO

Capítulo 1 Introdução

O treinamento de operadores do setor energético se torna cada vez mais crítico e necessário, a fim de se evitar acidentes e incidentes ocasionados por falha humana. No caso particular dos ambientes de subestação elétrica, os quais se caracterizam como um sistema complexo a ser operado, há a demanda por profissionais competentes capazes de tomar decisões corretas e dentro de prazos estritos.

A evolução tecnológica de áreas como a computação, tem impulsionado a utilização de novas formas de interação homem-máquina, com destaque para a Realidade Virtual (RV). O termo RV é muito abrangente e muitas são as suas definições. Neste trabalho será adotada a definição de (KIRNER, et al., 2004): "Realidade Virtual é uma interface avançada para aplicações computacionais, na qual o usuário pode navegar e com a qual pode interagir, em tempo real, em um ambiente tridimensional gerado por computador, usando dispositivos multi-sensoriais[1]"

A utilização de RV nas atividades de entretenimento, tais como jogos, já vem sendo explorada pelas indústrias do ramo há algum tempo. No contexto de treinamento, o Laboratório de Interfaces Homem-Máquina (LIHM) da Universidade Federal de Campina Grande (UFCG) vem desenvolvendo o projeto do simulador do ambiente de operação de uma subestação elétrica, neste trabalho referenciado como Simulador, representando o ambiente do supervisório e dos painéis de controle. Um dos objetivos deste simulador é propiciar aos treinandos a vivência em situações reais de operação de outra forma inacessíveis, tais como a realização de manobras complexas e pouco frequentes e portanto sujeitas ao erro. Isto é possível a partir da representação em realidade virtual do ambiente físico de trabalho e dos objetos de interação que o compõem. No simulador cada objeto é associado a uma representação visual em realidade virtual e a uma representação do seu comportamento, no formalismo de redes de Petri (CPN)(JENSEN, 1992).

Outros simuladores já foram desenvolvidos para o contexto de sistemas elétricos, a exemplo do SAGE/OTS (LEITE, et al., 2002), que é o resultado da integração do sistema supervisório SAGE com um simulador de sistemas elétricos de potência - o OTS (*Operator*

[1] Dispositivos multi-sensoriais são equipamentos que estimulam os sentidos humanos, normalmente incorporam sensores e atuadores que monitoram e podem responder as ações dos usuários.

Training Simulator) do EPRI (*Electric Power Research Institute*). Outro exemplo de simulador é o sistema computacional ASTRO (Ambiente Simulado para Treinamento de Operadores) desenvolvido pelo CEPEL/ ELETROSUL para operar integrado ao SAGE. Há ainda o sistema STPO (Simulador para Treinamento de Proteção e Operação de Sistemas Elétricos) desenvolvido para representar uma subestação da COELCE, no qual é possível simular faltas e re-configurar o sistema (BEZERRA, et al., 2007). O STPO foi integrado a um ambiente virtual de ensino-aprendizagem para permitir o treinamento à distância.

Contrastando com os trabalhos citados, nos quais o controle e a monitoração são restritos ao ambiente do supervisório, o ambiente do Simulador do LIHM possibilita o controle e a monitoração do sistema elétrico tanto através do ambiente do supervisório, quanto através de painéis de controle, ambos representados no ambiente virtual tridimensional. No entanto este projeto ainda não conta com um módulo para auxiliar um tutor na realização de um treinamento.

Em geral, a função principal do tutor é de apoiar o processo de ensino-aprendizagem facilitando a aquisição de conhecimentos por parte dos aprendizes. Nesse processo, um Sistema Tutor pode oferecer recursos para auxiliar um tutor humano ou substituí-lo.

Em alguns Sistemas Tutores Inteligentes (STI) o papel do tutor é desempenhado por uma máquina, esses sistemas aliam métodos e técnicas de inteligência artificial na concepção de ambientes de ensino e aprendizagem assistidos por computador. De acordo com (FOWLER, 1991) os STI são programas de computador com propósitos educacionais e que incorporam técnicas de Inteligência Artificial. Oferecem vantagens sobre outros sistemas, pois podem simular o processo do pensamento humano para auxiliar na resolução de problemas ou em tomadas de decisões.

O módulo tutor proposto no presente trabalho consiste de um ambiente interativo

1.1 Objetivo Geral

O objetivo neste trabalho é o desenvolvimento de um módulo tutor para o Projeto do Simulador. Este módulo apoiará a configuração do Ambiente Virtual (AV), e possibilitará a gestão de treinamentos a partir da edição e salvamento de cenários de treinamento, além de armazenar informações sobre os usuários e os resultados de seus treinamentos (desempenho e

registro do *log* de ações). Este módulo também é responsável pelo acesso remoto ao Simulador.

Para o desenvolvimento desde trabalho é necessária a validação da arquitetura proposta por (SILVA NETTO, 2010), com ênfase nos pontos destacados na Figura 1.

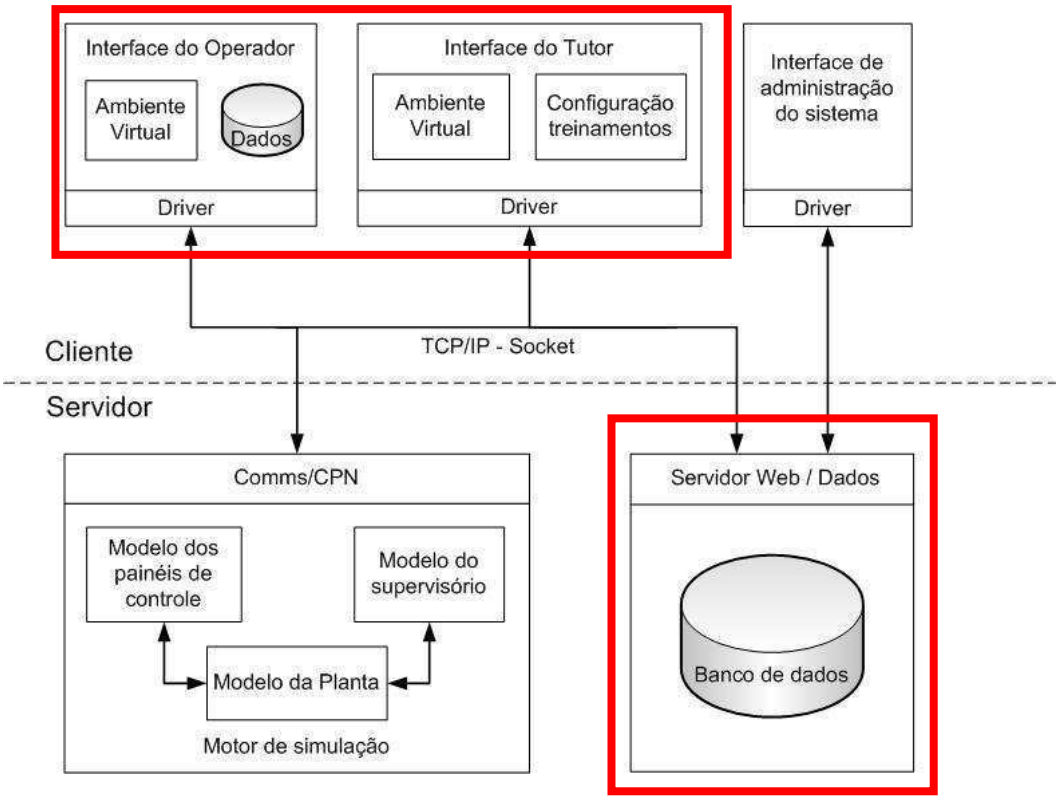

Figura 1 - Relação do módulo tutor com a arquitetura do simulador.

A arquitetura, ilustrada na Figura 1, é do tipo cliente-servidor e a comunicação entre o servidor, o ambiente do tutor e o ambiente do operador se dá através de socket utilizando protocolo TCP/IP (Transmission Control Protocol/ Internet Protocol).

Segundo a arquitetura apresentada, o simulador é composto por ambiente do tutor, ambiente do treinando, banco de dados e motor de simulação.

1.1.1 Objetivos Específicos

Como objetivos específicos presentes neste trabalho, temos:

- investigar sobre os diversos tipos de sistemas multiusuários para RV existentes e suas respectivas técnicas de desenvolvimento;
- implementar um banco de dados para o sistema;

- projetar interfaces ergonômicas para tutores e treinandos;
- produzir documentação formal do projeto;
- validar o módulo proposto a partir de testes com usuários.

1.2 Aspectos Metodológicos

Para a execução deste trabalho foi adotada a seguinte metodologia:

1. Revisão bibliográfica sobre os temas: Realidade Virtual (RV); motores de simulação; aplicações em RV; Sistemas de Gerenciamento de Banco de Dados (SGBD);
2. Reavaliação dos requisitos do Simulador, do ponto de vista do sistema tutor;
3. Análise, ampliação e validação do modelo do banco de dados (BD) proposto por (BARBOSA, 2010) integrando-o ao simulador a partir de um *driver* de comunicação entre o Ambiente Virtual e o BD;
4. Desenvolvimento e validação do módulo tutor (sistema de distribuição, rotinas de edição e salvamento de cenários);
5. Integração do módulo tutor ao projeto do simulador;
6. Redação do texto da dissertação e de um artigo técnico.

1.3 Estrutura do documento

No Capítulo 2, aborda-se uma breve fundamentação teórica onde são discutidos temas relacionados a esta pesquisa, tais como sistemas tutores, RV e motor de simulação.

No Capítulo 3, apresenta-se o projeto do módulo tutor, especificação, requisitos e alguns dos seus casos de usos, descreve-se ainda a arquitetura do sistema.

No Capítulo 4, abordam-se as ferramentas e técnicas utilizadas para o desenvolvimento e implementação do módulo tutor.

O Capítulo 5 apresenta a metodologia de validação do sistema e os resultados obtidos com a sua aplicação.

O Capítulo 6 apresenta as considerações finais e as sugestões para trabalhos futuros. Finalmente, são apresentadas as referências bibliográficas utilizadas para redação desta dissertação.

Capítulo 2 Revisão da literatura

Neste capítulo aborda-se uma breve fundamentação teórica onde são discutidos temas relacionados a esta pesquisa tais como RV, motor de simulação e XML.

2.1 Realidade Virtual

O termo Realidade Virtual (RV) é creditado a Jaron Lanier em 1987. Este termo é muito abrangente e muitas são as suas definições. Como já foi mencionado anteriormente, neste trabalho adotaremos o conceito proposto por (KIRNER, et al., 2004).

Três características são essenciais para os sistemas de RV: imersão, interação e envolvimento. Imersão está ligada à percepção de realismo provocada pela sensação de estar inserido no ambiente simulado, e de ser capaz de manipular os objetos ali presentes, como se fossem reais. A interação diz respeito à comunicação usuário-sistema. Já o envolvimento reflete o grau de motivação do indivíduo com uma determinada atividade, podendo ser passivo, a exemplo de ler um livro ou assistir televisão, ou ativo, a exemplo da prática de esportes.

2.2 Motor de simulação

O motor de simulação em um ambiente de RV é um componente de *software* que permite simplificar e abstrair o desenvolvimento de aplicações gráficas em tempo real, para computadores e/ou videogames. Ele oferece a infra-estrutura necessária para o desenvolvimento de uma aplicação (construção de um mundo virtual), e pode ser reaproveitado na criação de aplicações similares. Um termo comumente usado para designar este componente, principalmente quando se trata de concepção de jogos é "motor do jogo", derivado do termo em inglês *game engine*, sendo também denominado simplesmente *engine*.

(TORI, et al., 2004) cita algumas características de um motor de jogo, que são comuns aos motores de simulação: módulo para *renderização*, inteligência artificial, comunicação em rede, recursos para simulação do mundo físico (com detecção e tratamento de colisão),

recursos para representação do comportamento dos objetos, recursos multimídia (com ambientação de som e música) e recursos para o processamento das entradas do usuário.

Portanto, o motor de simulação é o integrador de diferentes componentes, que vão das representações gráficas 2D, modelos 3D e animações, áudio, interfaces com dispositivos de E/S e recursos de rede. O motor de simulação oferece ao desenvolvedor o reaproveitamento de código resultando em um considerável ganho de tempo, pois suas funcionalidades permitem o desenvolvimento rápido de aplicações semelhantes, sendo classificados como uma ferramenta RAD – *Rapid Application Development*.

Existem diversas propostas de arquiteturas para motores de simulação e motores de jogos, para ilustrar a estrutura de *software* destas soluções computacionais será apresentada a arquitetura genérica de um motor de jogo (Figura 2) definida por (BITTENCOURT, et al., 2004).

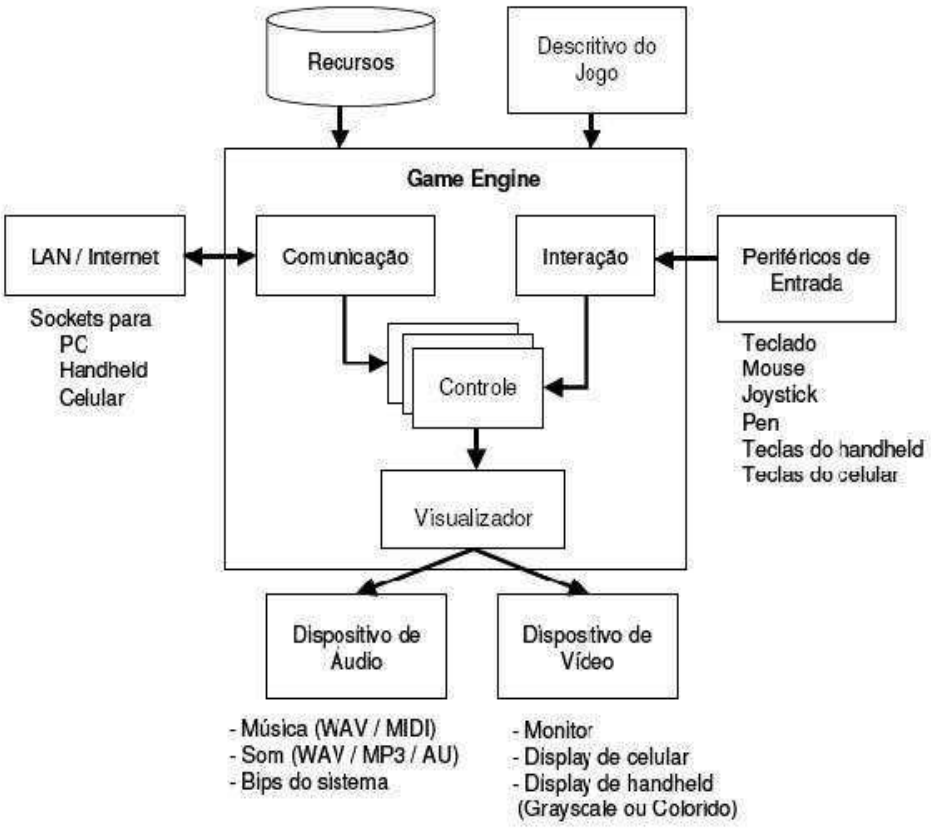

Figura 2 - Arquitetura genérica de um motor de jogo (BITTENCOURT, et al., 2004).

Nesta arquitetura um motor de jogo possui os seguintes componentes: Interação, Comunicação, Controle e Visualização.

O componente de Interação é responsável pelo tratamento de eventos gerados pelos periféricos de entrada, tais como teclado, *mouse* e *joystick*.

O componente de Comunicação permite a criação de aplicações em rede com múltiplos participantes. Na maioria das aplicações utiliza-se um processo de comunicação por *sockets* os quais são implementados de forma diferenciada em cada plataforma de execução. Os componentes de Comunicação e de Interação são responsáveis por fornecer as entradas para o motor de jogo.

O componente de Controle executa a lógica do jogo e manipula os objetos do jogo. Um motor de jogo pode conter inúmeros controladores cada um responsável por uma operação lógica, ou seja, podem existir controladores de simulações físicas e controladores de Inteligência Artificial, por exemplo. É importante destacar que estes controladores poderão ser reusados em diferentes jogos digitais.

O Visualizador cria as imagens que representam o estado atual do jogo, resultante do processamento realizado pelos componentes de Controle. O Visualizador utiliza um dispositivo de vídeo, a exemplo de um monitor, displays de PDAs ou telefones celulares para apresentar as imagens.

Em suma, pode-se afirmar que um motor é composto por diversos módulos, cada um responsável por tratar um aspecto envolvido na construção de jogos (BITTENCOURT, et al., 2004).

No projeto do simulador, o componente do motor de simulação modela o comportamento dos objetos representados no mundo virtual de uma sala de operação de uma subestação de sistema elétrico, tais como: disjuntores, transformadores e chaves seccionadoras. Os modelos de comportamento são construídos em redes CPN (JENSEN, 1992). A notação gráfica e matemática do formalismo das redes de Petri coloridas facilitaram a verificação e a validação dos modelos. Também, ao adotar o formalismo CPN foi possível representar o comportamento dos objetos diante de eventos com características de concorrência e paralelismo. O uso de modelos de objetos já validados e disponíveis em bibliotecas construídas a partir de trabalhos anteriores minimizou o esforço de modelagem e potenciais erros de representação (TORRES FILHO, et al., 2010).

2.3 XML (eXtensible Markup Language)

XML (*eXtensible Markup Language*) é uma linguagem de marcação de dados (*meta-markup language*) que provê um formato para descrever dados estruturados. Originada da

SGML (*Standard Generalized Markup Language*), tem como características principais a independência dos dados e a separação do conteúdo de sua apresentação.

A XML foi desenvolvida pela W3C (*World Wide Web Consortium*) e tem como propósito principal facilitar o compartilhamento de informações e permitir que o usuário crie a sua própria marcação, quando necessário. O XML é um padrão flexível, aberto, independente de dispositivo, permite múltiplas formas de visualização dos dados estruturados. Esta linguagem vem sendo cada vez mais utilizada em banco de dados, pois a natureza estruturada de um documento XML, ao invés da formatada, permite que ele seja manipulado por aplicativos de banco de dados (DEITEL, et al., 2003).

Capítulo 3 Projeto do módulo tutor

A partir do estudo descrito nos capítulos anteriores e com base no projeto inicial do Simulador elaborou-se o projeto do módulo tutor. Foram documentados alguns pontos que são essência para a compreensão do funcionamento do sistema, como, por exemplo, a análise de requisitos e o desenvolvimento de alguns diagramas comportamentais UML (*Unified Modeling Language*) e da arquitetura do sistema.

O projeto do Simulador se originou da associação de uma representação gráfica, ao modelo de navegação em interfaces de sistemas de supervisão, representado no formalismo rede de Petri Colorida (CPN) (FREITAS, et al., 2006b). O sistema modelado foi o de uma subestação de um sistema elétrico ao qual foi associada uma representação 3D, em Realidade Virtual, construída utilizando a linguagem VRML (V*irtual Reality Modeling Language*). O objetivo desta associação foi facilitar a análise da navegação na interface com o sistema para projetistas leigos no uso do formalismo. O passo seguinte do projeto consistiu na construção de uma biblioteca de modelos (NASCIMENTO, et al., 2007) (NASCIMENTO, et al., 2007) para representar os objetos de interação tipicamente encontrados em uma subestação, tais como chaves, botões, quadros de alarmes, etc. Estes objetos foram em seguida integrados ao modelo da navegação na interface para controle uma subestação do sistema elétrico.

Na Figura 3 é ilustrada a representação 3D da sala de controle de uma subestação (SE), construída na linguagem VRML por (FREITAS, et al., 2006a); constituindo a primeira versão do Simulador. Nesta versão, os painéis de controle e o comportamento dos objetos de interação (chaves e botões) foram representados em um modelo CPN. A conexão entre o modelo CPN e a representação visual dos objetos utilizou a biblioteca Comms/CPN e Java Script Interface (JSI) (HORSTMAN, 2004). O visualizador de RV utilizado foi o Freewrl (FREEWRL, 2009) para plataforma Linux.

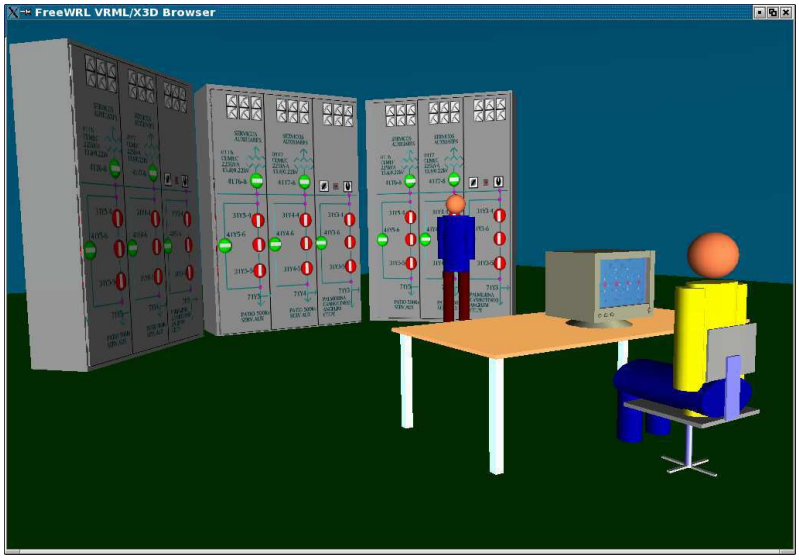

Figura 3 - Representação elaborada por Freitas. (Fonte: FREITAS, 2006a).

Nesta primeira versão do Simulador, a representação do ambiente virtual era extremamente simples e detalhes importantes do ambiente físico modelado, tais como o ambiente físico da sala de controle da subestação, não foram contemplados. Na Figura 4 é apresentada a arquitetura adotada por (FREITAS, et al., 2006a).

