CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Universidade Federal
de Campina Grande

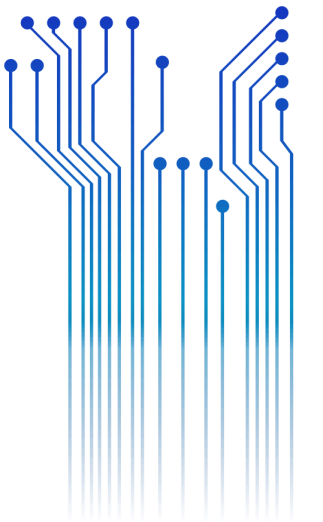
ANDRÉ GOMES MEDEIROS DE SOUZA

Centro de Engenharia
Elétrica e Informática

TRABALHO DE CONCLUSÃO DE CURSO

GERAÇÃO DE CENÁRIOS DE TREINAMENTO A PARTIR DE LOG DE INCIDENTES EM SUPERVISÓRIOS E RELATÓRIOS RDFH

Departamento de
Engenharia Elétrica



Campina Grande
2016

ANDRÉ GOMES MEDEIROS DE SOUZA

GERAÇÃO DE CENÁRIOS DE TREINAMENTO A PARTIR DE LOG DE INCIDENTES EM SUPERVISÓRIOS E RELATÓRIOS RDFH

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Controle e Automação

Orientador:

Professora Maria de Fátima Queiroz Vieira, Ph.D.

Campina Grande

2016

ANDRÉ GOMES MEDEIROS DE SOUZA

GERAÇÃO DE CENÁRIOS DE TREINAMENTO A PARTIR DE LOG DE INCIDENTES EM SUPERVISÓRIOS E RELATÓRIOS RDFH

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Controle e Automação

Aprovado em / /

Professor Avaliador
Universidade Federal de Campina Grande
Avaliador

Professora Maria de Fátima Queiroz Vieira, Ph.D.
Universidade Federal de Campina Grande
Orientadora, UFCG

Dedico este trabalho em primeiro lugar à minha falecida avó Eny Gomes de Medeiros, cujo exemplo jamais será esquecido. Dedico também aos meus pais Wilton Camelo de Souza e Luthgard Gomes Medeiros de Souza, pelo apoio e incentivo, assim como aos meus irmãos e amigos.

AGRADECIMENTOS

Agradeço à minha falecida avó Eny Gomes de Medeiros em primeiro lugar, cujo exemplo de dedicação e esforço em prol do estudo estará presente para sempre em minha mente.

Agradeço também à minha mãe, Luthgard, pelo apoio financeiro e emocional incessantes, pelas preocupações e, principalmente, pelo carinho e paciência em me guiar por essa jornada tão turbulenta.

Agradeço também ao meu pai, Wilton, que sempre esteve presente nas necessidades, e que com muita seriedade me aconselhou ao longo desta graduação. Agradeço também aos meus irmãos Filipe e Lucas, que sempre estiveram próximos, nos momentos de maiores necessidades.

Enfim, agradeço aos meus amigos Brenno, Carlos, Everton e Gustavo, assim como os demais colegas de graduação que, com toda atenção, foram indispensáveis para alcançar essa conquista.

"Meu objetivo é simples. É uma completa compreensão do universo, porque é como é e porque ele existe em tudo. "

Sthephen Hawking.

RESUMO

Em simuladores para o treinamento de operadores, a geração de um cenário de treinamento demanda muito tempo e criatividade. No ambiente SIMULOP, amplamente utilizado no setor elétrico brasileiro, a interface com o tutor para construção de cenários não é ergonômica, demandando além do tempo e criatividade, um grande esforço na sua descrição. Os processos de elaboração desses cenários têm um caráter geral, não levando em conta as necessidades de treinamento individual dos operadores. Neste projeto é proposta a geração automática de cenários de treinamento, para o ambiente SIMULOP, a partir da análise de arquivos do histórico da operação, durante falhas ocorridas no sistema (*log*) e do respectivo Relatório Desligamento da Falha Humana - RDFH, visando reproduzir no ambiente simulado ocorrências reais, evitando assim a repetição de falhas de origem humana.

Com a ferramenta *Gerador Automático de Cenários,* desenvolvida durante um projeto de P&D com a CHESF, no Laboratório de Interfaces Homem-Maquina (LIHM) da UFCG e de tendo analisado relatórios RDFH e *logs* de incidentes obtidos do sistema supervisório SAGE adotado no setor elétrico nacional, foram construídas duas ontologias que descrevem um cenário de treinamento, as quais foram seguida utilizadas para produzir um arquivo de texto puro o qual pode ser executado pelo OTS do sistema SAGE/SIMULOP.

Assim, com base nas duas ontologias tornou-se possível gerar cenários de treinamento a partir de ocorrências reais, os quais podem alimentar uma base de dados que permite a classificação dos eventos e cenários de acordo com uma tipologia do erro humano.

Palavras-chave: Geração automática de cenários de treinamento; SAGE, *Log* de Falhas; Relatórios de Falha Humana; Cenários de Treinamento, Ontologias.