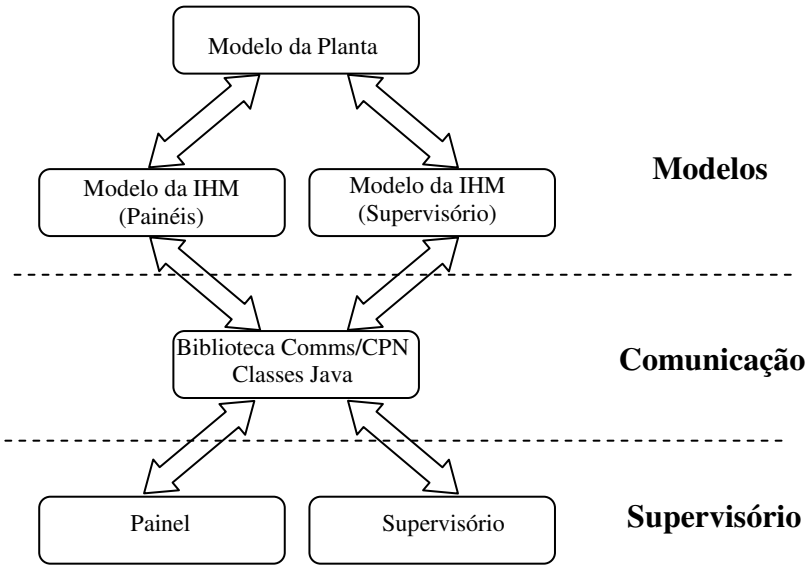

Figura 4 - Arquitetura adotada por Freitas. (Fonte: FREITAS, et al., 2006a).

Esta idéia inicial de arquitetura foi revista e ampliada dando origem a uma nova versão de arquitetura, proposta por (SILVA NETTO, 2010). Entretanto, está nova versão de arquitetura não incluía características importantes do módulo tutor e do módulo relativo ao supervisório, outro trabalho em andamento no LIHM.

Assim, a arquitetura do projeto foi novamente alterada para se adequar aos novos requisitos do módulo tutor e para acomodar o módulo supervisório. A Figura 5 representa esta nova arquitetura, atualmente em desenvolvimento.

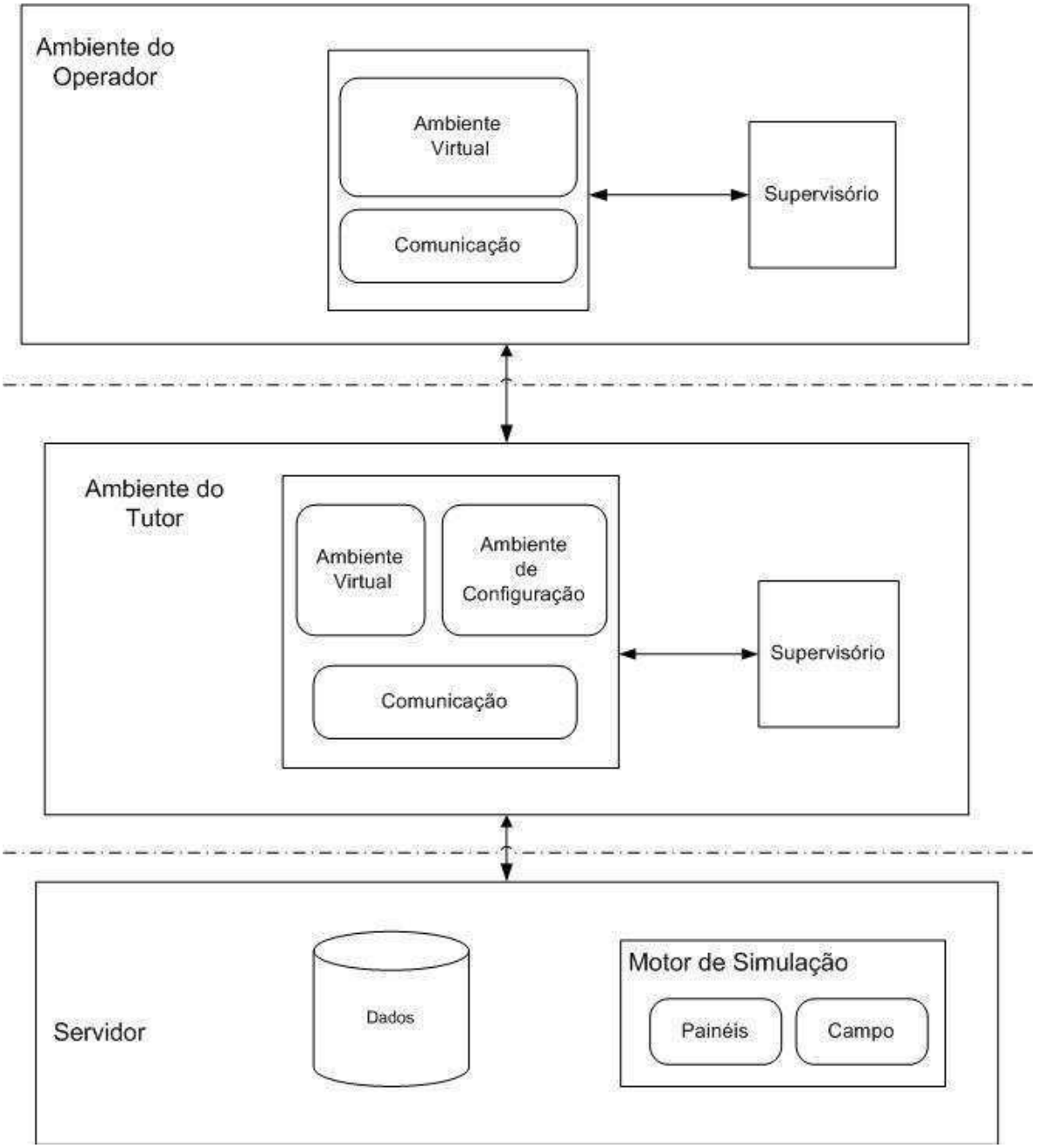

Figura 5 - Arquitetura do simulador em desenvolvimento pelo LIHM.

3.1 Especificações do sistema

Nesta seção é apresentada uma visão do sistema, mostrando de modo geral seus principais componentes, suas funcionalidades e como são conectados. Para tanto, são utilizados alguns diagramas UML que permitem a melhor compreensão desta perspectiva. Posteriormente, são abordados os componentes com maior complexidade e de maior importância para este projeto.

3.1.1 Requisitos do módulo tutor

A arquitetura proposta em (SILVA NETTO, 2010) teve como objetivo atender os seguintes requisitos: estender as funcionalidades do simulador; facilitar a representação de uma variedade de contextos; comunicar com processos externos (modelos do sistema, outros aplicativos); oferecer persistência de dados; acomodar tempos de simulação estritos; utilizar software livre na construção e execução do simulador. Os requisitos propostos por SILVA NETTO foram revistos para este trabalho e os que impactam no projeto do módulo tutor encontram-se listados a seguir:

- permitir o acesso remoto: os usuários (tutores e treinandos) devem poder acessar o simulador via rede (Local ou longa distância), possibilitando a realização de treinamentos à distância (ambiente do tutor distante geograficamente do ambiente dos operadores), reduzindo os custos do treinamento;
- oferecer um Ambiente colaborativo – multiusuário: possibilitar que mais de um operador interaja no mesmo ambiente, emulando o trabalho em equipe, permitindo o treinamento simultâneo de vários operadores no mesmo ambiente virtual, além de permitir ao tutor acompanhar a execução do treinamento;
- ser apoiado por um banco de dados: armazenar dados gerenciais sobre treinamentos, participantes, instalações, cenários e resultados, sobre as representações virtuais das instalações e seus equipamentos e sobre os modelos que compõem o motor de simulação;
- facilitar a configuração e o salvamento dos cenários de treinamento, a partir do reuso de código e de modelos;
- oferecer múltiplas interfaces com os vários tipos de usuário: disponibilizando um módulo no cliente para cada perfil de usuário. Assegurar a usabilidade nos múltiplos

níveis de interação, para todos os usuários: tutores, operadores e desenvolvedores, responsáveis pela adequação do simulador a novos contextos de treinamento;

- aumentar a fidelidade na representação gráfica do ambiente real simulado: a partir da escolha adequada da ferramenta de representação gráfica ;
- construir um ambiente multi-plataforma, compatível com Windows e Linux.
- interoperabilidade: levar em consideração a interoperabilidade do sistema (propondo, por exemplo, o uso de XML);
- optar por um Projeto modular para facilitar a manutenção, ampliação e adaptação a outros contextos.

3.1.2 Diagramas de casos de uso do módulo tutor

Um caso de uso pode ser definido formalmente como um tipo de classificador representando uma unidade funcional coerente provida pelo sistema, subsistema, ou classe manifestada por seqüências de mensagens intercambiáveis entre os sistemas e um ou mais atores. Um ator é um humano ou máquina que interage com o sistema para executar um trabalho significante. Ou seja, em suma, o caso de uso representa uma unidade discreta da interação entre um usuário (humano ou máquina) e o sistema. Algumas das funcionalidades propostas para o Simulador são apresentadas nas Figura 6 e 7 através dos diagramas de casos. Nos diagramas pode-se observar as duas classes de usuários que utilizarão o sistema: o treinando e o tutor.

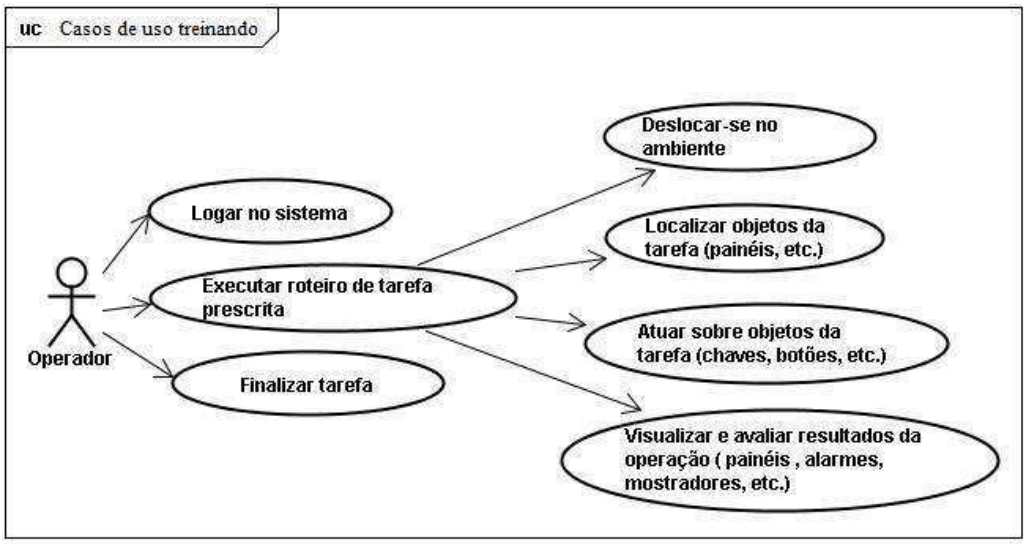

Figura 6 - Casos de uso do treinando.

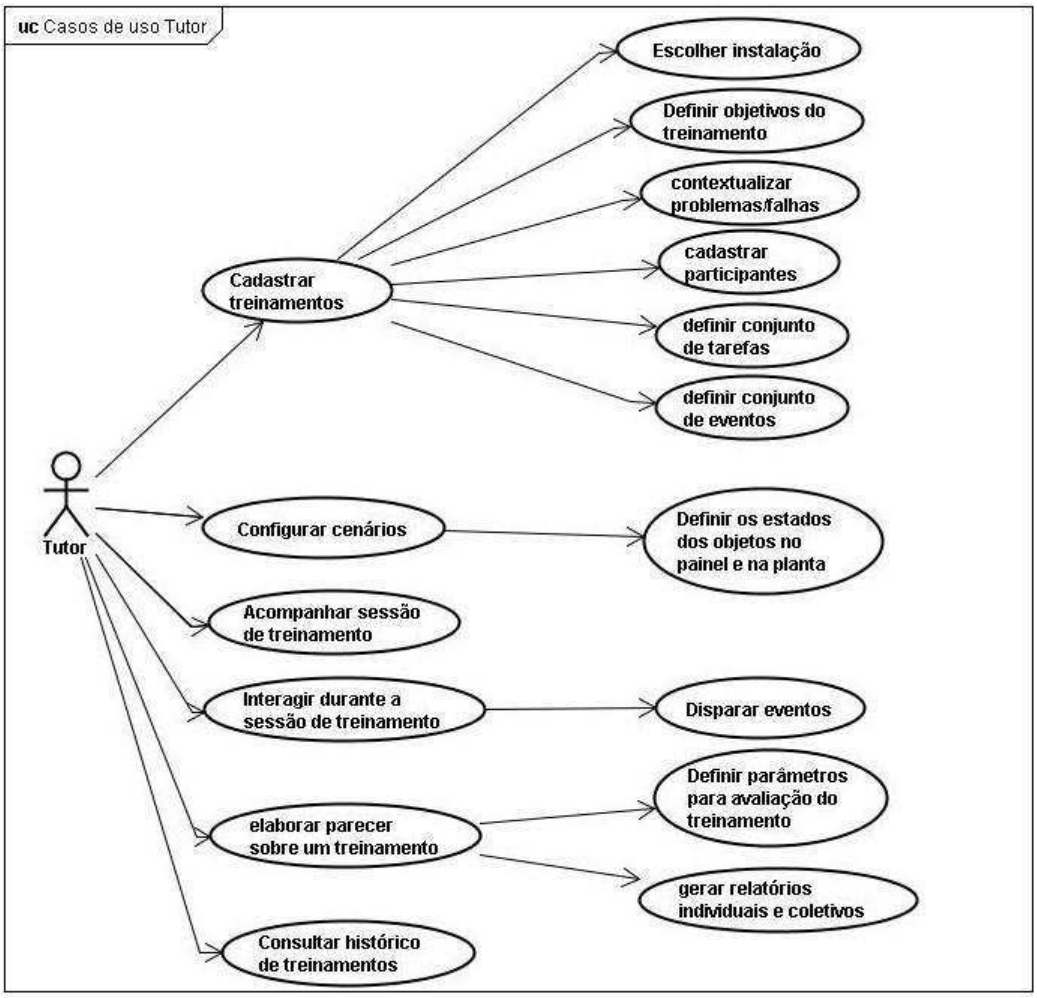

Figura 7 - Casos de uso do tutor.

O sistema possui atualmente duas interfaces gráficas (GUI – Graphics User Interface) distintas, uma para tutores outra para treinandos. Ambas as interfaces são acessadas a partir de uma tela de *login*, cuja função efetiva é restringir a entrada de usuários em áreas indevidas do sistema. Basicamente, ele coleta informações como login e senha e verifica a sua existência no banco de dados, caso ele encontre o usuário é liberado para a área referente ao seu tipo de usuário, caso contrário ele bloqueia o acesso ao sistema.

A partir da interface do tutor é possível carregar, editar e salvar cenários de treinamentos no BD, permitir o acesso remoto dos treinandos ao AV, acompanhar e interagir nas sessões de treinamento, elaborar pareceres individuais e coletivos, reproduzir as ações dos treinandos no AV e consultar o histórico de treinamentos.

Os treinandos poderão se conectar a um AV remoto gerido por um tutor e executar o roteiro de tarefas prescritas para o treinamento em questão, se locomovendo e interagindo

com os objetos da tarefa. Todas as ações dos treinandos são automaticamente salvas no BD, as quais ficam disponíveis para a análise dos tutores.

3.1.3 O sistema de distribuição do módulo tutor

Os sistemas de RV multiusuários constituem um campo crescente e próspero da pesquisa computacional pelo seu elevado potencial de aplicação. Estes sistemas permitem que os usuários geograficamente dispersos atuem em ambientes virtuais compartilhados por meio da troca de informações.

Sistemas de RV multiusuários podem ser centralizados ou distribuídos (GOSSWEILER, et al., 1994). No modelo centralizado um computador recebe todas as informações provenientes de usuários que estão conectados e que estejam em diferentes máquinas, armazena as mudanças ocorridas e as aplica no AV do computador central, que devolve para cada um dos usuários conectados os resultados das alterações feitas, atualizando-os. A Figura 8 ilustra a estrutura do modelo centralizado.

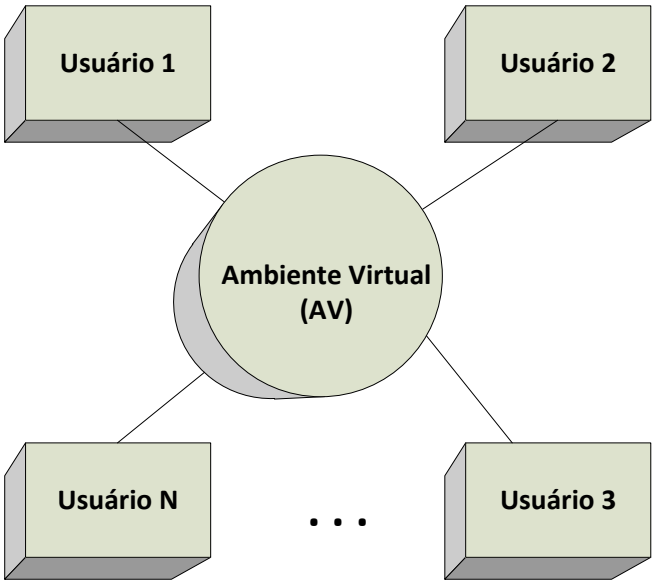

Figura 8 - Modelo centralizado de sistema de RV multiusuários.

O modelo centralizado apresenta facilidades em relação à implementação das estruturas de controle da comunicação entre usuários. Porém, ele possui restrições quanto à escalabilidade de distribuição. A escalabilidade refere-se ao fato do AV crescer e suportar uma grande quantidade de usuários sem perdas na qualidade de comunicação. Isso se deve ao

fato de que quanto maior o número de usuários em um AV, maior será o tráfego de mensagens com o computador central e dessa maneira a velocidade de acesso cai, pois o computador terá que receber, processar e devolver as mensagens a todos os usuários.

O modelo distribuído melhora a escalabilidade em relação ao modelo centralizado. Neste modelo cada usuário possui uma cópia ou parte do ambiente virtual em sua máquina, realizando, ele mesmo, as tarefas de renderização, computação e animação dos objetos do mundo virtual. Assim, quando um usuário realiza qualquer alteração sobre o AV, ele mesmo se encarrega de comunicar a todos os outros usuários sobre a atualização feita. A Figura 9 ilustra a estrutura de um modelo distribuído.

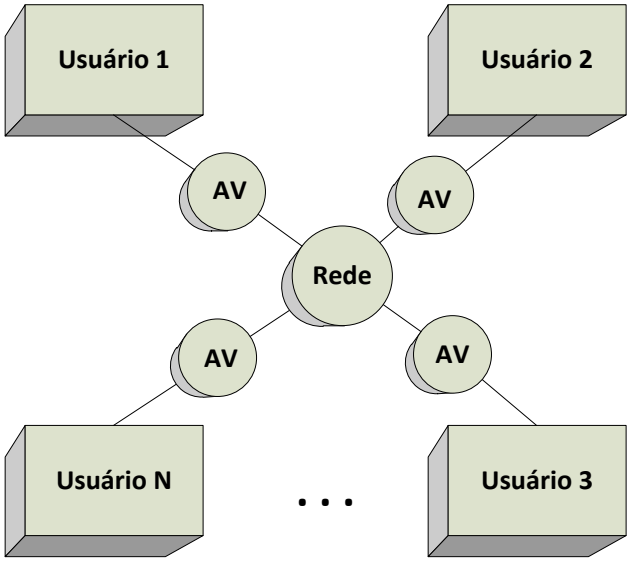

Figura 9- Modelo distribuído de sistema de RV multiusuários.

No modelo distribuído, o ambiente virtual pode ser replicado (para mundos pequenos) ou particionado (para mundos virtuais de grande porte). Em um sistema replicado com "n" usuários, qualquer alteração no ambiente virtual é comunicada para todas as suas (n-1) versões, onde estão os outros usuários, constituindo, dessa forma, a difusão chamada de *broadcast*, na qual uma única mensagem é enviada para todos os usuários que estão conectados na rede. A Figura 10 ilustra este modelo.

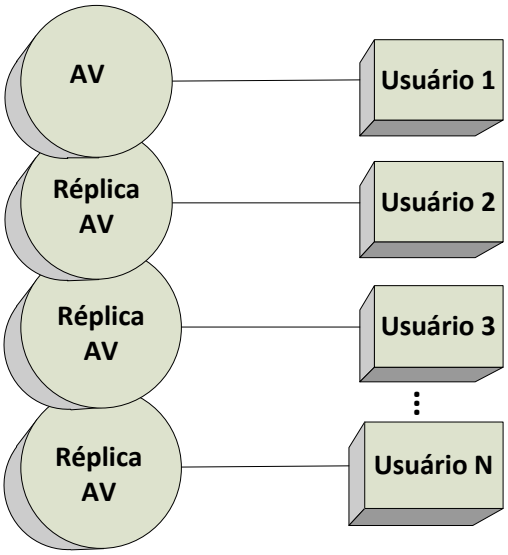

Figura 10 - Modelo distribuído replicado.

Já em um sistema particionado com "n" usuários, a situação é mais complexa, uma vez que o ambiente virtual é dividido em várias partes e cada máquina ficará encarregada de uma delas. Como o usuário pode navegar no ambiente virtual, ele poderá penetrar em outras regiões, de forma que sua máquina ou servidor deverá receber uma réplica da região, onde ele se encontra. Assim cada máquina estará cuidando de uma região fora da sua parcela. Se existirem vários usuários em uma mesma região do ambiente virtual, esse grupo de usuários receberá uma cópia dessa região. Qualquer alteração no ambiente virtual, feita por um membro do grupo, será retransmitida para o restante do grupo, constituindo a retransmissão por grupo conhecida como *multicast*. Da mesma forma que ocorre no *broadcast*, onde uma única mensagem é enviada para todos que estão conectados ao ambiente, ocorre com a transmissão *multicast*, com a diferença que só os usuários interessados nas mensagens irão recebê-las. A Figura 11 ilustra o modelo distribuído com sistema particionado.

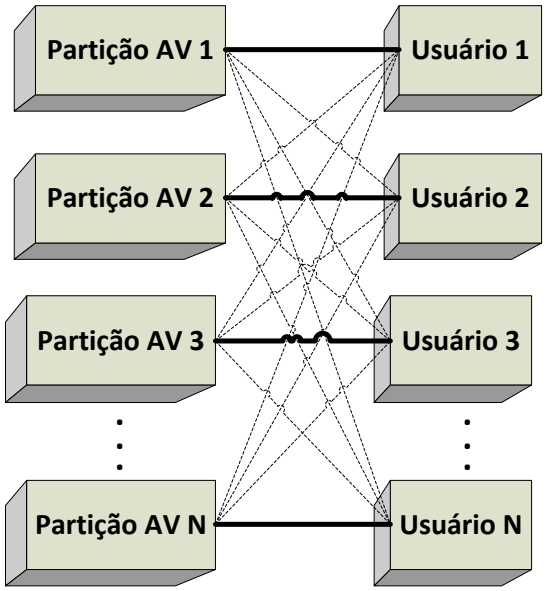

Figura 11 - Modelo distribuído particionado.

Para o projeto do Simulador foi implementado um modelo distribuído replicado, pois considerou-se como sendo um AV de pequeno porte, sendo assim, desnecessária a aplicação do modelo particionado. As próximas seções apresentam maiores detalhes sobre os métodos usados para realizar a distribuição do Simulador, tornando-o um ambiente colaborativo multiusuário.

3.1.4 Diagramas de atividade do sistema de distribuição

A comunicação entre os usuários é realizada com base no modelo distribuído replicado, sendo que a sua conexão baseia-se no modelo cliente/servidor. Deste modo entende-se cada réplica do AV como um cliente. O servidor é um componente presente no módulo tutor que possui a função de viabilizar a troca de dados entre ele e os demais clientes (treinandos). Em suma, em relação à distribuição, cada cliente pode realizar duas ações: enviar e receber dados.

Para enviar dados de um cliente para outro, inicialmente, o sistema os envia ao componente servidor presente no módulo tutor. O servidor, por sua vez, reenvia os dados recebidos para o restante dos clientes e para o motor de simulação que processa a informação retornando uma resposta que é retransmitida para todos os clientes. A Figura 12 apresenta o diagrama de atividades relativo à seqüência de processos descrita acima.

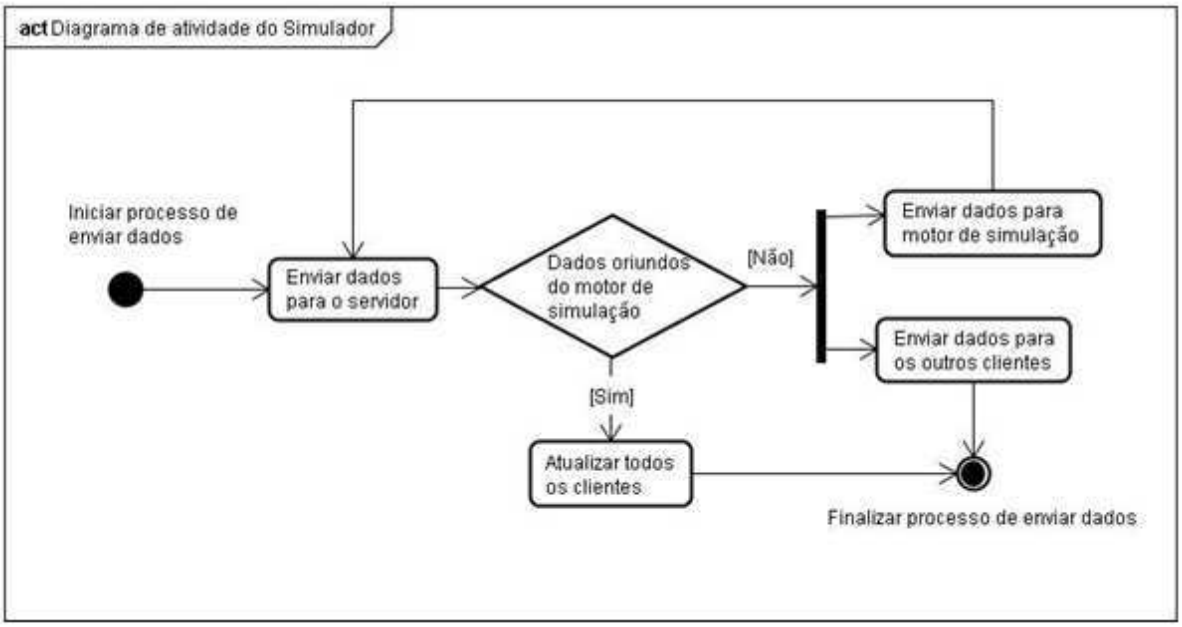

Figura 12 - Diagrama de atividade do sistema de distribuição do Simulador.

Capítulo 4 Desenvolvimento

Neste capítulo são abordados os detalhes de implementação do módulo tutor com maior relevância para o desenvolvimento deste trabalho. São apresentados, quando necessário, trechos do seu código e discutidas as suas respectivas funcionalidades no sistema.

4.1 Tecnologias utilizadas

Antes de apresentar efetivamente os detalhes de desenvolvimento deste sistema, será realizada uma breve exposição sobre as tecnologias necessárias para o seu funcionamento e concepção, bem como, será destacada a importância do uso de cada uma delas.

4.1.1 X3D

Existem diversas abordagens sobre a RV, algumas delas requerem o uso de tecnologias avançadas como plataformas de *hardware* e *software* sofisticadas. No entanto, a linguagem VRML, acrônimo de *Virtual Reality Modeling Language*, possibilita a criação de mundos virtuais tridimensionais, com alta qualidade, aplicáveis à Internet, utilizando apenas um *browser* e um *plug-in* para realizar a visualização. A VRML permite a descrição de um espaço tridimensional, tornando possível construir mundos correspondentes a uma dada realidade ou completamente imaginários. Fundamentalmente, o *browser* gera em tempo real a representação visual equivalente à descrição textual da cena.

O X3D (Extensible 3D) é uma linguagem de descrição de cenas 3D de padrão aberto, capaz de representar e comunicar cenas tridimensionais e objetos, desenvolvida com a sintaxe XML. Essa sintaxe foi escolhida porque permite uma melhor interoperabilidade entre aplicações (um dos principais objetivos para a criação do X3D) e permite incorporar novas tecnologias de forma padronizada(WEB 3D, 2009).

Desenvolvido a partir da VRML, o X3D apresenta um conjunto de novas funcionalidades, tais como interface avançada de programação de aplicações, formatos de codificação de dados adicionais, conformação mais estrita e arquitetura dividida em

componentes, o que permite o suporte modular ao padrão, e ainda ocorreram trocas de alguns nomes de campos para uma maior consistência. Porém ele mantém os recursos básicos da VRML como, por exemplo: oferecer suporte para gráficos 3D, transformações de geometria, iluminação, materiais e texturas; permite animação com temporizadores e interpoladores de condução contínua; permitir interação com o mouse e entradas de teclado; a navegação no ambiente acontece por meio do uso de câmeras, com características de colisão, proximidade e visibilidade, detecção e vários tipos de iluminação; por meio de um nó *script* é possível mudar dinamicamente a cena através de linguagens de programação como o JavaScript e o Java.

Pode-se considerar que o X3D estabelece duas classes de entidades: os objetos sensoriais, que envolvem características visuais, sonoras, etc., e objetos comportamentais que incluem cliques de mouse, temporizadores, etc. As ligações entre esses objetos, denominadas rotas, representam o fluxo de propagação de eventos. Esses sistemas de entidades assemelham-se a um grafo direcionado acíclico, sendo chamados de grafo de cena (o conjunto de objetos) e grafo de comportamentos (o conjunto de rotas)(WEB 3D, 2009).

O X3D emprega uma arquitetura modular para promover uma maior expansibilidade e flexibilidade. Ele introduz o conceito de perfis, uma coleção predefinida de componentes comumente encontrados em certos domínios de aplicações, plataformas, entre outros. O VRML97 requer um suporte completo das suas funcionalidades para estar em conformidade. Já o X3D permite vários graus de suporte do padrão para suprir uma variedade de necessidades (WEB 3D, 2009).

Para que a visualização de arquivos X3D seja possível em um browser se faz necessária à instalação de um *plug-in* específico. O Xj3D é uma ferramenta Java, com o código aberto, criada para a visualização e a manipulação de conteúdos escritos em VRML e X3D. Elaborada pelo grupo de trabalho da Web3D que idealizou o X3D, ela foi desenvolvida inicialmente como carregador de arquivos X3D para a API Java 3D. Contudo, o seu grau de desenvolvimento a tornou a principal ferramenta usada pelos grupos de teste da especificação X3D. A sua adoção pela comunidade X3D se deve ao fato do Xj3D ter ganhado algumas funcionalidades importantes, como, por exemplo, permitir o uso de diferentes vias para a renderização de cenas, tais como Java 3D, DirectX e OpenGL; e possibilitar o uso da interface de programação externa do X3D, a SAI - *Scene Access Interface*. Fundamentalmente, a SAI permite o uso de toda a estrutura da linguagem Java. Tal estrutura possui como grande trunfo um interpretador (*Java Virtual Machine - JVM*) que é instalado por padrão em muitos sistemas operacionais, ou pode ser implantado nos sistemas que não o possuem

automaticamente, sem que para isso seja necessário qualquer esforço adicional do usuário (BRUTZMAN, et al., 2007).

Para possibilitar o acesso às funcionalidades da linguagem Java, existe um nó denominado *script* dentro da estrutura do X3D, que possibilita efetuar uma ligação das cenas do AV com as classes Java. Em suma, a própria cena X3D ao ser carregada em um *browser* X3D invoca as classes Java, nas quais existe o código que efetuará as alterações pretendidas na cena.

Devido às características funcionais e possibilidades de desenvolvimento e aplicação de sistemas X3D e por: apresentarem portabilidade, ou seja, serem multiplataforma; contarem com a distribuição gratuita de sistemas que permitem executá-los; e disponibilizarem seus códigos abertos; optou-se por usar o padrão destas linguagens no projeto do simulador.

Optou-se ainda por utilizar a ferramenta Xj3D para a manipulação e visualização dos objetos virtuais destes cenários. O Xj3D foi escolhido neste projeto principalmente por fornecer o suporte dos recursos Java para a estrutura X3D e por já ter a sua utilização dentro do grupo de pesquisa, detalhes desta ferramenta encontram-se no apêndice A, na Figura 13 é ilustrada a arquitetura do visualizador Xj3D (X3D Browser). Nela observa-se que troca de informações entre o visualizador e uma aplicação externa ocorre através do *Scene Access Interface* (SAI) definido pela norma ISO/IEC 19775-2:2004, que é a API usada para fazer esta integração.

Existem duas formas de utilizar o SAI consistindo no acesso interno e externo (SILVA NETTO, 2010).

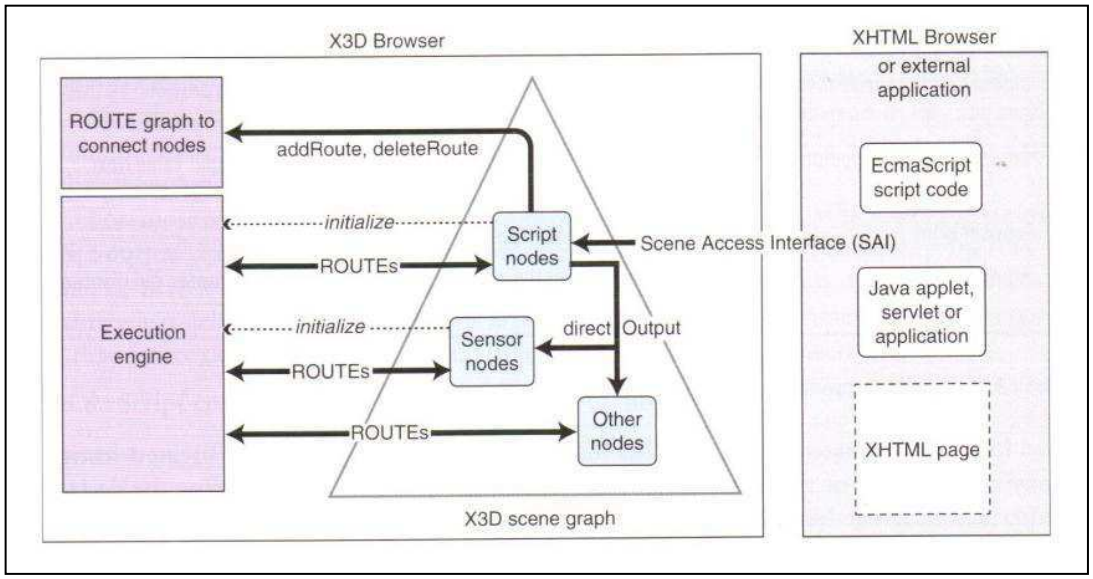

Figura 13 - Arquitetura do visualizador Xj3D (Fonte: BRUTZMAN, et al., 2007).

4.1.2 Ferramentas Java de distribuição

Java é uma linguagem de programação orientada a objeto, o início do seu desenvolvimento se deu em meados da década de 90 pelo programador James Gosling, na empresa *Sun Microsystems*. Diferentemente das linguagens convencionais, que são compiladas para código nativo, a linguagem Java é compilada para um "*bytecode*" que é interpretado por uma máquina virtual (JVM – *Java Virtual Machine*) (DEITEL, 2005). Esta estratégia de compilação seguida por interpretação é que garante a Java a sua portabilidade. A linguagem Java possui recursos consagrados para distribuição de dados via rede, como, por exemplo, o RMI e o *Socket*.

O RMI – *Remote Method Invocation* (Invocação de Métodos Remotos) é uma das abordagens da plataforma Java para prover as funcionalidades de uma plataforma de objetos distribuídos. O RMI permite ao programador invocar métodos de objetos remotos, ou seja, que estão alojados em JVMs distintas, de uma forma muito semelhante às invocações a objetos locais.

O protocolo utilizado para a comunicação entre os objetos pelo RMI é o JRMP – *Java Remote Method Protocol*. Este protocolo baseia-se na serialização de objetos Java, isto é, os objetos utilizados como parâmetros podem ser transportados como um fluxo de dados, por meio de uma sessão TCP – *Transport Control Protocol*.

O processo de comunicação envolve, primeiramente, a localização do objeto, que é fornecida através de uma string que contém o nome do objeto e o endereço da máquina que o abriga. O uso de um protocolo padrão facilita a implementação de RMI, porém pode provocar uma perda de desempenho, devido às mensagens de controle enviadas pelo TCP para garantir a entrega das mensagens.

O *Socket* é um elemento de software que provê uma interface com o intuito de facilitar o desenvolvimento de aplicações que envolvem a comunicação entre dois ou mais computadores interligados por uma rede. Um *Socket* representa um ponto de possível conexão entre duas máquinas onde tal conexão funciona como um duto de dados, isto é, um canal que permite a transmissão de dados de uma máquina para a outra de forma bidirecional possibilitando o envio e recebimento simultâneo de dados. Isto significa que cada *Socket* possui um canal de entrada e outro de saída, sendo que, o que é enviado pelo canal de saída de uma máquina é recebido pelo canal de entrada da outra máquina e vice-versa.

Os *Sockets* têm dois modos principais de operação: o modo baseado em conexões e o modo sem conexão. O modo a ser utilizado é determinado pelas necessidades de um aplicativo. Se a confiabilidade é importante, então a operação baseada em conexões é a melhor opção. Os servidores de arquivos precisam fazer todos os seus dados chegarem corretamente e em seqüência. Garantir a seqüência e a correção dos dados exige processamento extra e utilização de mais memória; esse overhead pode reduzir o tempo de resposta de um servidor.

A operação baseada em conexões emprega o protocolo TCP. Um Socket nesse modo de operação precisa se conectar ao destino antes de transmitir os dados. Uma vez conectados, os *Sockets* são acessados pelo uso de uma interface de fluxos: abertura-leitura-escrita-fechamento. Tudo que é enviado por um *Socket* é recebido pela outra extremidade da conexão. A operação baseada em conexões é menos eficiente do que a operação sem conexão, mas é garantida (HOPSON, et al., 1997) (HOPSON, et al., 1997).

Existe ainda a CORBA (Common Object Request Broker Architecture), criada pela OMG (Object Management Group), que é uma especificação-padrão de arquitetura de sistemas distribuídos que conquistou ampla aceitação. CORBA é um padrão aberto elaborado para habilitar a interoperação entre programas em sistemas homogêneos e heterogêneos. Similar à RMI, CORBA suporta objetos como parâmetros ou valores de retorno em procedimentos remotos durante a comunicação entre processos. Entretanto, diferentemente da RMI (que é baseada em Java), CORBA é independente de linguagem e de sistema, o que significa que aplicações escritas em linguagens de programação distintas e em sistemas operacionais distintos operam entre si por meio de acesso a um núcleo comum da arquitetura CORBA (KUROSE, et al., 2006).

A confiabilidade em um ambiente virtual distribuído é a garantia de que os dados enviados de um computador que esteja conectado a este ambiente cheguem aos demais computadores de forma correta, evitando assim a necessidade de reenviar pacotes periodicamente.

Se um ambiente virtual utilizasse protocolos de comunicação orientados à conexão, como o TCP que envia a confirmação do recebimento de mensagens, ele asseguraria a sua confiabilidade, porém, dependendo do modo com que é aplicado, ele nem sempre é a opção que garantirá a melhor qualidade do sistema distribuído. Segundo (MACEDONIA, et al., 1997) o atraso causado pelo envio da confirmação de recebimento dos dados prejudica o desempenho de simulações em tempo real, e dependendo da estratégia de desenvolvimento do sistema seria melhor usar o protocolo não orientado a conexão UDP. É um protocolo onde a

troca de mensagens não é confiável, pois não garante a entrega dos *datagramas*, a eliminação de duplicados, e a ordem de entrega. Contudo, como não há um controle de recebimento e de erro, a entrega das mensagens é agilizada, sendo esse um fator essencial em ambientes com interações em tempo real.

Porém, outros fatores podem influenciar na escolha do protocolo a ser usado. Por exemplo, o *Socket* TCP normalmente é utilizado em sistemas síncronos, e o UDP para assíncronos. Este fato ocorre devido exatamente às estruturas de trabalho dos *Sockets*, o TCP garante a entrega dos dados de forma ordenada e isso garante que os sistemas comunicantes funcionem de maneira compassada. O *Socket* UDP por sua vez, em virtude da possível perda de informações propicia fortemente o assincronismo dos sistemas. Uma estratégia comumente utilizada para sincronizar sistemas UDP é a aplicação de *Threads*. Deste modo, pode-se concluir que programadores que optaram pelo uso de *Sockets* para desenvolver sistemas que devem trabalhar sincronicamente, o ideal é utilizar *Sockets* TPC ou UDP sincronizados por *Threads*.

A arquitetura de distribuição projetada para o Simulador exige que os sistemas comunicantes trabalhem sincronicamente. Neste contexto, optou-se por utilizar o recurso *Socket* TCP da linguagem Java para desenvolver o sistema de distribuição do Simulador.

Para melhorar o desempenho do sistema realiza-se a conexão e o fechamento dos *Sockets* apenas uma vez, quando se inicia e termina a comunicação dos sistemas, respectivamente. Para estabelecer a conexão, inicialmente, o servidor começa a "escutar" uma determinada porta, o cliente tenta a conexão nesta porta, o servidor aceita a conexão e envia uma mensagem de sucesso para o cliente. Após este processo a conexão estará estabelecida e o fluxo de dados pode iniciar. O processo de conexão é relativamente lento, por isso faz-se necessário que seja feito uma única vez. Estabelecida à conexão, o fluxo de dados é liberado, sendo ele um processo rápido e seguro.

4.1.3 Banco de dados

O banco de dados (BD) é um componente fundamental no projeto do simulador, armazenando informações sobre a configuração das instalações simuladas, a ocorrência de eventos associados a cenários, e o registro do histórico de todas as interações do operador com o ambiente virtual durante um treinamento.

A partir do registro de uma sessão de interação no BD, será possível a análise dos erros dos operadores durante o treinamento, apoiando assim a proposição de novos cenários de treinamentos. Ao salvar um estado atual do ambiente, ao final de uma sessão de simulação e, permitir sua recuperação em uma próxima interação, o sistema oferece maior flexibilidade na realização do treinamento.

Um banco de dados é mantido e acessado por meio de um Sistema Gerenciador de Banco de Dados (SGBD). O SGBD selecionado para a versão atual do simulador foi o SQL Server 2008. O SQL Server foi criado pela Microsoft® em parceria com a Sybase em 1988 e inserido como produto complementar do Windows NT, a versão 2008 fornece uma plataforma de dados segura, confiável, escalonável e gerenciável com menor indisponibilidade de aplicações (MALCOLM, 2008). Outros fatores influenciaram na escolha deste SGBD, entre eles, a fácil integração com sistemas supervisórios e a funcionalidade XML, fundamental para o projeto.

O SQL Server possui um tipo de dado XML nativo que permite a validação perante as declarações na coleção de esquemas associada quando valores são inseridos ou atualizados no BD, o que possibilita impor regras sobre a estrutura de dados XML por motivos de conformidade ou compatibilidade.

O tipo de dado XML também fornece uma variedade de métodos, que podem ser usados para consultar e manipular os dados XML em uma instância.

A primeira versão do banco de dados do simulador foi desenvolvida por (SOUSA, 2008) e não foi integrada ao projeto do Simulador. O modelo de dados inicial abrangeu o armazenamento de informações sobre os usuários do simulador (treinandos e tutores) e a estruturação e resultados dos treinamentos.

O modelo entidade-relacionamento (MER) do banco de dados (BD) proposto originalmente é ilustrado na Figura 14.