ABSTRACT

In simulators for operator training, generating a training scenario demands a lot of time and creativity. In the SIMULOP software environment, widely used in the Brazilian electric sector, the interface with the tutor, for scenario building, is not ergonomic, demanding high efforts, time and creativity, in their description. The development process of these scenarios is general, not accounting for the individual training needs of operators. In this project, we propose the automatic generation of training scenarios, for the SIMULOP environment, based on the analysis of log files of the operation, during which failures happened in the system, as well as the descriptive report of human error – RDFH. This is done in order to simulate real environment occurrences aiming to avoid the reoccurrence failures of human origin. This project was developed based on a tool, built to support the tutor when interactively building Scenarios, which was developed during an R & D project with the company CHESF at LIHM at UFCG. It combines the information from the analysis of RDFH reports with that from incident logs, recovered from the supervisory system SAGE, also adopted by the electricity sector. This project is based on two ontologies which were built to describe a training scenario. From the ontologies, it is produced a plain text file which can be run by the OTS in the SAGE / SIMULOP system. Thus, based on these two ontologies it became possible to generate training scenarios from real events, which can be stored in a training scenario database, which allows for the classification of events and scenarios according to a typology of the human error.

Keywords: Automatic generation of training scenarios; SAGE, Log of failures; Reports of Human Failure; Training scenarios, Ontology.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CEPEL	Centro de Pesquisa de Energia Elétrica
CHESF	Companhia Elétrica de São Francisco
EMS	*Energy Management System*
EPRI	*Electric Power Research Institute*
HP	*Hewlett Packard*
IHM	Interface Homem-Máquina
LIHM	Laboratório de Interface Homem-Máquina
ONS	Operador Nacional de Sistema Elétrico
OTS	*Operator Training Simulator*
OWL	*Ontology Web Language*
RDFH	Relatório Desligamento por Falha Humana
SAGE	Sistema Aberto de Gerenciamento de Energia
SCADA	*Supervisory Control and Data Acquisition*
SIN	Sistema Integrado Nacional
UML	*Unified Modelling Language*
W3C	*World Web Wide Consortium*

SUMÁRIO

1 INTRODUÇÃO

Com o crescimento e a complexidade do setor elétrico brasileiro torna-se cada vez maior a procura pela melhor capacitação de operadores de sistemas. A demanda de energia é um fator importante na economia nacional, fator este que exige do setor elétrico robustez e segurança. Com uma hierarquia de controle organizada e distribuída Sistema Integrado Nacional *(SIN),* as diversas empresas do setor se interconectam e compartilham entre si informações através de seus sistemas de automação, baseados em sistemas supervisórios. Esta atividade exige cada vez mais competência dos operadores na operação dos sistemas [8].

Os operadores do sistema elétrico recebem aulas teóricas, porém a prática acontece principalmente diante do sistema real. Esse treinamento é incompleto [8], visto que se mostrou mais eficiente a utilização de simuladores para direcionar o treinamento desses operadores, com o objetivo de simular situações de emergência, e situações raras que ocorrem na prática.

O Centro de Pesquisa de Energia Elétrica *(CEPEL)* desenvolveu um sistema supervisório que atua na área de controle de sistemas elétricos. Esse supervisório foi projetado com características tais como: portabilidade, interconectividade, expansibilidade e modularidade. Com isso, o sistema SAGE – *Sistema Aberto de Gerenciamento de Energia* – pode ser adotado por diversas companhias de geração, transmissão e distribuição de energia, particularmente na Companhia Hidrelétrica de São Francisco *(CHESF)* a maior empresa do sistema elétrico nacional que opera no Norte e Nordeste [7].

Com o objetivo de treinar melhor seus operadores, a CHESF em parceria com o CEPEL, desenvolveu em 2001 o projeto SIMULOP – um simulador para treinamento de operadores. Este ambiente integra a ferramenta de supervisão e controle de sistemas de potência o SAGE, baseado no EMS – *Energy Management System,* desenvolvido pelo EPRI. A integração do EMS com o OTS – *Operator Training Simulator,* também desenvolvido pela EPRI (*Electric Power Research Institute*), originou o simulador digital em tempo real de sistemas elétricos- SIMULOP [8].

A interface homem-máquina *(IHM)* do OTS apresenta os diagramas unifilares das instalações do sistema CHESF tal como apresentados na IHM do supervisório SAGE, e possibilita a interação do operador em treinamento com os equipamentos que constituem o sistema elétrico real, mas cujo comportamento é simulado pelo OTS.

A interação do operador com o SAGE gera automaticamente um arquivo de *log* com todas as ações do operador, respostas do sistema e os eventos ocorridos no período observado. Esses arquivos de *logs* são armazenados, por períodos definidos na empresa, para análise posterior de situações de falha do sistema CHESF. Após a ocorrência de falha no sistema, é requisitado um relatório técnico do engenheiro responsável pela instalação, ao qual é anexado parte do arquivo de *log* – também conhecido como *arquivo de auditagem* – referente ao incidente. No caso de treinamentos, as informações contidas nos *logs* refletem o nível de competência dos operadores na operação do sistema elétrico.