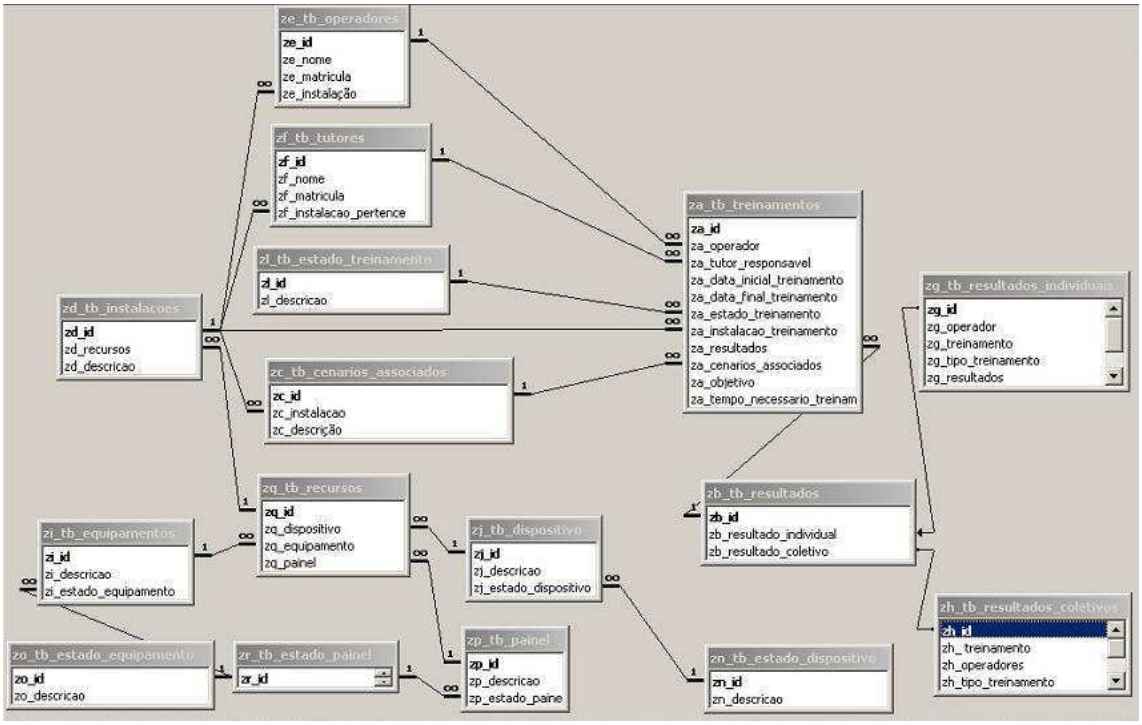

Figura 14 - MER do banco de dados do Simulador (SOUSA 2008).

Este modelo de dados atendia a maioria dos requisitos do módulo do treinando do Simulador, contudo deixava uma lacuna no que se refere ao módulo tutor. O projeto do módulo tutor requer o armazenamento de um conjunto mais amplo de informações. Dessa forma foi proposta a inclusão no modelo de dados de informações adicionais, tais como dados sobre cenários de treinamento, com o estado inicial, estados intermediários e estado final de uma subestação elétrica, durante um treinamento. Sendo a extensão e modificações no modelo de dados original uma das etapas deste projeto.

Na Figura 15 é apresentado o MER do banco de dados atual do projeto do simulador, conceitualmente é dividido em seis setores:

- Treinamento e resultados;
- Pessoal;
- Instalações;
- Eventos;
- Cenários e estados da subestação;
- Dispositivos.

Apesar da dificuldade de visualizar o MER em um só plano, para fins de contextualização dos setores, o modelo completo é ilustrado.

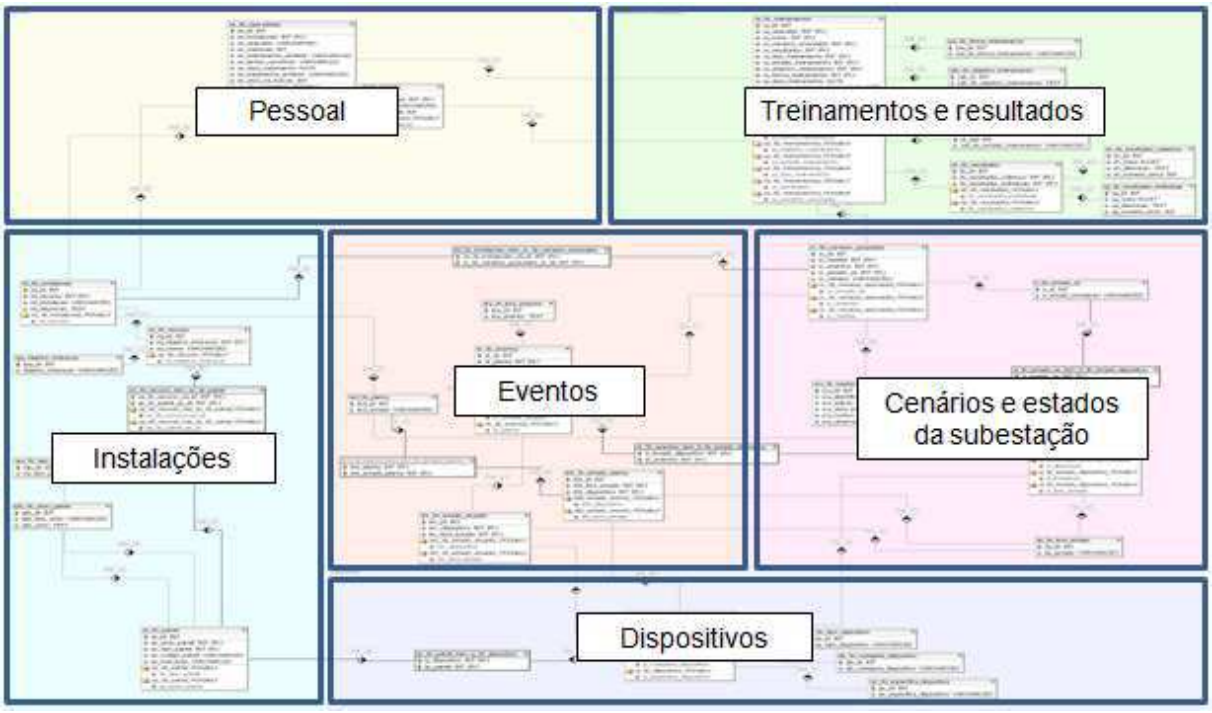

Figura 15 - MER do banco de dados atual do simulador.

Detalhes da modelagem de dados encontram-se no Apêndice A.

4.1.4 Comunicação da RV com o BD

Os SGBD contam com *drivers* para gerenciar o acesso, manipulação e organização de seus dados. Assim um programa cliente pode se conectar a diversos servidores SGBDs e enviar requisições de consultas e transações usando a Interface para Programação de Aplicações (API).

Quando o acesso ao banco de dados é solicitado pelo usuário, através do Simulador, é estabelecida uma conexão com o SGBD instalado no servidor. Uma vez criada à conexão, o programa cliente pode se comunicar com o SGBD. Após o processamento de uma transação, o resultado é fornecido pelo servidor de dados, atualizando um conjunto de tabelas. Os resultados das transações são enviados para o programa cliente, que pode processá-los ou apenas exibi-los.

A linguagem Java (JAVA, 2010) permite o acesso a bancos de dados relacionais através das funcionalidades definidas no pacote JDBC. O JDBC é uma API (Interface para Programação de Aplicações) para execução e manipulação de resultados de consultas SQL através de Java.

Para desenvolver uma aplicação com Java e bancos de dados relacionais, é necessário:

- o pacote JDBC (padrão na distribuição da plataforma de desenvolvimento Java, desde sua versão 1.1);
- acesso ao servidor de banco de dados relacional, ou seja, ao sistema gerenciador de banco de dados;
- um *driver* JDBC adequado ao SGBD utilizado.

Uma vez que esses recursos estejam disponíveis, a aplicação Java obtém o acesso ao banco de dados relacional, executando os seguintes passos:

1. Habilitando o *driver* JDBC a partir da aplicação cliente;
2. Estabelecendo uma conexão entre a aplicação cliente e o servidor do banco de dados;
3. Montando e executando a consulta SQL desejada;
4. Processando no cliente o resultado da consulta.

Para exemplificar a comunicação do AV com o motor de simulação e com o BD foi desenvolvido um diagrama de sequência de mensagens (MSC – *Message Sequence Charts*) utilizando a ferramenta JUDE (JUDE, 2010).

O diagrama, ilustrado na Figura 16, representa a atuação de um operador (abertura ou fechamento) sobre um dispositivo do Ambiente Virtual (AV). Ao atuar sobre um dos disjuntores é enviada uma mensagem para as redes CPN e uma mensagem para o BD, informando sobre a abertura ou fechamento do dispositivo. As redes CPN recebem a informação processam e enviam uma mensagem de volta para o AV, o qual encaminha a mensagem para o BD e atualiza a interface.

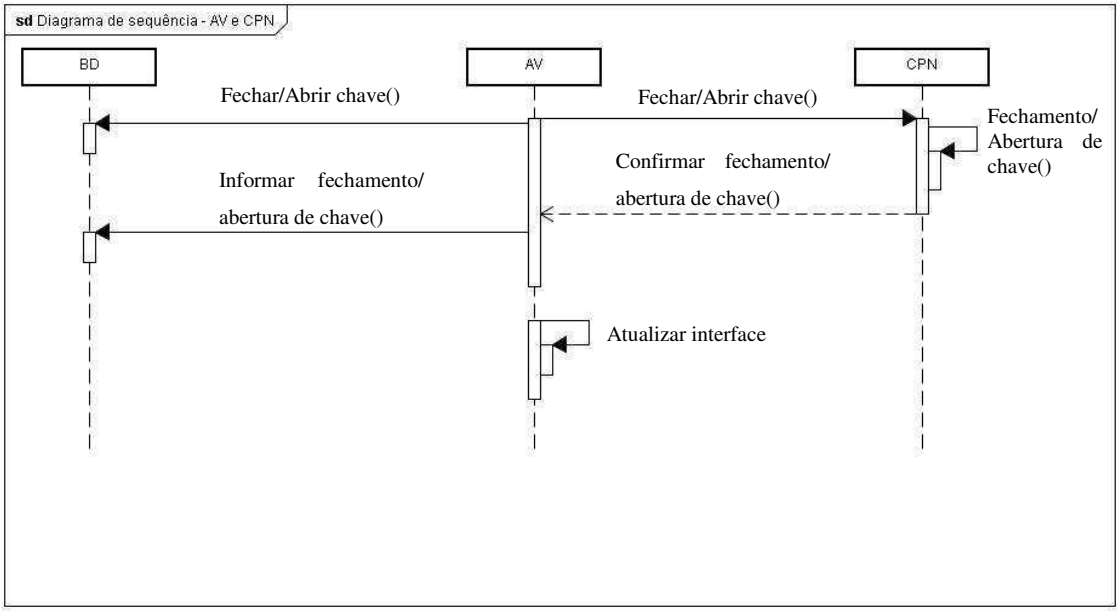

Figura 16 - MSC da abertura/fechamento de um disjuntor.

4.2 *Implementação do módulo Tutor*

Nesta seção são abordados os detalhes de desenvolvimento do módulo tutor. Primeiramente é apresentada a interface do tutor e seus elementos constituintes (Menu, AV, painel de configuração, etc.). Posteriormente é apresentada a interface do treinando, o conjunto de classes desenvolvidas e, por fim, é discutida a implementação do sistema de distribuição.

4.2.1 *Interface do tutor*

A interface do tutor foi desenvolvida visando facilitar a configuração do AV (criação e edição de cenários) e o acompanhamento e registro das seções de treinamento, inclusive podendo disparar eventos durante a simulação. Atualmente é composta por uma barra de menus, uma representação do AV e um painel de configuração formado por um conjunto de abas que acessam funcionalidades diversas do sistema, como ilustra a Figura 17.

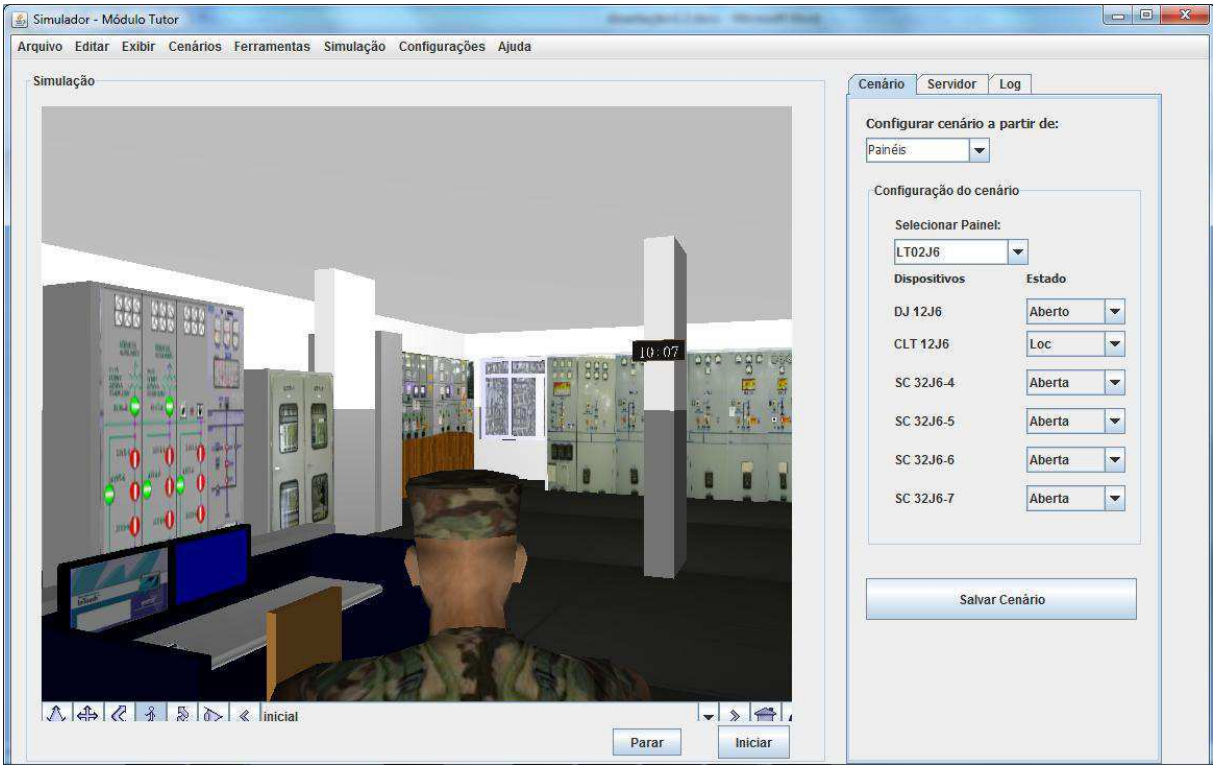

Figura 17 - Interface do módulo tutor.

A partir do menu é possível acessar a todas as funções implementadas, através da representação do AV o tutor pode navegar no ambiente para configurar o estado inicial dos objetos presentes, já o painel de configuração fornecer um atalho aos recursos de maior importância, como o estabelecimento da conexão dos usuários ao sistema, outra forma de configuração dos estados dos dispositivos dos painéis do AV a partir da interface, salvamento de cenários e acompanhamento das ações do usuário a partir de uma tela de *log*.

O estabelecimento da conexão entre os usuários pode ser realizado a partir da aba "*Servidor*", o tutor pode permitir a conexão de treinandos ao seu AV clicando no botão "*Iniciar*" presente na interface, detalhes deste processo são apresentados em seções posteriores deste trabalho.

A aba "*Cenário*" fornece um recurso adicional para manipulação dos dispositivos presentes nos painéis do AV, através dela é possível selecionar um painel de interesse, como está destacado na figura abaixo, verificando e alterando os estados dos objetos desejados possibilitando ainda o seu salvamento.

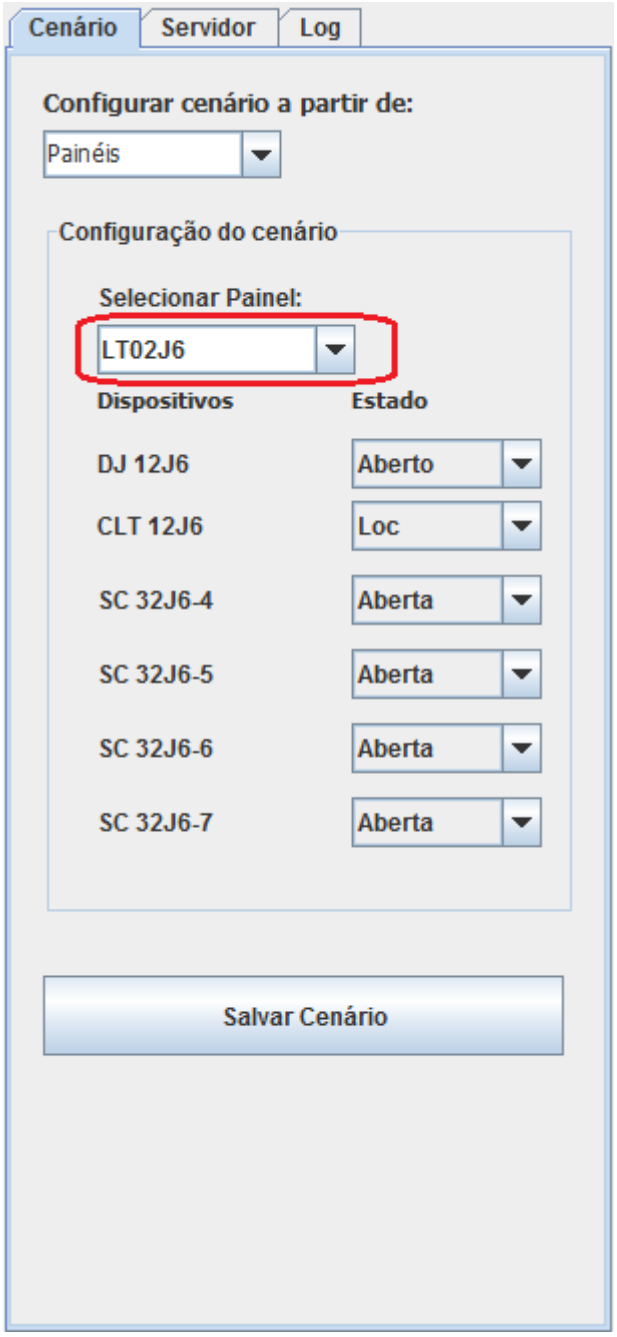

Figura 18- Aba "*Cenário*".

Todas as ações dos usuários em uma seção de treinamento são registradas no BD para sua posterior análise, com o intuito de permitir ao tutor um acompanhamento em tempo de execução do treinamento, um resumo destas informações (momento da atuação, dispositivo e estado atual) é apresentado na tabela presente na aba de "*Log*" como pode ser observado na figura a seguir, vale ressaltar que as informações são ordenadas pelo momento da ação do usuário.

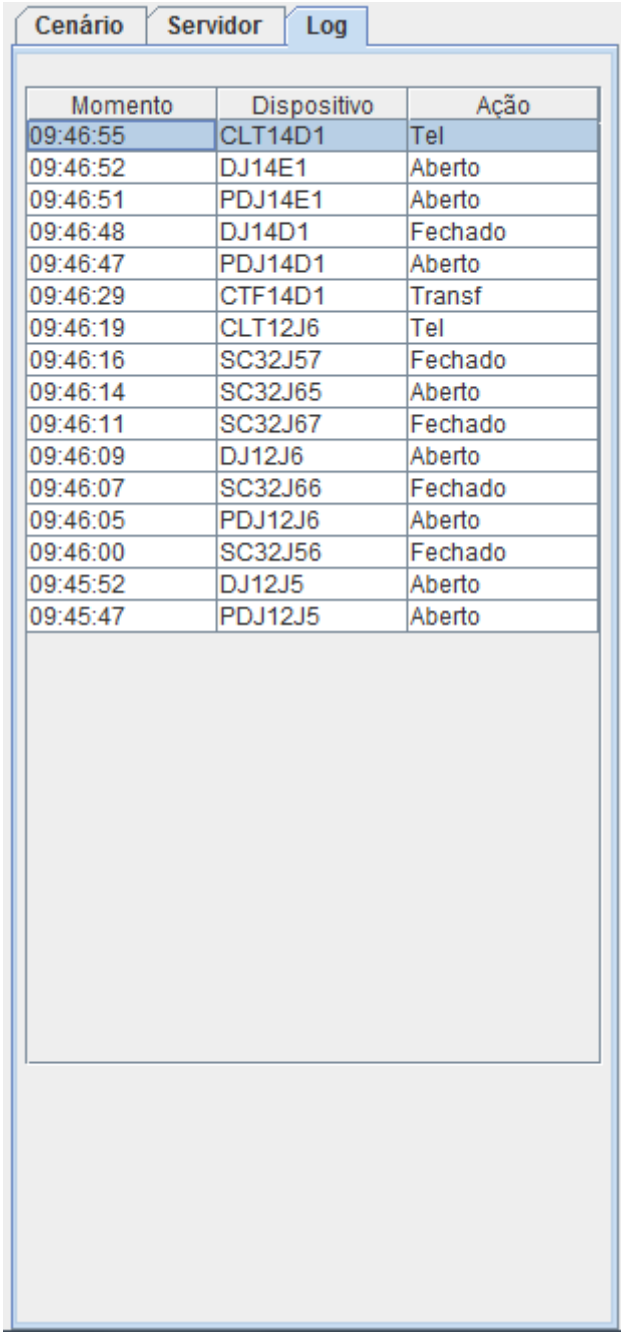

Figura 19 - Aba de "Log".

4.2.2 Interface do treinando

A interface do treinando é composta basicamente por uma barra de menus e uma representação do AV, a Figura 20 apresenta a interface criada. Nesse contexto é dado destaque ao AV com intuito de envolver o treinando na situação simulada para que o aprendizado adquirido em um treinamento possa ser reproduzido em um ambiente real.

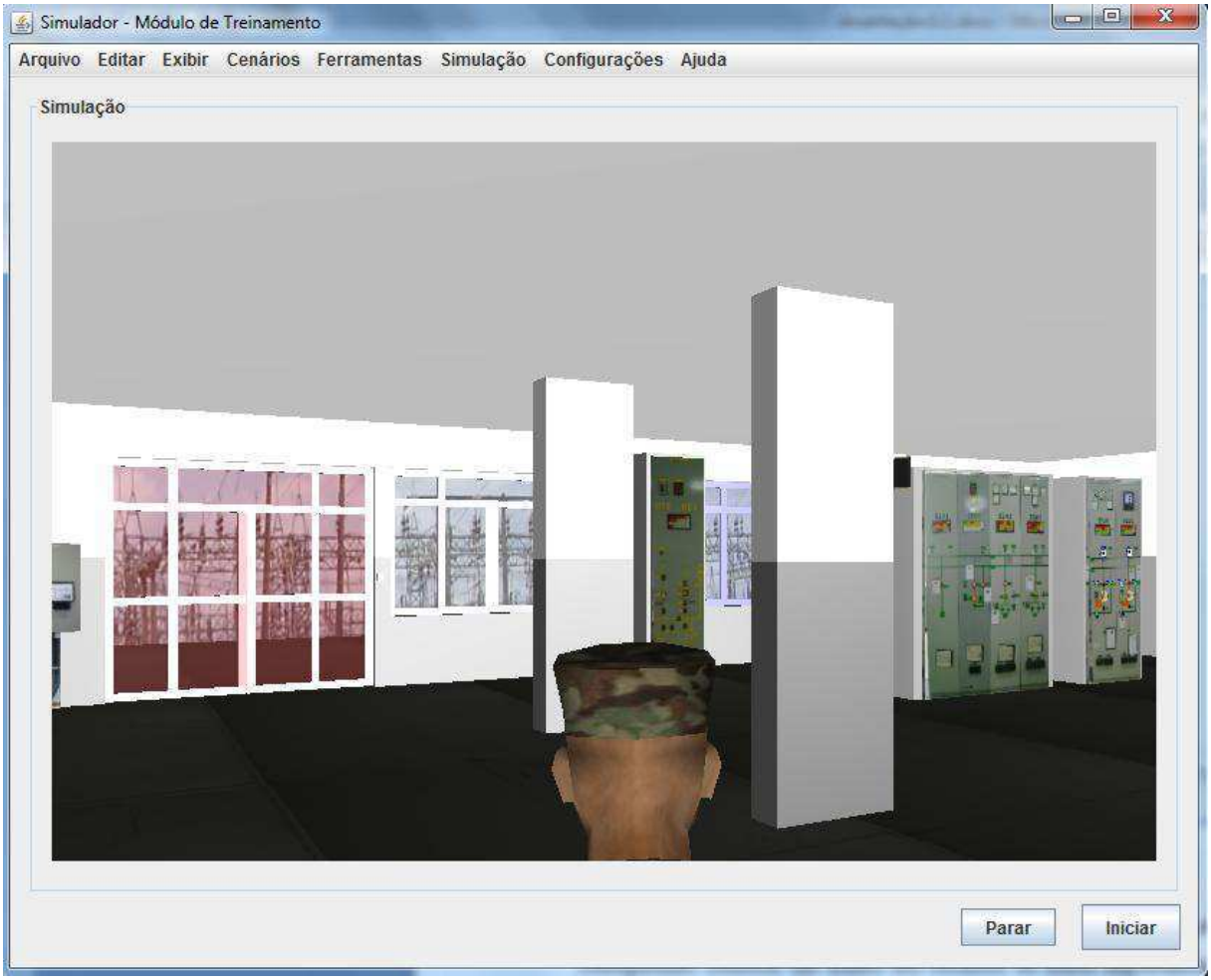

Figura 20 - Interface do treinando.

4.2.3 Desenvolvimento das classes

No atual estágio de desenvolvimento do Projeto do Simulador temos desenvolvidas as classes presentes na Figura 21.

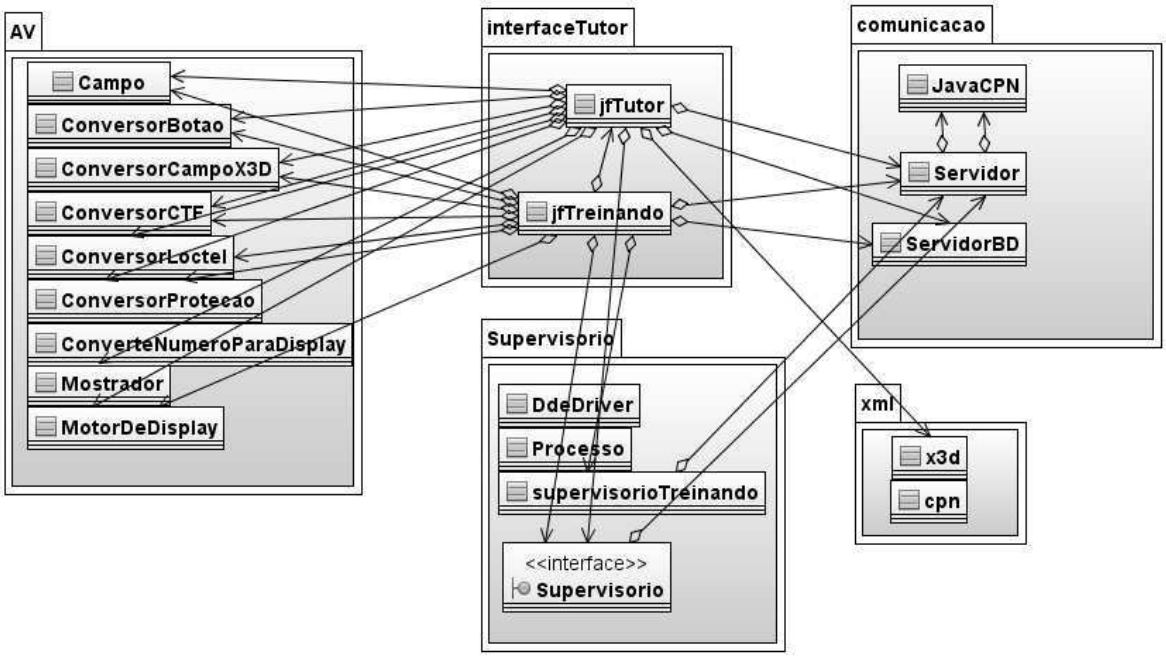

Figura 21 - Diagrama de classe do simulador.

A classe *AV* contém métodos com funções de conversão dos valores recebidos e enviados ao motor de simulação. Ela contém também a função que monitora as ações do mundo virtual. A classe de *comunicacao* contém os métodos para comunicação com banco de dados, com o AV e com as CPNs, além dos métodos de comunicação entre os usuários do sistema. A classe *interfaceTutor* contém as interfaces do tutor e do treinando. Dentro da classe *supervisorio* encontram-se os métodos *processo* cuja finalidade é abrir um processo externo qualquer e o método *supervisorio* que abre um processo com um sistema supervisório especifico.

4.3 Implementação do sistema de distribuição do Simulador

Nesta seção serão abordados os detalhes de desenvolvimento do sistema de distribuição do simulador. Em suma, são apresentados os componentes que possibilitam a distribuição e discutidas as suas respectivas funcionalidades junto ao sistema. Apresentam-se ainda trechos de códigos dos componentes criados.

4.3.1 Cliente e Servidor Java TCP

Os *sockets*, da linguagem Java, são criados por um conjunto de classes do pacote *java.net*. Na conexão TCP, o cliente e o servidor são oriundos de classes diferentes. O servidor usa a classe *ServerSocket* para "escutar" uma porta de rede da máquina a espera de requisições de conexão, sendo assim, o seu construtor possui como um de seus parâmetros o número desta porta.

O cliente, por sua vez, deve conhecer em qual máquina o servidor está sendo executado e a porta a qual ele está escutando. Neste contexto, o cliente utiliza a classe *Socket* para requisitar conexão a um servidor específico e então transmitir dados. O seu construtor tem como parâmetros o endereço IP do servidor e a porta escutada.

O servidor usa o método *accept()* para bloquear a execução e aguardar até que um cliente requisite conexão. Uma vez recebida à conexão o servidor responde ao cliente, e então é criado um fluxo de conexão.

No modelo de fluxos (*streams*) dos *sockets* utilizam-se dois fluxos: um de entrada e outro de saída, criados a partir das classes *InputStream* e *OutputStream*, respectivamente. Um processo envia dados a outro quando escreve em um fluxo de saída associado a um *socket*. E um processo recebe dados escritos por outro processo lendo um fluxo de entrada associado a um socket.

Para encerrar a conexão tanto cliente quanto servidor utilizam o método *close()*. A Figura 22 ilustra os processos de estabelecimento de conexão, troca de informação e encerramento de conexão em *sockets* cliente e servidor.

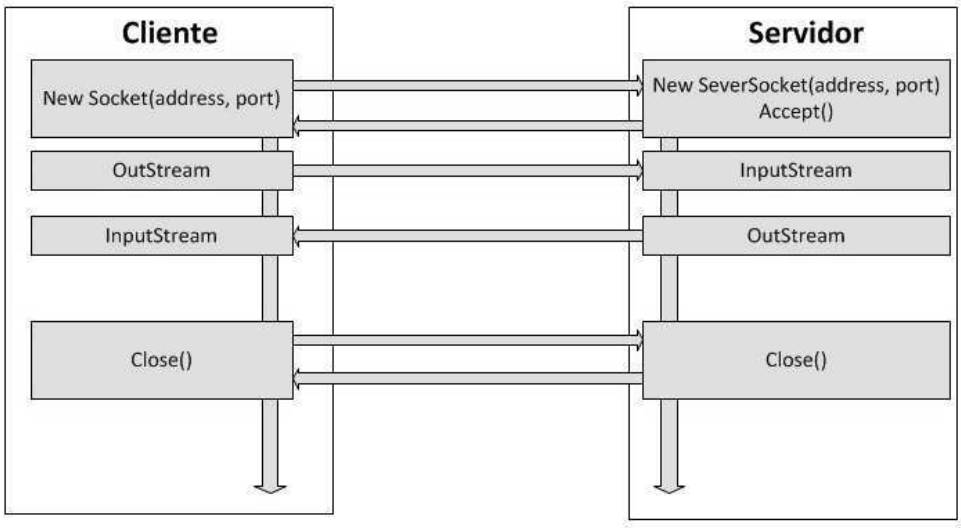

Figura 22 - Sockets cliente e Servidor.

Considerando o funcionamento dos *sockets* TCP implementou-se os serviços do cliente e servidor para o simulador. Para tanto, foram desenvolvidos dois métodos dentro da classe de comunicação que são responsáveis pelas etapas de estabelecimento e encerramento da conexão dos clientes (treinandos) e o servidor (tutor).

A Figura 23 apresenta a aba "*Servidor*" presente na interface do tutor.

Figura 23 - Interface de estabelecimento e encerramento de conexão do servidor.

Por meio da interface do tutor é possível executar o servidor, clicando no botão "*Iniciar*". Esta ação coloca o servidor em espera por uma requisição de conexão de um cliente, Figura 24. Pode-se observar que o servidor deve ser executado antes do cliente para aguardar a esta requisição.

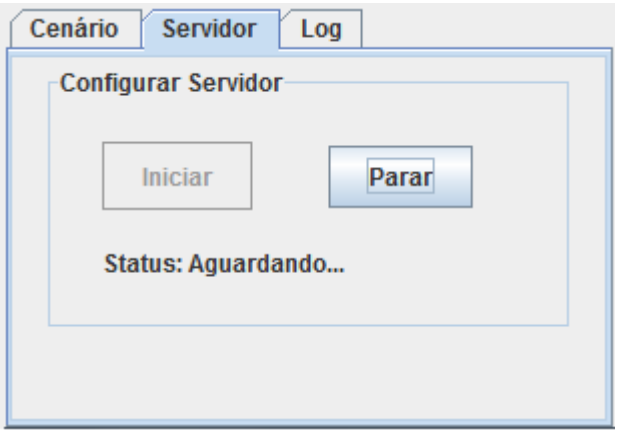

Figura 24 - Servidor em espera por uma requisição de conexão de um cliente.

A interface apresenta ainda uma barra de status, que oferece as informações: "*Aguardando...*", que indica a espera de uma ação do usuário ou de uma requisição de

conexão de um cliente; "*Conectado*", aponta que o servidor está conectado a algum cliente; "*Desconectado*", indica que a conexão foi interrompida.

Caso o tutor queira encerrar o servidor basta pressionar o botão "*Parar"*. A Figura 25 apresenta o trecho de código que implementa o método *iniciarServidor*.

```
public void iniciarServidor() {
    try {
        serverSocket = new ServerSocket(3000);
        connection = serverSocket.accept();
        input = new ObjectInputStream(connection.getInputStream());
        output = new ObjectOutputStream(connection.getOutputStream());
        output.flush();

        lStatus1.setText("Conectado");
    } catch (IOException ioException) {
        ioException.printStackTrace();
    }
}
```

Figura 25 - Trecho de código do método *iniciarServidor*.

O método *iniciarServirdor* deve ser executado antes da solicitação de conexão dos clientes. Em termos de implementação, o desenvolvedor pode optar por chamá-la em uma aplicação isolada ou dentro de um módulo da própria aplicação, como o caso deste trabalho. Sendo que, logo após a sua chamada, em ambos os casos, deve-se implementar os respectivos recursos para promover a troca de informações.

A interface do cliente possui dentro do menu "*Configurações*" a opção "*Conectar ao Servidor*". Ao selecionar esta opção o cliente requisita conexão ao servidor, que responde a requisição, e deste modo à conexão é estabelecida e a troca de informações pode ser iniciada. Caso o usuário queira se desconectar, basta fechar a janela.

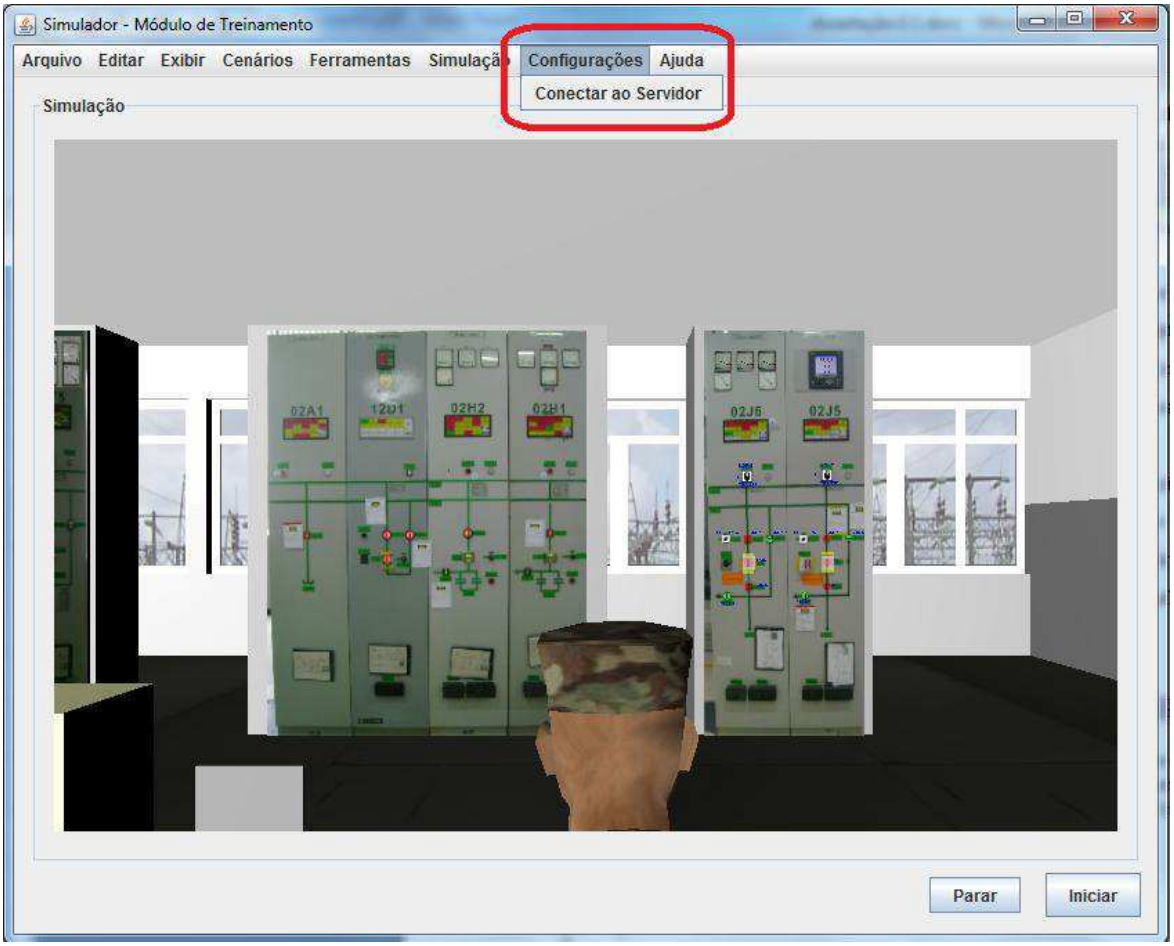

Figura 26 - Interface de conexão do cliente.

Diferentemente método *iniciarServirdor* o método de conexão do cliente obrigatoriamente deve ser associada a uma aplicação do projeto do simulador, no caso a interface do treinando. Os métodos *iniciarServidor* e *conectarServidor* são análogos mudando apenas alguns componentes Java de interface e os componentes de conexão apresentados na figura abaixo.

```
    private void mConectarActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        clientSocket = new Socket(InetAddress.getByName(ipServer), 3000);
        output = new ObjectOutputStream(clientSocket.getOutputStream());
        input = new ObjectInputStream(clientSocket.getInputStream());
        output.flush();

    } catch (IOException ioException) {
        ioException.printStackTrace();
    }
    tutor.lStatus.setText("Status: Conectado");

}
```

Figura 27 - Trecho de código do método conectarServidor

Por meio da criação dos métodos apresentados, foi possível construir o sistema de distribuição do simulador.

Capítulo 5 Validação

Com o objetivo de validar o trabalho que foi desenvolvido, foram realizados testes com usuários que desempenharam os papéis de um operador de sistemas elétricos iniciante e de um tutor conduzindo uma sessão de treinamento.

No cenário de teste planejado, o usuário no papel de tutor deve configurar o ambiente do simulador para realização do treinamento, gerar a ocorrência de um alarme e acompanhar as ações do treinando. Para isso, ele poderá interagir com o módulo supervisório, fruto de outro trabalho em andamento no LIHM e com os elementos presentes na interface do tutor (barra de menus, representação do AV e painel de configuração).

O usuário no papel de operador deverá simular a realização de uma manobra no sistema elétrico que foi representado no simulador, interagindo com a representação do AV presente na interface do treinando e com o supervisório.

No que segue, será descrito o experimento que foi realizado, com a configuração do ambiente de teste, o perfil dos usuários participantes e a manobra que deveriam executar. Em seguida, apresentam-se os critérios adotados para avaliação, os dados coletados e a análise dos resultados.

5.1 Planejamento do experimento

O principal objetivo na realização desse experimento é validar o módulo tutor que foi desenvolvido e avaliar sua integração com os demais módulos do simulador. Como objetivos específicos pretende-se avaliar se:

- os recursos e funcionalidades oferecidos pelo módulo tutor são suficientes para realização de treinamentos;
- as interfaces do tutor e do treinando são adequadas;
- o tempo de resposta da interface é suficiente e tolerável pelo usuário;
- os estados dos dispositivos representados no painel de configuração da interface do tutor são coerentes com as correspondentes representações nos painéis do AV;

- existe erro na comunicação entre os mundos virtuais do tutor e do treinando, comunicação com o módulo supervisório, com o motor de simulação e com o banco de dados;
- o grau de dificuldade na configuração de um cenário é baixo;
- a navegação no AV é adequada;
- as informações são apresentadas na tela em uma sequência coerente e agradável ao usuário;

O ensaio de avaliação foi realizado no Laboratório de Interfaces Homem-Máquina (LIHM), que possui a infraestrutura necessária para gravação da sessão de teste em áudio e vídeo para posterior análise. Os usuários, tutor e treinando, realizaram as tarefas do teste em uma sala com câmeras e microfones, permitindo que os avaliadores pudessem acompanhar o experimento de uma sala vizinha (sala de controle), através de monitores.

Na sala de teste também existiam dois computadores, um executando a aplicação do tutor e o outro a do treinando. Antes de iniciarem a execução das tarefas, os usuários foram informados do propósito do experimento, o papel que deveriam desempenhar, os recursos e funcionalidades do simulador. Os avaliadores também demonstraram como navegar no ambiente 3D, como interagir com os objetos virtuais e como iniciar o supervisório.

Foram realizadas duas sessões de teste com três usuários homens, na faixa etária de 25 a 35 anos, com ensino superior completo e com conhecimento na área de sistemas elétricos em nível médio, mas sem nenhuma experiência de operação. Essas características são similares a de um operador iniciante.

Dos três usuários selecionados, apenas o UT1 (usuário de teste 1) tinha experiência no uso da versão atual do simulador. Na Tabela 1, pode ser observado o papel desempenhado pelos usuários em cada sessão de teste.

Tabela 1 - Papel desempenhado pelos usuários em cada sessão de teste.

	Sessão A	Sessão B
UT1	Tutor	Não participou
UT2	Treinando	Tutor
UT3	Não participou	Treinando

UTn – usuário de teste n

5.1.1 Critérios de avaliação

Para atingir os objetivos do experimento, foram adotados alguns critérios de avaliação, os quais estão reunidos na Tabela 2, que apresenta também como os dados são coletados.

Tabela 2 - Papel desempenhado pelos usuários em cada sessão de teste.

Critério de avaliação	Como coletar?
Consistência dos dados	Observação e registro audiovisual
Corretude da informação apresentada	Observação e registro audiovisual
Tempo de resposta da interface	Análise do questionário pós-teste, registro audiovisual e entrevista
Facilidade de aprendizado	Após breve apresentação do simulador (recursos de navegação, interação e acesso ao ambiente supervisório), verificar o desempenho do usuário na tarefa.
Facilidade de uso e navegação	Análise do questionário pós-teste, registro audiovisual e entrevista
Facilidade de compreensão	Análise do questionário pós-teste, registro audiovisual e entrevista

5.2 Apresentação das tarefas de teste

Para orientar os usuários na realização das tarefas, foram produzidos os documentos "Roteiro de tarefa de teste do tutor" e "Roteiro de tarefa de teste do operador", os quais se encontram respectivamente nos apêndices D e E.