1.1 MOTIVAÇÃO

Durante o processo de elaboração de cenários de treinamento, os tutores procuram reproduzir uma ocorrência com base na própria experiência em campo ou durante ocorrências nos centros ou salas de operação. Dessa forma, os processos de elaboração desses cenários têm um caráter geral não levando em conta as necessidades de treinamento individual dos operadores. Por outro lado, são eficazes na preparação do operador para um tipo especifico de ocorrência.

A geração de um cenário de treinamento demanda muito tempo e criatividade. No ambiente SIMULOP, a interface com o tutor não é ergonômica exigindo além do tempo e criatividade, um esforço considerável na descrição do cenário. Uma das maiores dificuldades impostas é o conhecimento amplo da base de dados do sistema elétrico a qual possui a referência de todos os equipamentos e as ações possíveis vinculadas aos equipamentos de uma instalação [6].

Em parceria com o LIHM – Laboratório de Interface Homem-Máquina, a CHESF, através de um P&D, construiu um ambiente de geração de cenários para o SIMULOP. Esse ambiente fornece ao tutor uma interface ergonômica para edição e criação de cenários de treinamento. Esse ambiente contém duas estruturas de dados: Um banco de dados compartilhado, que contém uma cópia de todos os equipamentos e

subestações da CHESF contida no SAGE, assim como uma Ontologia no domínio da aplicação. Com o *Gerador de Cenário para Treinamento de Operadores*, a CHESF dispõe de uma ferramenta para criação e edição desses cenários de treinamento.

Apesar da redução das dificuldades para gerar cenários decorrentes dos resultados do projeto, a geração automática de cenários de treinamento para o SIMULOP continua sendo a maior aspiração dos tutores que o utilizam. O desejado é uma ferramenta capaz de gerar automaticamente um cenário de treinamento, a partir do pressionar de um botão.

Neste projeto é proposta e desenvolvida uma solução nesta direção, ainda em estágio preliminar, a qual possibilita a geração automática de cenários de treinamento a partir da análise do arquivo de auditagem (*log*) de uma ocorrência e do respectivo *Relatórios de Desligamento por Falha Humana*- RDFH, visando reproduzir em um ambiente simulado ocorrências reais e evitar a repetição de falhas humanas.

1.2 OBJETIVOS

Desenvolvimento de uma API (*Application Programming Interface*) para gerar cenários de treinamento de forma automática, a partir da extração de informações do arquivo de auditagem (log), armazenados no supervisório SAGE, e do respectivo relatório RDFH. Os cenários gerados consistirão do estado inicial do sistema elétrico antes da situação de falha e, da sequência de eventos de acordo com a evolução temporal dessa situação.

A API deverá produzir um arquivo OWL (*Ontology Web Language*) contendo a descrição do cenário gerado, assim como um arquivo executável no formato reconhecido pelo SIMULOP (*texto puro*).

1.3 ESTRUTURA DO TRABALHO

No capítulo 2 discutiremos os fundamentos teóricos importantes para a implementação do projeto – o que é um cenário de treinamento, o sistema supervisório SAGE, o uso de Ontologias; os relatórios RDFH e as classificações do erro humano.

Em seguida, no capítulo 3, discorreremos sobre o desenvolvimento do sistema, abordando a sua arquitetura e os diagramas associados, os requisitos da interface homem-máquina e uma discussão sobre as suas principais classes.

No capítulo 4 discutiremos as estratégias de validação e os resultados obtidos, seguidos da conclusão do relatório com considerações sobre o projeto, no capítulo 5.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo apresentaremos os conceitos fundamentais para alcançar os objetivos propostos nesse trabalho. Inicialmente, discutiremos o que é um cenário de treinamento, em seguida discutiremos alguns aspectos importantes sobre o sistema supervisório SAGE – principalmente porque este supervisório fornecerá um objeto fundamental para o desenvolvimento deste trabalho, o arquivo de auditagem. Por fim, será discutida a estrutura de dados utilizada na organização da hierarquia dos objetos que compõem o sistema (Ontologia) e, os relatórios RDFH, que são a fonte de informação para a escolha dos objetos que representarão um cenário de erro.

2.1 CENÁRIO DE TREINAMENTO

Devido ao alto grau de complexidade em seus sistemas elétricos, a CHESF realiza periodicamente treinamentos com os seus operadores de sistema e de subestações, com a intenção de reciclá-los. Inicialmente, estes treinamentos limitavam-se a aulas teóricas e visitas técnicas, tornando inviável lidar com situações graves e de emergência [8].

Por esta razão, a CHESF em parceria com a CEPEL, desenvolveu em 2001 o projeto SIMULOP, visto que a eficiência de um treinamento pode ser adequadamente alcançada utilizando-se simuladores do sistema para treinamento de operadores.

O SIMULOP é uma ferramenta que integra o simulador digital em tempo real, o OTS, desenvolvido pela EPRI, à ferramenta EMS, o sistema supervisório SAGE. A arquitetura do SIMULOP é apresentada na figura 1.

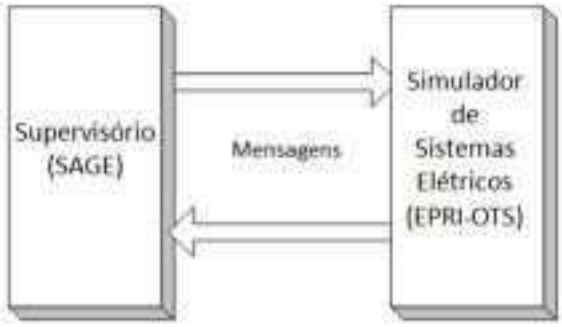


Figura 1: Diagrama representativo da integração entre o SAGE e o OTS, formando o projeto Simulop (Fonte: [8]).

O SIMULOP é capaz de oferecer aos operadores em treinamento uma interação com os mesmos componentes do sistema presentes no mundo real. No processo de elaboração de um treinamento, uma equipe de tutores elabora um cenário de treinamento, que será interpretado pelo simulador como uma evolução do estado do sistema elétrico representado.

Na elaboração de um cenário de treinamento, é necessário definir eventos sequenciais que serão executados pelo simulador OTS. Estes eventos são ações específicas sobre os equipamentos descritos no supervisório SAGE, por exemplo a abertura de um disjuntor. Um conjunto de eventos está associado a um grupo de eventos. O conjunto de grupos de eventos compõe um cenário de treinamento.

Atualmente, a descrição desses cenários de treinamento é realizada utilizando a interface do OTS, e sua estrutura é apresentada na figura 2. Os Casos Bases são *snapshots* do estado das instalações elétricas representadas no supervisório, enquanto os grupos de eventos e eventos são definidos pelo tutor na concepção do cenário de treinamento.

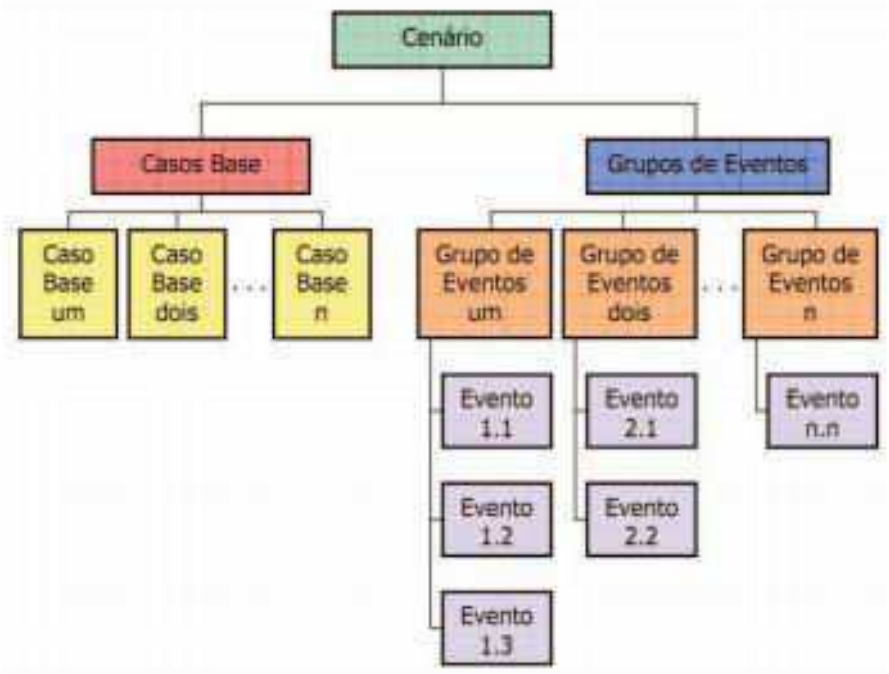


Figura 2: Representação da estrutura de cenários de treinamento no OTS (Fonte: [8]).

A elaboração de um cenário de treinamento para o OTS consiste na concepção e definição de eventos que descrevam uma falha no sistema elétrico. Na maior parte dos casos se baseiam em falhas reais que os tutores têm conhecimento.

2.1.1 GRUPOS DE EVENTOS

Um Grupos de Evento corresponde a um ou mais eventos, acompanhado de uma descrição textual. O SIMULOP limita a quantidade de eventos por grupo a 99 eventos, e a quantidade de grupos de eventos por cenário, ao máximo de 20 grupos por cenário. Estes podem ser ativados ou desativados durante uma simulação.

2.1.2 EVENTOS

Eventos são ações específicas que definem o comportamento de um equipamento durante a simulação. No escopo deste relatório, nos ateremos aos eventos disparados temporalmente, ou seja, eventos que são disparados em um tempo específico da simulação, previamente determinado pelo tutor.