A seguir, apresenta-se uma descrição sucinta das tarefas realizadas pelos dois tipos de usuários em que eles interagem com o supervisório.

Usuário Tutor: Configurar o Cenário

O usuário é solicitado a configurar um cenário de treinamento, antes que o operador inicie a execução de uma manobra. O processo de configuração consiste em verificar e

atribuir os estados corretos aos dispositivos presentes no cenário simulado. Nesse caso, o usuário, interagindo com o AV, com o painel de configuração e com supervisório, deve configurar os dispositivos presentes na linha de transmissão LT04V2 de acordo com a Tabela 3.

Tabela 3 - Dispositivos a serem configurados das diversas interfaces.

Nome	Tipo	Estado inicial	Interface usada
DJ12J5	Disjuntor	Fechado	AV
SC32J5-4	Seccionadora	Fechado	AV
SC32J5-5	Seccionadora	Fechado	AV
SC32J5-6	Seccionadora	Aberto	AV
SC32J5-7	Seccionadora	Aberto	AV
CLT12J5	LocTel	Loc	AV
DJ12J6	Disjuntor	Fechado	Painel de configuração
SC32J6-4	Seccionadora	Fechado	Painel de configuração
SC32J6-5	Seccionadora	Aberto	Painel de configuração
SC32J6-6	Seccionadora	Aberto	Painel de configuração
SC32J6-7	Seccionadora	Aberto	Painel de configuração
CLT12J6	LocTel	Loc	Painel de configuração
14D1	Disjuntor	Aberto	Supervisório
14V2	Disjuntor	Fechado	Supervisório
34V2-4	Seccionadora	Fechado	Supervisório
34V2-5	Seccionadora	Fechado	Supervisório
34V2-6	Seccionadora	Aberto	Supervisório
34V2-7	Seccionadora	Aberto	Supervisório

Na Figura 28 é apresentada a configuração desejada para os dispositivos da linha LT02J6 a partir da interface desejada, o painel de configuração.

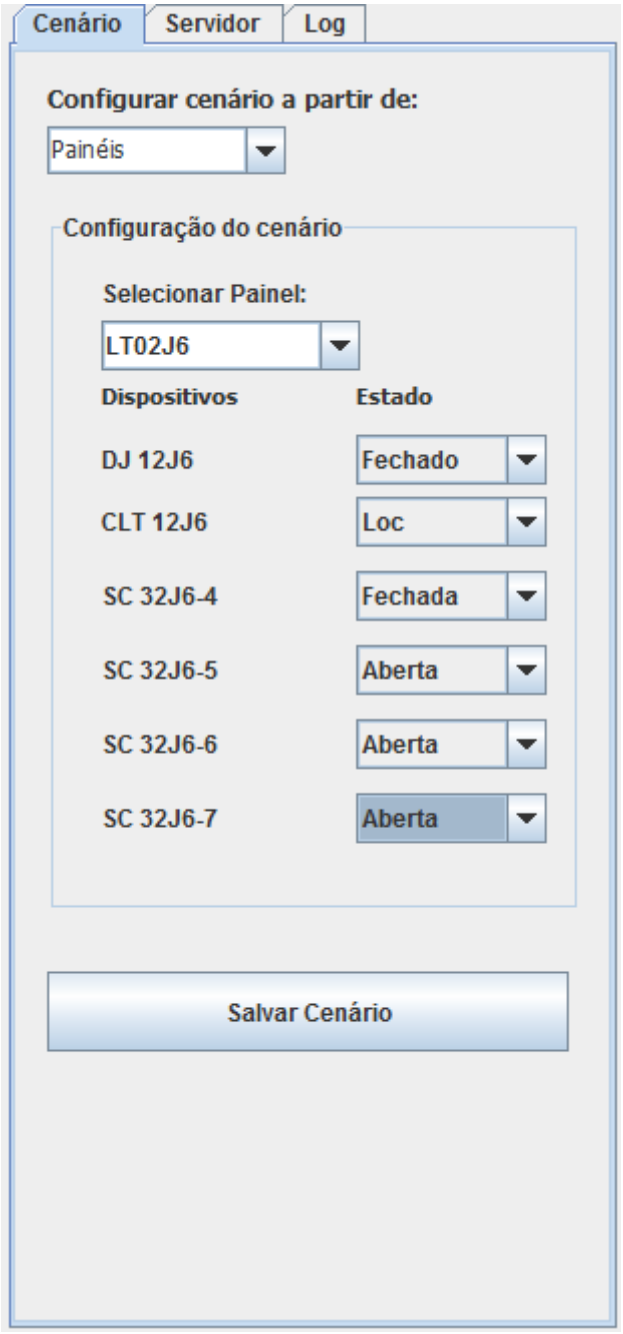

Figura 28 - Linha de transmissão 02J6 com os dispositivos no estado inicial do cenário.

Na Figura 29 é apresentada a configuração desejada para linha LT04V2 através do supervisório. Observe que as chaves seccionadoras são identificadas no diagrama apenas com o último digito de seu nome.

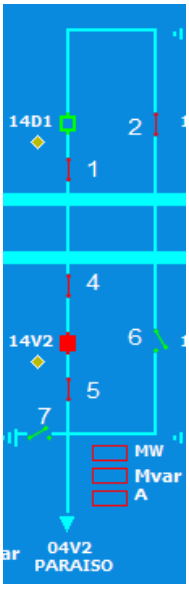

Figura 29 - Linha de transmissão 04V2 com os dispositivos no estado inicial do cenário.

Usuário Tutor: Acompanhar o treinamento

Nessa tarefa, o usuário tutor deve acompanhar as ações do operador durante a execução da manobra. Apesar de não estar explicitamente indicado no roteiro de tarefa (Ver Anexo G), o tutor tem a opção de abrir a aba de "*Log*" para visualizar a mudança de estado dos equipamentos.

Usuário Treinando: Executar o roteiro de manobra

O usuário é solicitado a realizar a manobra de "Liberação do 14V2". Para isso, ele deverá realizar a sequência de passos enumerados a baixo.

1.1. CGD: Receber do responsável solicitação liberação 14V2.
1.2. CGD: Solicitar CROL liberação 14V2.
1.3. CROL: Informar COSR-NE liberação 14V2/CGD.
1.4. CROL: Autorizar CGD liberação 14V2.
1.5. CGD: Confirmar 14D1 aberto.
1.6. CGD: Fechar 34V2-6.
1.7. CGD: Colocar chave 43 -14D1 na posição 'TRANSFERÊNCIA'.
1.8. CGD: Colocar chave 43 -14V2 na posição 'TRANSFERÊNCIA'.
1.9. CGD: Fechar 12D1.
1.10. CGD: Abrir 14V2.
1.11. CGD: Colocar chave CLT-14V2 na posição 'LOC'.

1.12. CGD: Abrir 34V2-4 e 34V2-5.
1.13. CGD: Entregar 14V2 isolado ao responsável.
1.14. CGD: Informar CROL conclusão liberação 14V2.

Usuário Treinando: Alarmes de Eventos

O usuário deve identificar o painel onde um alarme sonoro foi disparado pelo tutor e depois desligá-lo. Para ajudar a identificar o equipamento associado ao alarme/evento, foi sugerido o acesso a tela de alarmes do supervisório.

5.3 Análise dos resultados

Nesse experimento foi avaliada a usabilidade do módulo tutor (interfaces do tutor e do treinando) e a consistência e corretude dos dados nos diversos módulos. Os seguintes pontos foram observados nas sessões de teste e comprovados pelos usuários em um questionário pós-teste:

- o grau de dificuldade nas tarefas de: configuração de cenários, conexão com o operador, acompanhamento de treinamento e geração de eventos;
- percepção sobre o cenário executado (consistente, completo e realista);
- facilidade de navegação;
- preferência pelos modos de configuração do ambiente (via supervisório, via AV e via painel de configuração);
- consistência dos dados representados no ambiente do Simulador.

O UT1, que desempenhou o papel de tutor na primeira sessão de teste, declarou preferir usar o supervisório na tarefa de configuração do cenário de teste em detrimento das outras formas possíveis (AV e painel de configuração). Pois no supervisório em uma única tela o tutor pode atribuir o estado inicial de todos os dispositivos envolvidos no cenário, sem precisar se deslocar pelo ambiente virtual a procura dos painéis de interesse.

Apesar disso, o UT1 demonstrou insatisfação com o tempo necessário para inicializar o supervisório (de 5 a 10 segundos). Porém, quanto ao tempo de resposta da interface, necessário para atualização dos estados dos dispositivos representados na tela, o UT1 considerou adequado.

O UT1 considerou que as tarefas de configuração de cenário e conexão com o operador apresentam um grau de dificuldade baixo, porém o acompanhamento do treinamento e geração de eventos tiveram um grau de dificuldade elevado. Além disso, sugeriu a adição de avatares representando os treinandos no ambiente.

O UT2, que na primeira sessão atuou como tutor e na segunda como operador, classificou como baixo o nível de dificuldade na realização de todas as tarefas, o que demonstra a facilidade de uso do simulador. Porém, considerou ruim a identificação dos dispositivos nos painéis de controle.

Quanto à configuração do cenário, o UT2 declarou preferir o painel de configuração.

O UT3, que atuou no papel de operador na segunda sessão de teste, declarou que o cenário executado foi consistente, completo e bastante realista. O usuário comentou após o experimento: “Me senti realmente envolvido com a situação”.

Na ocorrência do alarme sonoro, o UT3 demonstrou nervosismo e irritação porque não conseguia localizar o painel associado ao evento, onde ele deveria desligar o som do alarme.

Em relação à oferta de recursos para configuração e acompanhamento do treinamento todos os usuários consideraram adequada.

Capítulo 6 Considerações finais

Neste capítulo são apresentadas as conclusões relativas ao desenvolvimento do módulo tutor, bem como sugestões de trabalhos futuros.

7.1 Considerações finais

O desenvolvimento do módulo tutor permitiu criação de um ambiente para gestão e acompanhamento de treinamentos remotos, disponibilizando ao responsável por um treinamento, recursos de construção e edição de cenários, além do registro de todas as ações dos usuários, informações estas fundamentais para avaliação e discussão dos resultados e que servem de base para proposição de outros cenários.

Uma das principais dificuldades encontradas foi realizar a comunicação entre os ambientes virtuais distribuídos, onde se utilizou a ferramenta Java sockets devido a sua performance em relação às demais ferramentas. A arquitetura do sistema e os algoritmos de envio e recebimentos desenvolvidos garantiram que a quantidade de informações transmitida fosse minimizada, tornando o tráfego na rede mais rápido e fluente.

Estes aspectos tecnológicos permitiram estabelecer um conjunto de diretrizes para o desenvolvimento de aplicações distribuídas em X3D de qualidade. Esta linguagem é pouco pesquisada no campo da RV distribuída devido ao fato de ser estática e baseada em eventos, o que dificulta significativamente a transferência e atualização de parâmetros entre ambientes virtuais comunicantes.

Os ambientes distribuídos desenvolvidos (interface do tutor e do treinando) mostraram-se adequados para a efetivação dos testes em relação à treinamentos.

Porém, para obter informações mais completas, concretas e contundentes sobre a qualidade da arquitetura e dos algoritmos de distribuição propostos neste projeto, faz-se necessária à realização de uma avaliação específica quanto aos parâmetros qualitativos de uma distribuição, como por exemplo, desempenho, escalabilidade, tolerância a falhas, segurança, etc.

O uso de linguagens como Java e X3D garantiu a possibilidade de desenvolver e utilizar um sistema portável sem custos de aquisição de *softwares* e *hardwares* adicionais.

7.2 Trabalhos futuros

Como trabalhos futuros, que compõem possíveis propostas para a continuação deste trabalho, pode-se sugerir:

- A implementação da estrutura de distribuição do Simulador por meio de outras tecnologias de comunicação, como por exemplo, Java RMI, CORBA, etc.;
- A realização de testes no sistema visando análises específicas quanto a parâmetros de distribuição, como: o desempenho, a escalabilidade, a conectividade, a segurança, a confiabilidade e a tolerância a falhas. Apenas através destes testes que poderão ser tomadas conclusões efetivas em relação à eficiência da arquitetura de distribuição proposta;
- A criação de avatares individuais para representar outros personagens dentro do AV (outros operadores, tutor, etc).;
- Utilização de técnicas de inteligência artificial para evolução do módulo tutor atual a um sistema tutor inteligente.

REFERÊNCIAS

BARBOSA, J. G. 2010. *Projeto do simulador para treinamento de operadores: Módulo do banco de dados.* Departamento de Engenharia Elétrica, Universidade Federal de Campina Grande. 2010. Trabalho de Conclusão de Curso.

BEZERRA, H., et al. 2007. Sistema simulador para treinamento de proteção e operação de sistemas elétricos. *ICECE'2007 - Conferência Internacional em Educação em Engenharia e Computação.* Anais da Conferência Internacional em Educação em Engenharia e Computação, 2007, p. 90.

BITTENCOURT, J. R. e GIRAFFA, L. M. M. 2004. Desenvolvendo Jogos Computadorizados Multiplataforma com Amphibian. *V Workshop sobre Software Livre.* Anais do V Workshop sobre Software Livre, 2004, pp. 119-122.

BRUTZMAN, D. e DALY, L. 2007. *X3D: Extensible 3D Graphics for Web Authors.* s.l. : Morgan Kaufmann, 2007.

DEITEL. 2005. *JAVA COMO PROGRAMAR.* s.l. : PEARSON, 2005. p. 1152.

DEITEL, H. M., et al. 2003. *XML Como Programar.* [trad.] L. A. SALGADO e E. FURMANKIEWICZ. Porto Alegre : Bookman, 2003.

FOWLER, D. G. 1991. *A Model for Designing Intelligent.* s.l. : Journal of Medical Systems, 1991. Vol. 15.

FREEWRL. 2009. Visualizador de VRML E X3D. *FreeWRL.* [Online] 2009. [Citado em: 10 de Novembro de 2009.] http://freewrl.sourceforge.net/download.html.

FREITAS, R. C., TURNELL, M. F. Q. V. e PERKUSICH, A. 2006a. Mecanismo para visualización y comunicación bidireccional entre modelos Redes de Petri coloreadas y modelos en realidad virtual. *CONFERÊNCIA INTERNACIONAL CONVENCIÓN FIE´06.* 2006a.

—. 2006b. Representando a IHM de uma subestação através de modelos formais e Realidade Virtual. *Simpósio Brasileiro em Sistemas Elétricos.* Anais do Simpósio Brasileiro de Sistemas Elétricos, 2006b.

GOSSWEILER, R., et al. 1994. An Introductory Tutorial for Developing Multi-User Virtual Environments. *Presence.* 1994, Vol. 3, 4, pp. 255-264.

HOPSON, K. C. e INGRAM, S. E. 1997. *Desenvolvendo Applets com java.* Rio de Janeiro : Campus, 1997. pp. 335-351.

HORSTMAN, C. 2004. *Big Java.* [trad.] E. FURMANKIEWIECZ. Porto Alegre : Bookman, 2004.

HOUAISS, A. e VILLAR, M. S. 2001. *Dicionário Houaiss da língua portuguesa.* Rio de Janeiro : Objetiva, 2001.

JAVA. 2010. Remote Method Invocation Home. *ORACLE.* [Online] 2010. [Citado em: 10 de Março de 2010.] http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136424.html.

JENSEN, K. 1992. *Colored Petri Nets – Basic concepts, analyses methods and practical use.* s.l. : Spring – Verlag, 1992. Vol. 1.

JUDE. 2010. *JUDE.* [Online] Change Vision, Inc., 2010. [Citado em: 20 de Fevereiro de 2010.] http://jude.change-vision.com/jude-web/index.html.

KIRNER, C. e TORI, R. 2004. Introdução à Realidade Virtual, Realidade Misturada e Hiper-realidade. *Realidade Virtual: Conceitos e Tendências.* São Paulo : Mania de Livro, 2004, pp. 3-20.

KUROSE, J. F. e ROSS., K. W. 2006. *REDES DE COMPUTADORES E A INTERNET - Uma Abordagem Top-Down.* s.l. : Pearson Education, 2006.

LEITE, O. R., RODRIGUES, J. J. e OLIVEIRA, J. G. 2002. *O uso de simuladores no treinamento de operadores da Chesf como ferramenta para disseminação de conhecimentos na operação de sistemas elétricos.* 2002. Projeto de P&D da Chesf em parceria com o CEPEL 2002.

MACEDONIA, M. R. e ZYDA, M. J. 1997. *A Taxonomy for Networked Virtual Environments.* s.l. : Multimedia, IEEE, 1997. pp. 48-56.

MALCOLM, G. 2008. Visão Geral do Produto. *SQL Server 2008 R2.* [Online] Agosto de 2008. http://www.microsoft.com/sqlserver/2008/pt/br/white-papers.aspx.

MYSQL. 2010. MySQL: The world's most popular open source database. *MySQL.* [Online] ORACLE, 2010. [Citado em: 15 de Janeiro de 2010.] http://www.mysql.com/.

NASCIMENTO, J. A. N., TURNELL, M. F. Q. V. e SANTONI, C. 2007. Modelagem da Rotina Operacional de uma Subestação elétrica em HCPN. *VIII SBAI - Simpósio Brasileiro de Automação inteligente.* VIII SBAI, 2007.

NETBEANS. 2009. NetBeans. [Online] ORACLE Corporation, 2009. [Citado em: 13 de Novembro de 2009.] http://netbeans.org/.

SILVA NETTO, A. V. 2010. *Arquitetura para um ambiente de treinamento representado em realidade virtual.* Departamento de Engenharia Elétrica, Universidade Federal de Campina Grandre. 2010. Dissertação de Mestrado.

SOUSA, B. A. 2008. *Banco de dados para simulador de cursos da CHESF.* Engenharia Elétrica, Universidade Federal de Campina Grande. Campina Grande : s.n., 2008. Trabalho de Conclusão de Curso.

TORI, R., et al. 2004. *Realidade Virtual: Conceitos e Tendências.* São Paulo : Mania de Livro, 2004. pp. 159-176.

TORRES FILHO, F. e TURNELL, M. F. Q. V. 2010. Motor de simulação baseado em modelos CPN aplicado a um sistema para treinamentos de operadores. *CBA*. 2010.

WEB 3D. 2009. Web3D Consortium. *Web3D Consortium, Open Standards for Real-Time 3D Communication.* [Online] 2009. http://www.web3d.org.

APÊNDICE A: MODELAGEM DOS DADOS

MODELAGEM DOS DADOS

O projeto do sistema de banco de dados adotou o modelo Entidade Relacionamento, que representa o projeto conceitual do sistema, e se encontra descrito a seguir.

A.1 Tuplas

O modelo de dados relacional representa os dados da base de dado como uma coleção de relações. Cada relação pode ser entendida como uma tabela ou um simples arquivo de registros.

Cada linha da tabela é formada por uma lista ordenada de colunas que representa um registro, ou tupla. Os registros não precisam conter informações em todas as colunas, podendo assumir valores nulos quando assim for necessário. Na seqüência é apresentado o conjunto de relações base do projeto deste BD, na forma de tuplas, onde PK (*Primary Key*) são chaves primárias e FK (*Foreing Key*) são chaves estrangeiras.

zaa_tb_forma_treinamento (zaa_id(PK), zaa_tb_forma_treinamento);
zab_tb_objetivo_treinamento (zab_id(PK), zab_tb_objetivo_treinamento);
zac_tb_tipo_treinamento (zac_id(PK), zac_tb_tipo_treinamento);
zad_tb_estado_treinamento (id_zad(PK), zad_estado_treinamento);
za_tb_treinamentos (za_id(PK), za_operador(FK), za_tutor(FK),
za_cenarios_associados(FK), za_resultados(FK), za_tipo_treinamento(FK),
za_estado_treinamento(FK), za_objetivo_treinamento(FK), za_forma_treinamento(FK),
za_data_treinamento, za_tempo_necessario_treinamento, za_numero_operadores, za_turma);
zb_tb_resultados (zb_id(PK), zb_resultados_coletivos(FK), zb_resultados_individuais(FK));
zca_tb_tarefas (zca_id(PK), zca_identificacao, zca_edicao, zca_data, zca_motivo,
zca_observacao);
zc_tb_cenarios_associados (zc_id(PK), zc_tarefas(FK), zc_eventos(FK), zc_estado_se(FK),
zc_cenario);
zd_tb_instalacoes (zd_id(PK), zd_recurso, zd_instalacao, zd_descricao);

zd_tb_instalacoes_tem_zc_tb_cenarios_associados (zd_tb_instalacoes_zd_id(FK), zc_tb_cenarios_associados_zc_id(FK));

ze_tb_operadores (ze_id(PK), ze_instalacoes(FK), ze_operador, ze_matricula, ze_treinamento_anterior, ze_lentes_corretivas, ze_data_nascimento, ze_experiencia_anterior, ze_anos_na_funcao);

zf_tb_tutores (zf_id(PK), zf_instalacao(FK), zf_tutor, zf_matricula);

zg_tb_resultados_individuais (zg_id(PK), zg_nota, zg_descricao, zg_numero_erros);

zh_tb_resultados_coletivos (zh_id(PK), zh_nota, zh_descricao, zh_numero_erros);

zi_tb_estado_se (zi_id(PK), zi_estado_instalacao);

zja_tb_tipo_dispositivo (zja_id(PK), zja_tipo_dispositivo);

zjb_tb_categoria_dispositivo (zjb_id(PK), zjb_categoria_dispositivo);

zjc_tb_especifica_dispositivo (zjc_id(PK), zjc_especifica_dispositivo);

zj_tb_dispositivo (zj_id(PK), zj_tipo_dispositivo(FK), zj_categoria_dispositivo(FK), zj_especifica_dispositivo(FK));

zka_tb_lista_eventos (zka_id(PK), zka_evento);

zkb_tb_estado_planta (zkb_id(PK), zkb_lista_estado(FK), zkb_dispositivo(FK));

zkc_tb_estado_atuador (zkc_id(PK), zkc_dispositivo(FK), zkc_lista_estado(FK));

zkd_tb_planta (zkd_id(PK), zkd_estado);

zkd_tb_planta_tem_zkb_tb_estado_planta (zkd_planta(FK), zkb_estado_planta(FK));

zk_tb_eventos (zk_id(PK), zk_planta(FK), zk_lista_eventos(FK), zk_estado_atuador(FK), zk_pelo_tempo, zk_por_dispositivo, zk_pelo_tutor);

zk_tb_eventos_tem_zl_tb_estado_dispositivo (zl_estado_dispositivo(FK), zk_eventos(FK));

zla_tb_lista_estado (zla_id, zla_estado);

zl_tb_estado_dispositivo (zl_id(PK), zl_instalacao, zl_dispositivo, zl_momento);

zpa_tb_tipo_painel (zap_id(PK), zja_tipo_painel);

zpb_tb_aviso_painel (zpb_id(PK), zpb_tipo_aviso, zpb_aviso);

zp_tb_painel (zp_id(PK), zp_aviso_painel(FK), zp_tipo_painel(FK), zp_codigo_painel, zp_marcacao);

zp_tb_painel_tem_zj_tb_dispositivo (zj_dispositivo(FK), zp_painel(FK));

zqa_objetos_interacao (zqa_id(PK), objetos_interacao);

zq_tb_recurso (zq_id(PK), zq_objetos_interacao(FK), zq_nome);

zq_tb_recurso_tem_zp_tb_painel (zq_tb_recurso_zq_id(FK), zp_tb_painel_zp_id(FK));

A terminologia utilizada mantém o padrão do banco de dados original, de modo a facilitar uma consulta ao BD.

A.2 Dicionário de Dados

Além do modelo de Entidade e Relacionamento, é necessário armazenar um documento com a descrição dos objetos do BD. Este documento, denominado Dicionário de Dados, contém informações sobre todos os objetos do BD, de forma textual.
O intuito é padronizar a semântica do BD; e a representação dos dados, de acordo com os tipos (inteiro, caracter, entre outros). O dicionário de dados deste projeto é apresentado no anexo A.

A.3 Apresentação do Modelo Entidade-Relacionamento

Nesta etapa foi utilizada a ferramenta DBDesigner (DBDesigner, 2010), que oferece uma interface de usuário simples além de seu desenvolvimento para código aberto MySQL[2].
O diagrama MER elaborado no DBDesigner foi muito extenso para uma visualização satisfatória em apenas uma página desse documento. Para superar essa limitação o modelo foi seccionado em 6 setores:

- treinamento e resultados;
- pessoal;
- instalações;
- eventos;
- cenários e Estado da Subestação;
- dispositivo.

A seguir são apresentados cada setor, com suas propriedades. Apesar da dificuldade de visualizar o MER em um só plano, para fins de contextualização dos setores o modelo completo e é ilustrado na Figura 1.

[2] SGBD que utiliza a linguagem SQL

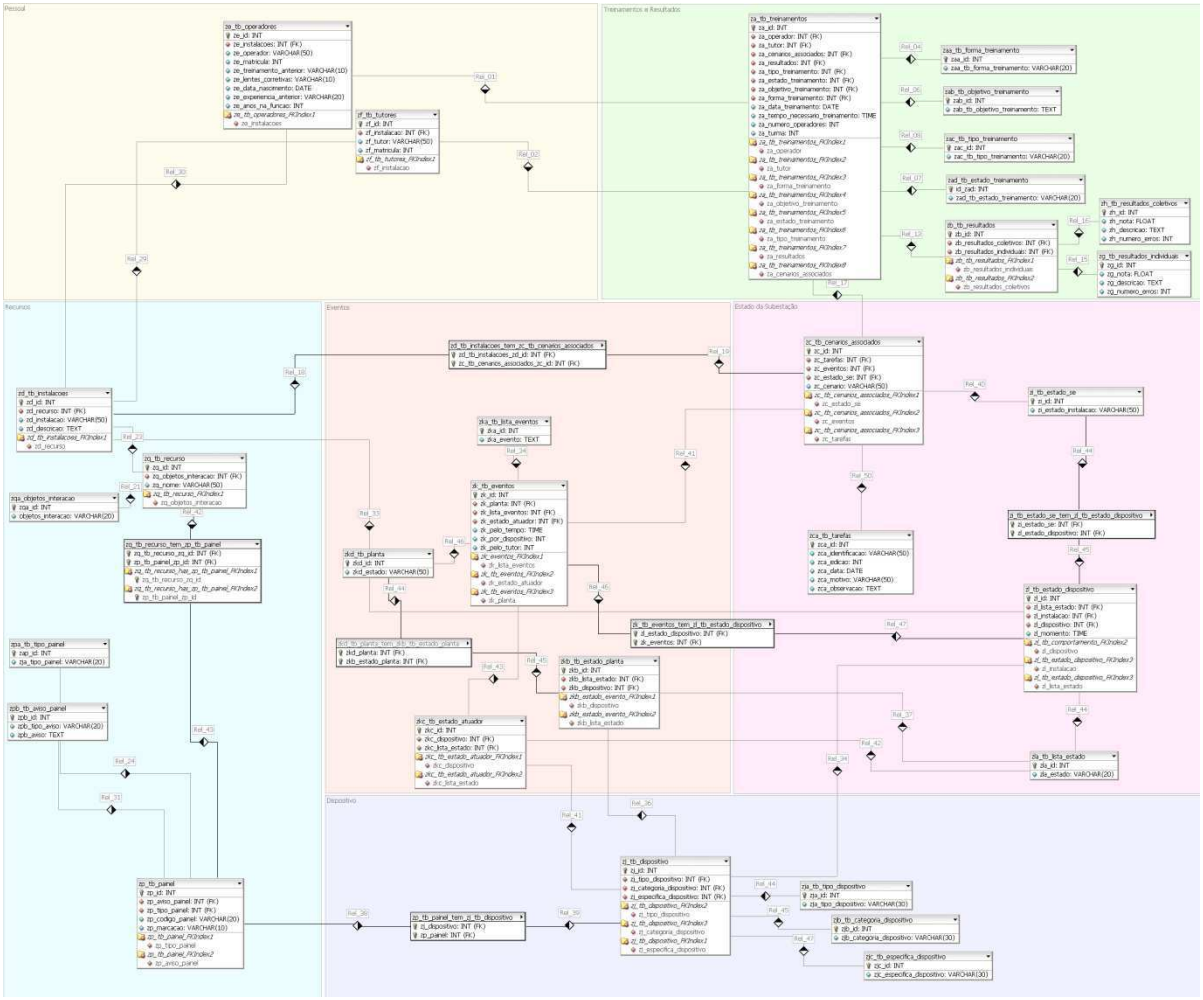

Figura 1: Visualização inteira do MER

Para cada setor, são apresentadas as características descritas a seguir.

No setor de **Treinamento e resultados** (Figura 2) há duas entidades: treinamentos e resultados. As demais tabelas que aparecem associadas a treinamento, através de chaves estrangeiras, foram introduzidas para atender as condições de normalização. A entidade treinamentos está associada a múltiplos resultados, tutores e operadores.

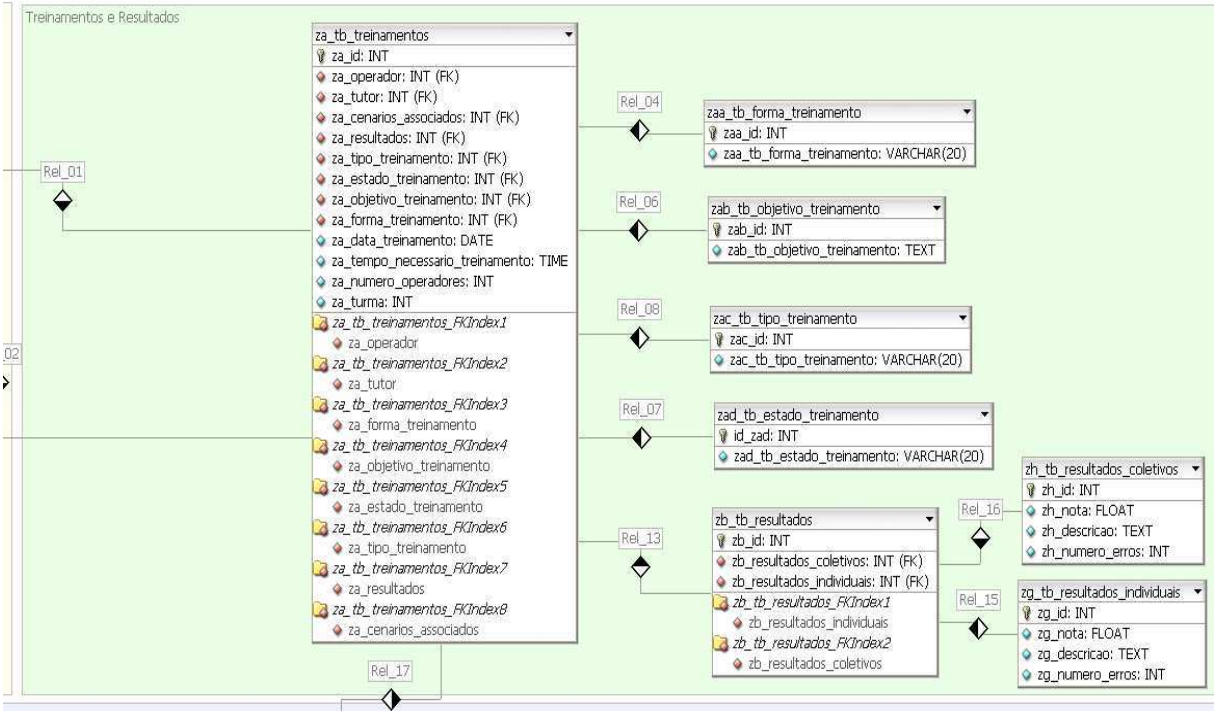

Figura 2: Treinamento e resultados

O setor de **Pessoal** (ver Figura 3) é composto pelas entidades operadores e tutores. Como citado anteriormente, ambas entidades possuem relacionamento 1:n com a entidade treinamento. Além disso, cada um (operadores e tutores) está lotado em uma única instalação.

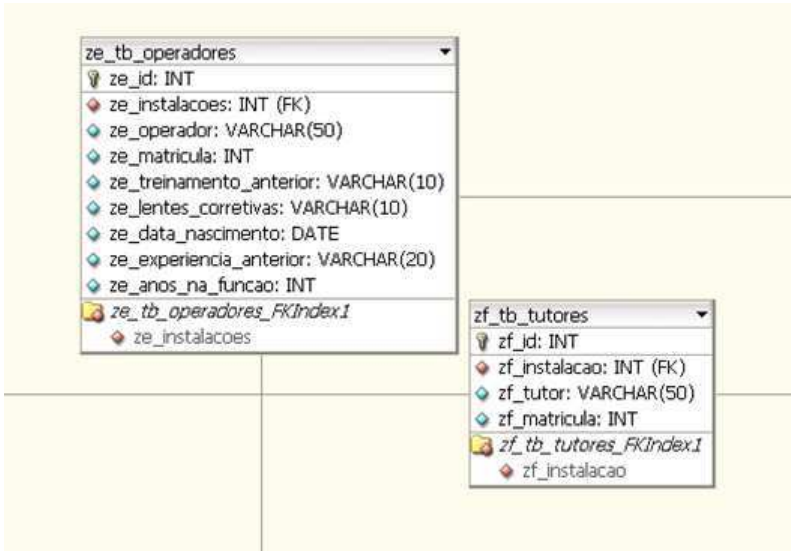

Figura 3: Pessoal

Na seção **Instalações** (ver Figura 4) há as entidades: instalações, recursos e painel. As demais tabelas são oriundas de chaves estrangeiras obedecendo as regras de normalização.
Cada instalação possui muitos recursos mas, um dispositivo é único numa instalação. Entre instalação e cenários associados é estabelecida uma relação do tipo n:m. Uma relação de muitos para muitos é estabelecida entre recurso e painel e entre painel e dispositivo.

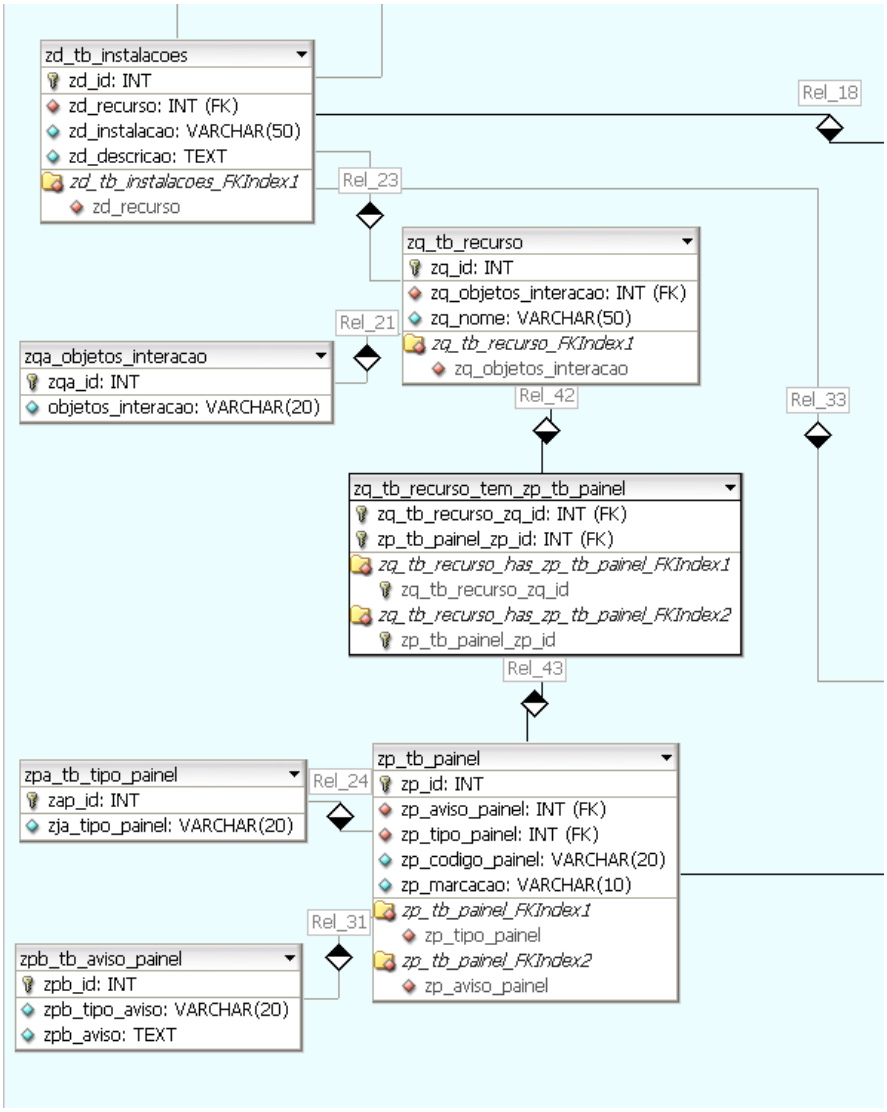

Figura 4: Instalações

O setor de **Eventos** (Figura 5) é de grande importância para o módulo tutor. Nesse setor encontram-se as entidades: eventos, planta, estado da planta e estado do atuador.
A entidade eventos possui uma lista de eventos, está associada a muitas plantas e a muitos estados dos atuadores, e está associada a um cenário. Além de possuir uma relação n:m com estado do dispositivo.

A planta possui uma relação n:m com o estado da planta. O estado do atuador e o estado da planta, possuem uma relação n:1 com a lista de estados e com os dispositivos.

Há uma tabela na parte superior da Figura 5 indicando uma relação n:m entre instalações e cenários associados.

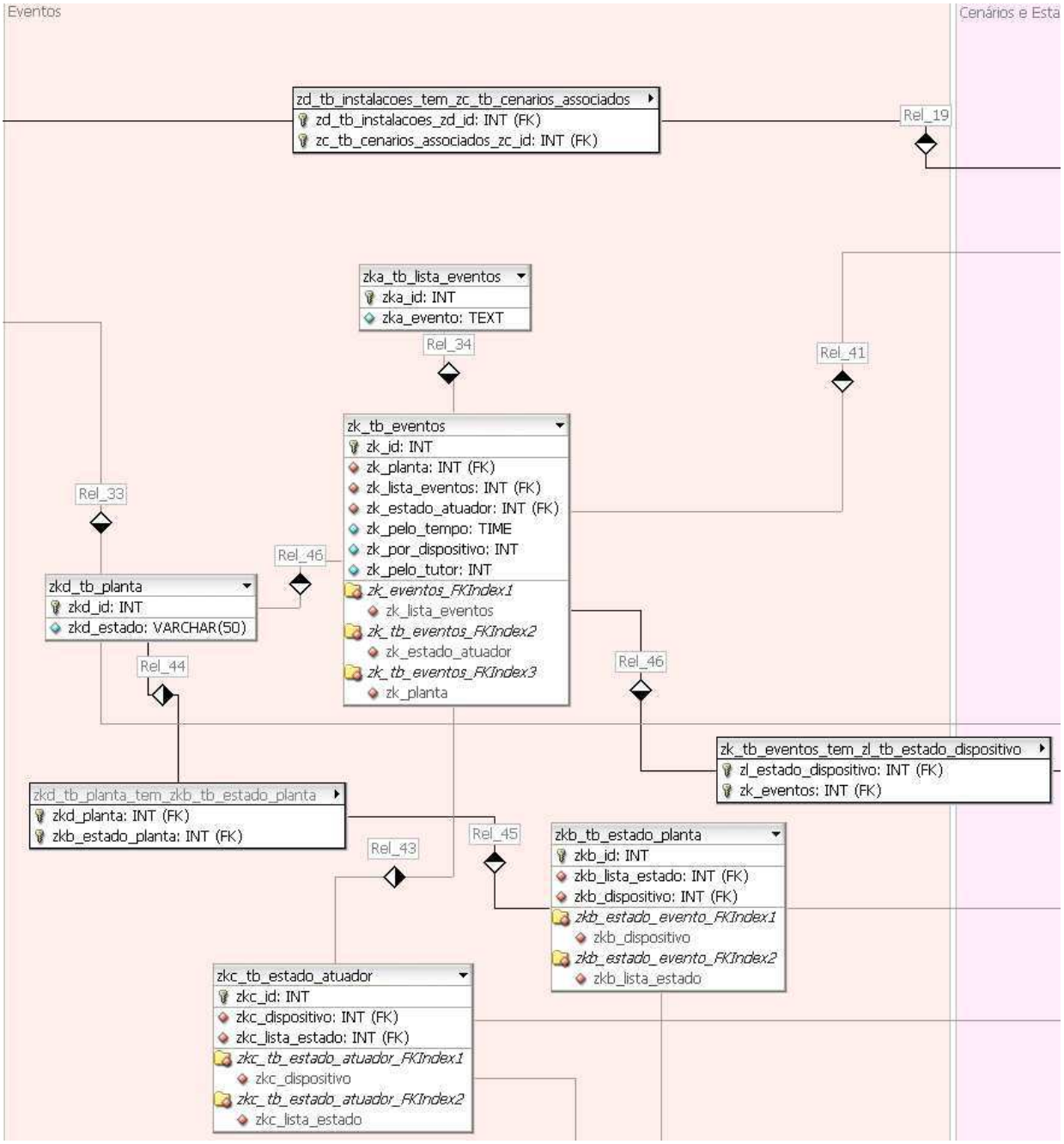

Figura 5: Eventos

Na área de **Cenários e Estados da Subestação (Figura 6)** há as entidades: cenários associados; estado da SE; estado do dispositivo; lista de estados; e tarefas.

Um cenário associado está associado a: n estados da SE, n tarefas e n eventos. Um Cenário associado está em um treinamento, e tem uma relação n:m com instalações.

O Estado do dispositivo é único em uma instalação, e tem relação n:m com estado da SE e eventos. A entidade possui n lista de estados e n dispositivos.

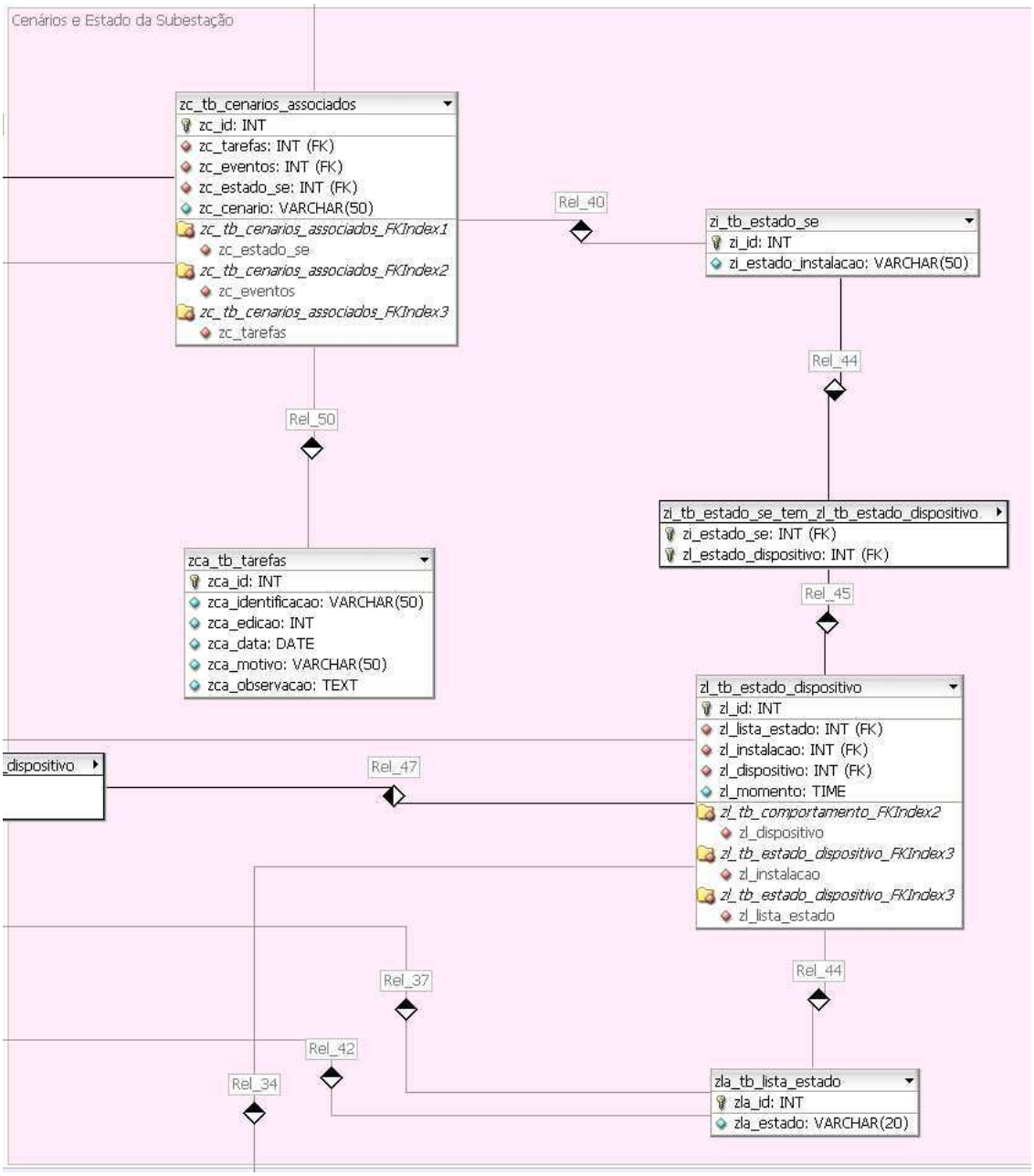

Figura 6: Cenários e Estado da Subestação

No setor **Dispositivo** (Figura 7) encontram-se o dispositivo e suas classes, em ordem crescente de nível há: tipo do dispositivo, categoria do dispositivo e especifica dispositivo. Todos os níveis tem uma relação n:1 com dispositivo.