A estrutura de um evento consiste na definição dos seguintes parâmetros:

i. Instalação: Identifica a instalação que o equipamento pertence;
ii. Equipamento: Identifica o equipamento relacionado ao evento;
iii. Ação: Identifica o tipo de ação à qual o equipamento será submetido;
iv. Tipo de disparo: Pode ser do tipo temporal ou condicional;
v. Valor do disparo: um valor de tempo no formato HH:MM:SS ou uma *tag* que identifique uma condição;
vi. Mensagem: A mensagem que será exibida no OTS, quando o evento for executado.

O próprio OTS fornece uma interface para criação desses eventos. Na figura 3 é ilustrada a tela de edição de grupo de eventos, enquanto na figura 4 é ilustrada a tela de edição de eventos. Outra característica importante é que o OTS possibilita a exportação e importação de cenários de treinamento.

Figura 3: Tela de edição de grupos de eventos no simulador OTS (Fonte: [8]).

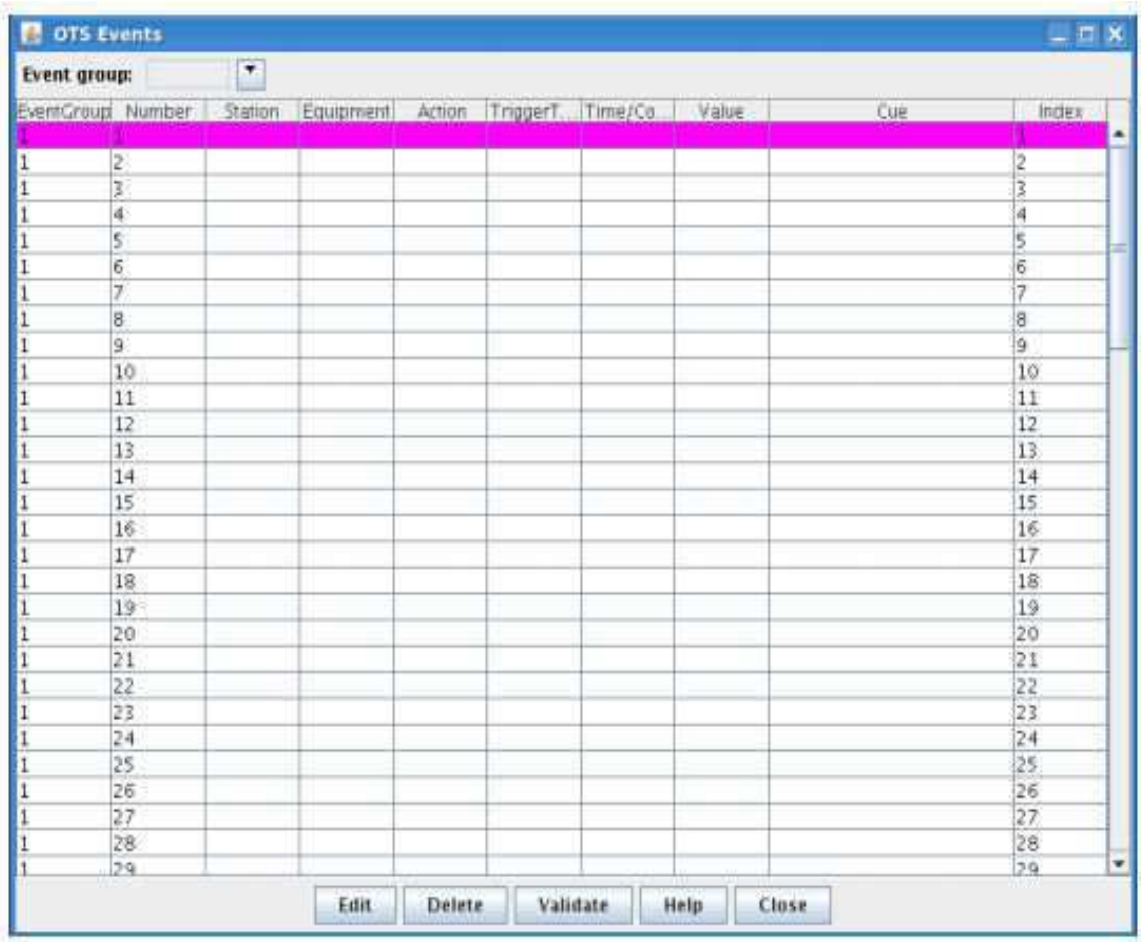


Figura 4: Tela de edição de eventos no simulador OTS (Fonte: [8]).

2.2 SISTEMA SUPERVISÓRIO SAGE E ARQUIVO DE AUDITAGEM (*LOG DE INCIDENTES*)

Este capítulo se fundamenta no texto *SAGE – Um sistema aberto para evolução* [10]. O SAGE é um sistema supervisório SCADA/EMS (*Supervisory Control and Data Acquisition*) desenvolvido pelo CEPEL para implementação da supervisão e controle de sistemas elétricos, adotado por diversas empresas que atuam no âmbito da geração, transmissão e distribuição de eletricidade no Brasil. O SAGE é amplamente usado em escala nacional, em especial pelas empresas associadas ao CEPEL: CHESF, Furnas, Eletronorte e Eletrosul, além do ONS *(Operador Nacional de Sistemas Elétricos)* [10].