Dispositivo está em estado do dispositivo, em estado do atuador e em estado da planta. Uma relação n:m é estabelecida entre dispositivo e painel.

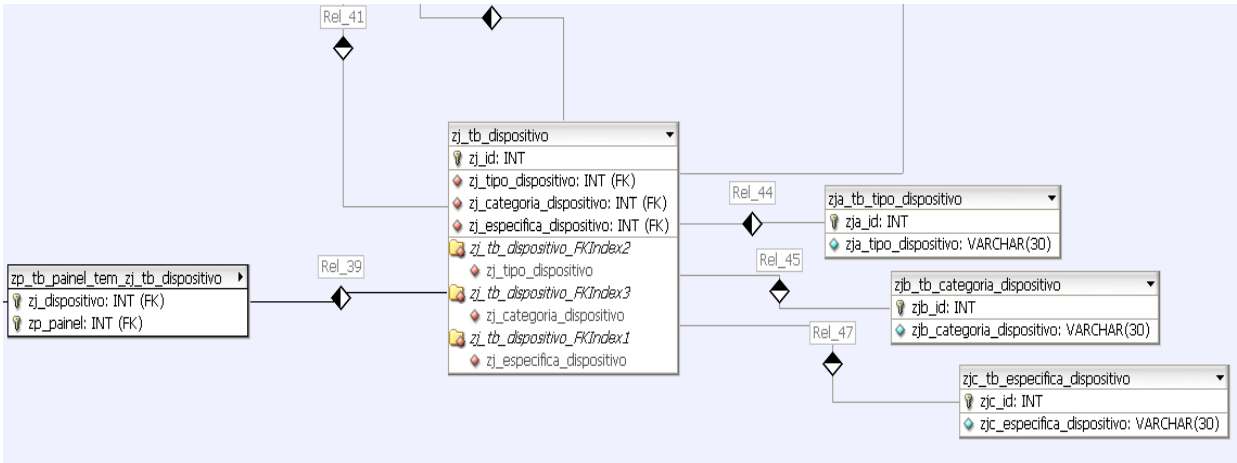

Figura 7: Dispositivo

A modelagem no DBDesigner gera uma DDL[3] em SQL.

[3] *Data Definition Language* ou Linguagem de Definição de Dados

APÊNDICE B: FERRAMENTAS

B.1 NETBEANS

O desenvolvimento de aplicações em Java é apoiado por diferentes ambientes de desenvolvimento integrado (IDE - *Integrated Development Environment*). Dentre as opções mais utilizadas destacam-se: o Eclipse e o NetBeans. Ambas são gratuitas, de código aberto e multi-plataforma, porém existe uma tendência de adoção do NetBeans devido às suas funcionalidades (Mobile Pack, NetBeans GUI Designer, NetBeans Plataform, JRuby Editor, etc.) (NETBEANS, 2009) que facilitam a criação de interfaces gráficas, a comunicação com banco de dados, a depuração dos códigos, etc. Outro fator determinante para escolha do NetBeans como plataforma de desenvolvimento neste trabalho é sua documentação de boa qualidade, disponível em Português.

B.2 CPNTools

O CPNTools é uma ferramenta de edição, simulação e análise de modelos construídos com o formalismo de Redes de Petri Coloridas (*Coloured Petri Net*). Usando CPN Tools, é possível investigar o comportamento do sistema modelado utilizando simulação, para verificar as propriedades por meio da geração do espaço de estado e verificação do modelo, e realizar análise de desempenho baseada em simulações. A interação do usuário com o CPN Tools é baseada em manipulação direta da representação gráfica do modelo CPN utilizando técnicas de interação, tais como paletas de ferramentas e menus de marcação. A Figura 1 representa a interface do software na versão 2.9.11.

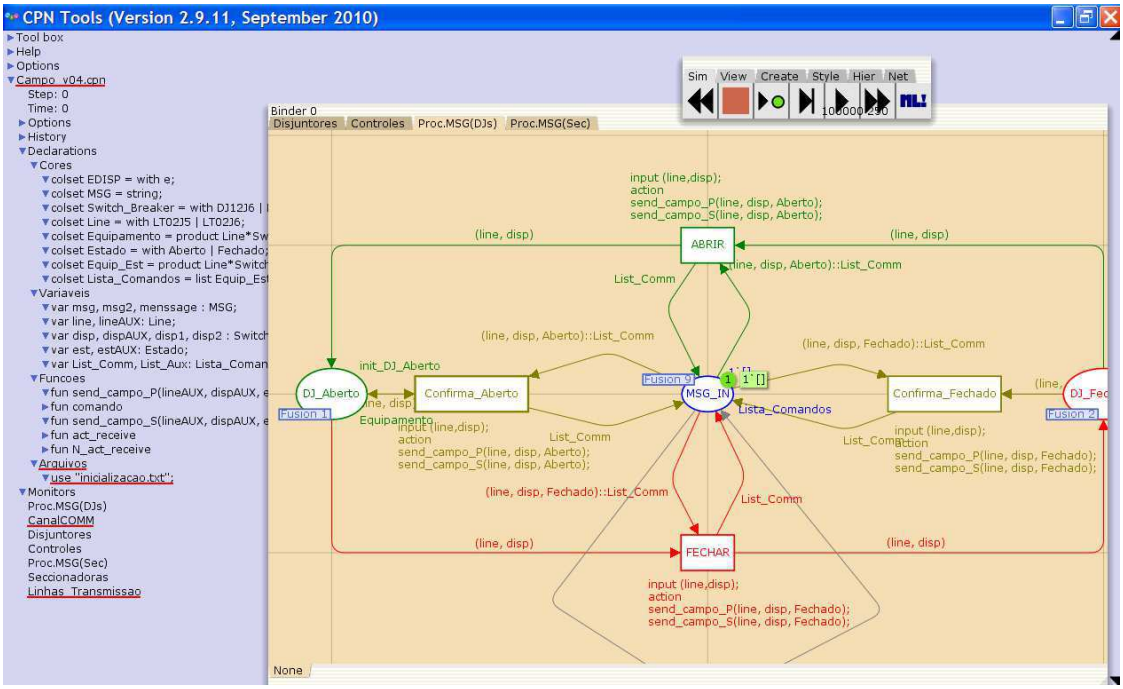

Figura 1: CPN Tools, ferramenta para edição, análise e simulação de modelos CPN

APÊNDICE C: PLANEJAMENTO E ARTEFATOS DO TESTE DE VALIDAÇÃO

Produto	Simulador para treinamento de operadores de subestações elétricas
Empresa/Responsável	LIHM – Laboratório de interfaces homem-máquina
Equipe de Avaliação	Flávio e Raffael

Plano geral do ensaio de avaliação

1	**Cliente**	
	Objetivos	Corrigir falhas; Avaliar a representação do módulo supervisório; Avaliar a navegação entre o ambiente virtual e o ambiente do supervisório; Avaliar a comunicação do módulo supervisório com demais módulos do sistema (motor de simulação, banco de dados e módulo tutor); Avaliar satisfação do usuário;
	Tempo disponível	Planejamento: 01 semana; Condução: 02 dia (03 e 04 de Mar/2011); Relato: 01 semana
	Recursos disponíveis	Produto: Simulador para treinamento de operadores de subestações elétricas
	Resultados esperados	Com relação aos usuários: Nível de satisfação no uso do produto; Nível de interesse no produto; Com relação ao produto: Identificar Falhas; Identificar problemas de usabilidade;
2	**Produto/Organização**	
	Descrição do produto	
	Descrição do contexto de uso	O produto pode ser usado no processo de treinamento de operadores em subestações elétricas
	Equipamentos auxiliares	Computadores conectados a internet.
	Treinamento, Conhecimento e Qualificação	Um dos avaliadores deve apresentar o produto aos usuários antes do início do teste.

	Natureza do produto	( ) Simples	(**X**) Complexo	
		( ) Sistema crítico	(**X**) Sistema simulado	
		( ) Sistema de automação	( ) Sistema web	
		( ) Sistema para dispositivos móveis		
		( ) Hardware	(**X**) Software	
		( ) Outro _____		
3	**Atividade**			
	Descrição da atividade	- O usuário treinando deve simular a realização de manobras no sistema elétrico, interagindo com painéis virtuais e um sistema supervisório; - O usuário tutor deve montar um cenário de treinamento, acompanhar as ações do treinando e também pode interagir com os painéis e com o supervisório.		
	Realização da atividade	(**X**) Individual	( ) Coletiva	
	Envolvidos na realização da atividade	Um usuário treinando. Integrante do LIHM atuando com Tutor.		
	É necessário o uso de uma linguagem específica?	(**X**) Sim	( ) Não	
	Em caso afirmativo, qual?	Própria dos operadores de subestações		
	É necessário o uso de equipamentos auxiliares?	( ) Sim	(**X**) Não	
	Em caso afirmativo, qual(is) e para quê?			
4	**Universo amostral**			
	Descrição do perfil desejado (características)	Pessoas com conhecimentos básicos de sistemas elétricos; Operadores de subestações		
	Descrição do perfil indesejado (características)	Pessoas sem os conhecimentos básicos de sistemas elétricos.		
	Categoria dos usuários de teste	(**X**) Inexperiente		
		( ) Intermediário		
		(**X**) Experiente (operadores de subestações)		
	Dimensão do universo amostral	3 usuários		
	Estratégia de recrutamento			
	Meio de contato	( ) Endereço	(**X**) Telefone	(**X**) Email ( ) Outro. Qual?
	Treinamento			
	Existe a necessidade de treinamento do universo amostral?	(**X**) Sim	( ) Não	
5	**Avaliação**			

Objetivo geral	Validar a integração do módulo supervisório ao ambiente do simulador.		
Objetivo específico	i. Verificar a facilidade de interação e navegação no módulo supervisório; ii. Analisar imersão do usuário durante a realização da manobra; iii. Mensuração do nível de satisfação após a execução das tarefas.		
Natureza da avaliação	(**X**) Somativa	( ) Formativa	
	(**X**) Objetiva	(**X**) Subjetiva	
	(**X**) Qualitativa	(**X**) Quantitativa	
Hipóteses	i. A representação do supervisório no ambiente de simulação é adequada; ii. O tempo de resposta da interface é suficiente e tolerável pelo usuário; iii. As informações são apresentadas na tela em uma sequência coerente e agradável ao usuário; iv. Não existe erro na comunicação do supervisório com demais módulos do simulador; v. Os estados de todos os dispositivos estão coerentemente representados em todos os módulos do simulador; vi. Todas as telas, objetos de interação e alarmes, necessários a realização da manobra escolhida, foram representados.		
Tempo estimado	40 min		
Métodos de avaliação com usuário			
	(**X**) Observação direta	(**X**) Perfil do usuário	(**X**) Satisfação do usuário
Meta			
Objetivo			
Resultados previstos			
Natureza dos testes	(**X**) Em laboratório		
	( ) Em campo		
	( ) In loco		
Número de sessões de teste	2 testes efetivos + 1 teste piloto		
Ambiente de teste (descrição)	LIHM		
Tempo estimado	40 minutos		
Características a explorar	Pessoais (Nível de conhecimento em informática)		
	Profissionais (formação acadêmica, área de atuação)		
	Uso do produto (experiência no uso do produto ou similar)		
Métodos de avaliação com especialista			
	( ) Inspeção	( ) Heurísticas	( ) Guias
Características a			

explorar

7 Planilha de responsabilidades

Avaliador	Papel	Responsabilidades
Flávio Torres	Avaliador	*Elaborar o planejamento da avaliação;* *Elaborar roteiro de teste;* *Observador;* *Elaborar e aplicar questionários (pré- e pós-teste);* *Análise de dados.*
Raffael Carvalho	Avaliador	Observador; *Aplicar questionários (pré- e pós-teste);* *Análise de dados.*
Ademar Netto	Usuário Tutor	Participar do teste como tutor e interagir com o usuário treinando.
Fátima Vieira	Coordenadora	*Elaborar o planejamento da avaliação;* *Elaborar roteiro de teste;* *Elaborar questionários (pré- e pós-teste);* *Análise de dados.*

8 Cronograma

Planejamento	**21 – 25 de Fevereiro**
Preparação do material do ensaio	**28 – 02 de Março**
Validação do ensaio (Teste Piloto)	**03 de Março**
Realização do ensaio	**04 de Março**
Análise dos dados	**05 – 08 de Março**
Relato do ensaio	

APÊNDICE D: ROTEIRO DE TAREFA DE TESTE DO TUTOR

Laboratório de Interface Homem-Máquina

http://lihm.dee.ufcg.edu.br/index.html

lihm.avaliaprodutos@gmail.com

(83) 2101-1387

Roteiro de tarefa de teste – Tutor

Neste experimento será avaliado o projeto de um simulador para treinamento de operadores na sala de comando de uma subestação elétrica. Você desempenhará o papel do tutor no treinamento e como tal deverá realizar as seguintes atividades: elaborar e configurar cenários de treinamento, gerar eventos de alarmes e acompanhar a execução do treinamento realizado pelo operador. O roteiro da atividade está organizado em quatro tarefas.

Tarefa 01 – Configuração do Cenário

A primeira tarefa consiste em configurar o cenário de treinamento para que o operador possa executar a manobra. Para isto é fornecido em anexo um guia com a configuração inicial de todos os objetos que serão utilizados na atividade. O processo de configuração consiste em verificar e assegurar que todos os objetos listados estarão no estado indicado no inicio do treinamento.

A configuração do simulador pode ser feita de três modos: (a) através da interação com os painéis no ambiente virtual, (b) através do ambiente do supervisório acessado através do ambiente virtual e, (c) através da interface do módulo tutor que oferece um menu posicionado na lateral da tela do tutor. Observe a sequência de passos no documento GUIA DE CONFIGURAÇÃO DO CENÁRIO, para que o ambiente seja configurado corretamente e o treinamento seja realizado com sucesso.

Após configurar o cenário, o tutor deve salvar as alterações efetuadas.

Tarefa 02 – Iniciar conexão com Operador

Nesta tarefa o tutor deverá interagir com o operador. Inicialmente o tutor deve iniciar o servidor para que o operador possa se conectar com o simulador. Após a conexão o tutor deve

avisar ao operador que o servidor está pronto e que ele deve tentar se conectar. Por fim o tutor deve verificar se o operador está conectado e informá-lo do sucesso da operação, concluindo assim esta tarefa.

Tarefa 03 – Acompanhar o treinamento

Esta tarefa inicia com o operador conectado, e deve ser autorizado pelo tutor para iniciar a execução da manobra. Durante esta tarefa você deve acompanhar as ações executadas pelo operador e registrar em anotações o que achar pertinente.

Roteiro de Manobras do treinamento

Origem:CROL

Equipamento: 14V2-CGD

Motivo da Revisão: Substituição do 14D1

Configuração:

- 14D1 aberto com chaves associadas fechadas, 14V2 energizando LT 04V2 CGD/NTD e todas as chaves by Pass 230kV abertas.
- Chave 43T-CS do 14D1 na posição DESATIVADO.
- Chave 43T-C/I do 14D1 na posição REATOR.

1. LIBERAÇÃO
   1.1. CGD: Receber do responsável: solicitação e liberação 14V2.
   1.2. CGD: Solicitar CROL liberação 14V2.
   1.3. CROL: Informar COSR-NE liberação 14V2/CGD.
   1.4. CROL: Autorizar CGD liberação 14V2.
   1.5. CGD: Confirmar 14D1 aberto.
   1.6. CGD: Fechar 34V2-6.
   1.7. CGD: Colocar chave 43 -14D1 na posição 'TRANSFERÊNCIA'.
   1.8. CGD: Colocar chave 43 -14V2 na posição 'TRANSFERÊNCIA'.
   1.9. CGD: Fechar 12D1.
   1.10. CGD: Abrir 14V2. (**INICIAR A TAREFA 04**).
   1.11. CGD: Colocar chave CLT-14V2 na posição 'LOC'.
   1.12. CGD: Abrir 34V2-4 e 34V2-5.
   1.13. CGD: Entregar 14V2 isolado ao responsável.
   1.14. CGD: Informar CROL conclusão liberação 14V2.

Tarefa 04 – Gerar um evento de alarme

Durante a execução da manobra, quando o operador for executar o **passo 1.10: Abrir o 14V2**, o tutor deve **acionar o evento de alarme** através da botoeira de alarme no Painel 12J6 (botão branco na parte superior do painel). Se o alarme foi corretamente acionado, você ouvirá um alarme sonoro e o botão ficará com a **luz amarela** acesa.

Após o operador identificar a origem do alarme, ele lhe informará em qual painel o alarme foi acionado e você deverá orientá-lo para ele desligá-lo e continuar a executar a manobra, uma vez que o evento que disparou o alarme poderá ser tratado posteriormente.

APÊNDICE E: ROTEIRO DE TAREFA DE TESTE DO OPERADOR

Laboratório de Interface Homem-Máquina
http://lihm.dee.ufcg.edu.br/index.html
lihm.avaliaprodutos@gmail.com
(83) 2101-1387

Roteiro de tarefa de teste – Operador

Neste experimento será avaliado o projeto de um simulador para treinamento de operadores na sala de comando de uma subestação elétrica. Você desempenhará o papel do tutor no treinamento e como tal deverá realizar as seguintes atividades: elaborar e configurar cenários de treinamento, gerar eventos de alarmes e acompanhar a execução do treinamento realizado pelo operador. O roteiro da atividade está organizado em três tarefas.

Tarefa 01 – Conectar ao Servidor

Inicialmente você deve aguardar o contato do tutor para iniciar a conexão. Quando for autorizado pelo tutor você deve iniciar a conexão com o servidor, selecionando a opção "Conexões > Conectar ao Servidor".

Para concluir você deve verificar na barra de status (canto inferior esquerdo da tela) se a conexão foi realizada com sucesso.

Tarefa 02 – Executar o roteiro de manobra

Uma vez que existem no simulador dois modos de execução de uma manobra (através do supervisório ou do ambiente virtual 3D); no roteiro de tarefa é descrito onde você deve executar cada passo. A sequência deve ser seguida para que o experimento seja realizado com sucesso.

Obs.: Caso algum alarme sonoro seja disparado durante a execução dessa manobra, interrompa a execução da etapa e prossiga para a próxima tarefa. Você poderá continuar a etapa interrompida após localizar o painel e reconhecer (desligar) o alarme.

1. Liberação do disjuntor 14V2

1.1. Realizar login no supervisório e acessar o “Visor de Telas”

1.1.1. Usuário: operador

1.1.2. Senha: 1234

1.1.3. Confirmar se o disjuntor 14D1 está aberto, na linha LT04V2 (Diagrama Unifilar de 230 KV)

1.2. CGD: Fechar 34V2-6, no painel 04V2.

1.2.1. Na “Tela de Alarmes” do supervisório, visualizar que um alarme foi gerado em decorrência do fechamento do 34V2-6.

1.2.2. Reconheça esse alarme clicando no botão “Reconhecer”.

1.3. Colocar chave 43 -14D1 na posição 'TRANSFERÊNCIA', no painel 14D1

1.4. Colocar chave 43 -14V2 na posição 'TRANSFERÊNCIA', no painel 04V2

1.5. Fechar disjuntor 14D1 da linha 04V2 no supervisório.

1.6. Abrir disjuntor 14V2 da linha 04V2 no supervisório.

1.7. Colocar chave CLT-14V2 na posição 'LOC' no painel 14V2.

1.8. Abrir chaves seccionadoras 34V2-4 e 34V2-5 no painel 14V2.

1.9. Acessar a tela de alarmes do supervisorio.

1.9.1. Visualizar os alarmes que foram gerados em decorrência da mudança de estado dos equipamentos.

1.9.2. Reconhecer os alarmes clicando no botão “Reconhecer”.

Tarefa 03 – Alarme de Eventos

Na ocorrência de uma alarme sonoro você deve identificar o painel associado ao evento. Para reconhecer o alarme no painel existe uma botoeira de alarme que está localizada na parte superior do painel a qual deverá estar acesa (indicação em amarelo), após reconhecer o alarme você deve informar ao tutor e perguntar qual o procedimento a ser executado.

Uma forma alternativa de identificar o equipamento associado ao alarme/evento, está disponível na tela de alarmes do supervisório.