Com uma arquitetura aberta e distribuída, o SAGE resultou em melhorias na interconexão de postos de controle e na qualidade da informação e suporte a

plataformas heterogêneas de hardware [10]. O sistema apresenta as seguintes características:

- Portabilidade – o sistema é capaz de executar suas funções em diferentes tipos de hardware;
- Expansibilidade – o sistema é capaz de expandir-se em termos de hardware e de software, provendo melhorias em termos de capacidade computacional e de novas funcionalidades;
- Modularidade – os diferentes módulos de software podem ser alterados ou removidos sem comprometer o funcionamento do sistema;
- Interconectividade – utiliza-se de uma rede padrão, permitindo a conexão de diferentes tipos de hardware.

A característica do SAGE mais importante para este trabalho é sua capacidade de produzir relatórios diários, com arquivos contendo *logs* de ocorrências, as quais são usadas para auditagem. O SAGE possui uma interface gráfica que permite acessar estes arquivos. A estrutura do arquivo de log é padrão, como ilustra um trecho apresentado na figura 5.

```
12:04:48 Chave Abriu                                  ACD:34T2-1:89          Seccionadora 34T2-1
12:04:48 Chave Fechou                                 ACD:34T2-2:89          Seccionadora 34T2-2
12:04:48 Chave Abriu                                  ACD:34T2-4:89          Seccionadora 34T2-4
12:04:48 Chave Abriu                                  ACD:34T2-5:89          Seccionadora 34T2-5
12:04:48 Chave Fechou                                 ACD:34T2-6:89          Seccionadora 34T2-6
12:04:48 Retornou à região de normalidade (68.98)     BGI:02BP-1:KVBC        Medi??o de Tens?o Fase BC
12:04:48 Retornou à região de normalidade (68.98)     BGI:02BP-2:KVBC        Medi??o de Tens?o Fase BC
12:04:48 Retornou à região de normalidade (68.16)     PEN:02BP:KVBC          KV Fase B do 02BP-PEN
```

Figura 5: Trecho do arquivo de *log* cedido pela CHESF, relativo a 02 de outubro de 2014. (Fonte: Empresa CHESF).

É possível identificar a partir deste trecho algumas informações relevantes, tais como: equipamento *(e.g., ACD:34T2-1:89, na primeira linha)* e a ação correspondente sobre este equipamento *(Chave Abriu)*. Com estas informações, é possível extrair do arquivo as informações necessárias para a construção de um cenário de treinamento, a partir de um conjunto de eventos decorrentes de uma ocorrência, registrada no arquivo.

2.3 ONTOLOGIAS

A palavra ontologia tem origem no Grego. *Ontos*¸ significa "ser", enquanto *logia*, "palavra" [6]. Filosoficamente, o termo ontologia tem sido usado para descrever o conhecimento da natureza e a organização do ser.

O conceito de ontologia tem sido aplicado à ciência da informação para organizar hierarquicamente grupos de conceitos que definem um domínio. Em outras palavras, "Uma ontologia define um conjunto de termos ordenados hierarquicamente para descrever um domínio e pode ser usada como o esqueleto de uma base de conhecimentos. " [6].

Uma vez formalizada, a ontologia é passível de ser processada por máquinas. Seus componentes básicos são classes, relações, axiomas e instâncias. Enquanto as classes são dispostas de forma hierárquica, as relações são representações da forma como os conceitos do domínio da ontologia interagem; os axiomas são regras que definem as associações entre as classes e as relações; as instâncias são os elementos que compõem o domínio.

Dentro da engenharia de *software*, o conceito de ontologia está relacionado à especialização de um vocabulário, reduzindo a probabilidade de erros durante o desenvolvimento causados por erros conceituais ou de terminologia. Além disso, pode ser compartilhada entre desenvolvedores, de forma que a aquisição de informações pode ser feita através de uma análise ontológica.

Através da ontologia – enquanto base conceitual – é possível identificar, classificar e definir componentes de software, facilitando a estruturação, modularização e o reuso, desde que a aplicação deste software se encontre no mesmo domínio da ontologia.

Por esta razão, enquanto um ou mais projetos de *softwares* se encontrarem no mesmo domínio, o uso de ontologias é adequado, principalmente porque sua utilização permite identificar componentes reutilizáveis de *software* no sistema, garantindo a reusabilidade e a modularidade. Dessa forma, de acordo com Torres [6] apud [11], os objetivos que podem ser alcançados ao se utilizar ontologia em projeto de software são:

- Especificação – definição de requisitos e funcionalidades do sistema;
- Validação – através de processos semiautomáticos, pode-se avaliar e verificar a consistência de um projeto;

- Reuso – quando organizada de forma modular, sua estrutura hierárquica de domínio, subdomínios e tarefas podem ser reutilizadas em outros problemas;
- Busca – uma ontologia é um repositório de informação;
- Manutenção – melhoria no uso e no armazenamento de informações para manutenção do sistema;
- Aquisição de conhecimento – serve como guia para aquisição de novos conhecimentos;

2.3.1 LINGUAGENS

Atualmente existem diversas linguagens que apoiam a especificação de ontologias, neste projeto utilizaremos a linguagem OWL, recomendada pelo W3C – *(World Web Wide Consortium)*. A linguagem OWL consiste de três sublinguagens, que diferem entre si pelo nível de expressividade – OWL-Lite, OWL-DL e OWL-Full [6], ver figura 6.

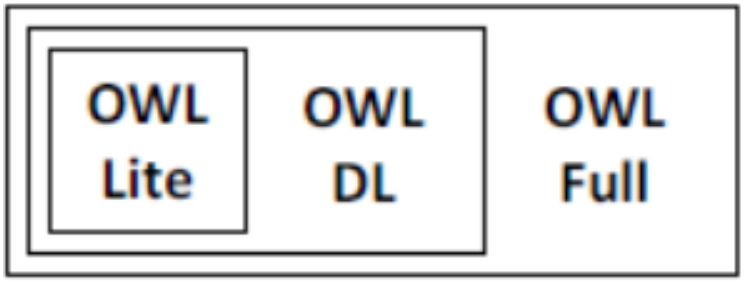


Figura 6*: Sub-Linguagens* OWL (Fonte: [6]).

De acordo com [6], as sublinguages OWL classificam em:

- OWL-Lite: sintaxe simples, utilizada quando a hierarquia e as restrições entre as classes são necessariamente simples;
- OWL-DL: trata-se de uma linguagem mais expressiva que a anterior, organizada a partir de uma lógica descritiva, passível de raciocínio automático – dessa forma, é possível verificar sua consistência automaticamente;
- OWL-Full: mais expressiva que as anteriores, é utilizada quando é necessário um alto grau de expressividade para conceituar, ou formalizar uma ontologia. Não é possível realizar inferências nessa sublinguagem.

Porém, para facilitar o uso da linguagem OWL, utilizamos a ferramenta *Protégé* que provê uma interface ergonômica para a criação de projetos com base em ontologias.

2.3.2 FERRAMENTAS

O *Protégé* é uma ferramenta desenvolvida pelo departamento de informática da Universidade de Stanford com o objetivo de prover ao usuário um ambiente de desenvolvimento para ontologias [6].

Desenvolvido em Java, o *Protégé* adota uma política de código aberto, o que permite o desenvolvimento de componentes integráveis de software, utilizando as APIs do *Protégé* [6].

A figura 7 apresenta a interface da ferramenta e uma organização hierárquica de uma ontologia como exemplo. O guia do usuário está disponível em <http://protege.stanford.edu>.

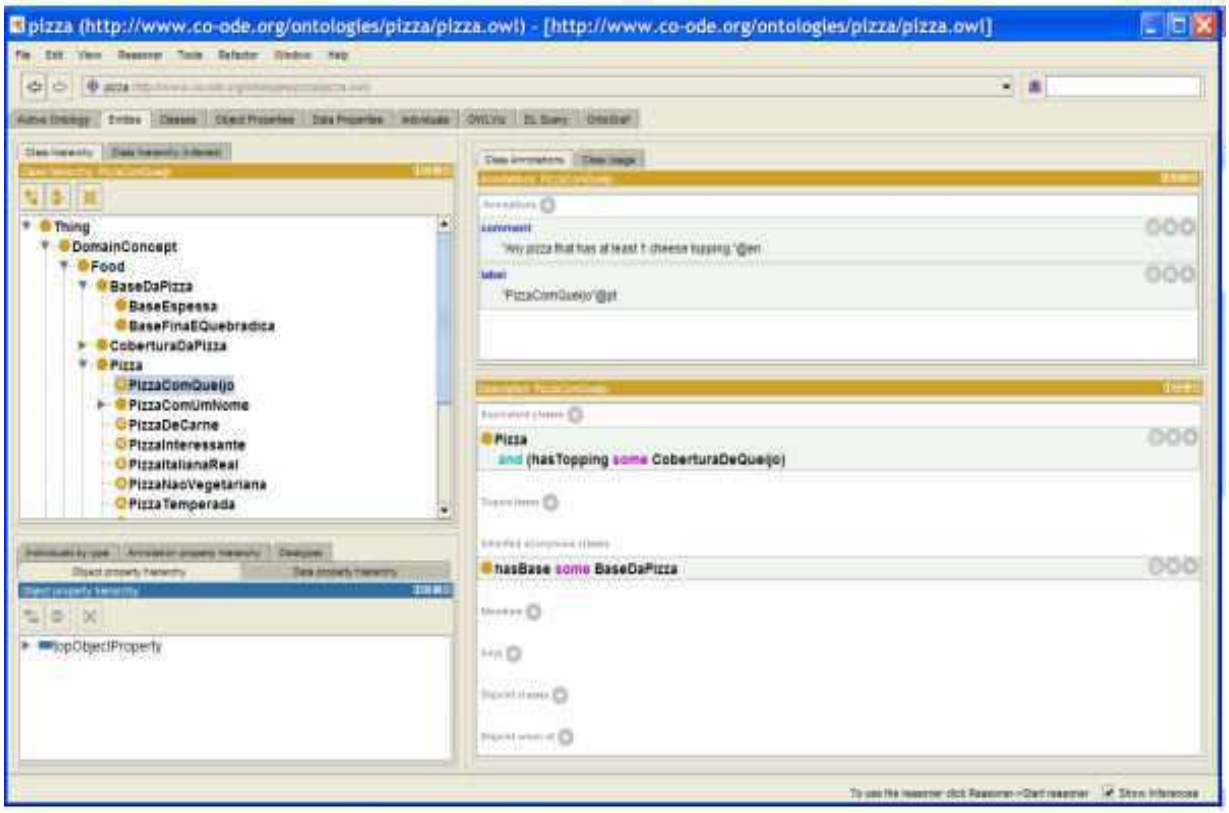


Figura 7: Interface gráfica do *Protégé* (Fonte: [6]).

2.3.3 API JENA

A API Jena é um framework desenvolvido pelo núcleo de pesquisa de *Web Semântica* da HP *(Hewlett Packard)*. Utilizando-se da linguagem Java, esta API oferece suporte para linguagens de *Web Semântica* que manipulam: RDF, RDFS, OWL e DAM+OIL [12].

No projeto *Gerador Automático de Cenários,* são utilizadas duas ontologias, uma como repositório local de um cenário de treinamento, e a outra como um repositório local de um cenário de erro. A API Jena é utilizada na criação de duas fachadas, capazes de se comunicar com as duas ontologias de domínio.

2.4 RELATÓRIOS RDFH E CLASSIFICAÇÃO DO ERRO

No contexto deste trabalho, fundamentaremos o uso da ontologia de domínio de um cenário de erro, e sua tipologia, com base na tese de doutorado de Serey [13].

Aqui, abordaremos o conceito de erro humano, a etiologia do erro humano e sua classificação, com o intuito de propor uma ontologia de conceitos que contemple a análise do erro humano.

Quando um acidente provocado por falha material ou erro humano acarreta em perda de partes ou de todo o sistema, e/ou perdas de vidas humanas e/ou perdas financeiras em um sistema, diz-se que este sistema é um **sistema crítico.** O erro humano tem sido uma das principais causas desses acidentes [13].

No caso de operadores de sistemas elétricos, situações imprevistas exigem um raciocínio rápido seguido de tomada de decisões importantes em um curto intervalo de tempo. Ou seja, citando Serey [13]: “Tais situações elevam a sua carga cognitiva e o nível de estresse, propiciando ainda mais a ocorrência de erros e provocando assim incidentes e acidentes”.

2.4.1 ERRO HUMANO

A noção sobre erro humano é ampla e seu conceito muito vasto por ser visto de várias óticas (psicologia, ergonomia, filosofia, etc.) [13].

Todo erro pode ser considerado humano, uma vez que este se encontra dentro de um contexto sócio-técnico. Portanto a origem do erro não tem origem exclusivamente humana, mas decorre da relação entre o ser humano e máquina. Ou seja, só existirá erro

quando o objetivo desejado após uma sequência de atividades – seja ela mental ou física – não é alcançado [13].

Porém, quando um operador se encontra diante de uma situação, na qual ele não tem liberdade de escolha, não existe possibilidade de erro por parte do operador, e se ainda assim ocorrer o erro, sua causa deste não deve ser imputada ao operador. Conclui-se assim que há diversas definições e interpretações do erro humano [13].

Etiologia do Erro Humano

Do ponto de vista abordado, podemos fazer uma distinção entre dois tipos de erro [13]. Erros ativos – provenientes da interação entre o operador e o sistema, os efeitos desse tipo de erro são sentidos rapidamente. E, erros passivos – provenientes das atividades de terceiros (projetista, construtores, tomadores de decisão) de forma indireta com o sistema. Estes erros podem permanecer latentes por um longo período de tempo, até que combinados a um conjunto de fatores manifestam-se revelando falhas no sistema.

Enquanto os erros ativos permitem uma maior percepção do risco, os erros latentes são aqueles que têm maior probabilidade de ocasionar acidentes. Os operadores são treinados para responder bem aos erros ativos, quando estes já são conhecidos, enquanto são susceptíveis às latências de causas desconhecidas [13] apud [14].

Quando um operador se depara com uma situação conhecida, a tendência é buscar na memória um caso similar que lhe permita simplificar uma tarefa mais complexa, estratégia que em muitos casos se mostra suficiente para compreender a situação. Porém, diante de uma situação menos conhecida, ou de um caso mais complicado, a construção de uma representação desta situação demanda mais tempo [13] apud [14].

Este tipo de habilidade pode ser inconveniente, visto que existe uma dependência forte entre casos passados e o caso presente, podendo levar a novos erros.

Assim, o erro pode ocorrer a partir de certas características organizacionais restritas – tais como: baixa carga de trabalho, sobrecarga de trabalho, pressão temporal, estresse, falta de confiança e excesso de confiança [13].

Desta forma, a classificação do erro, de acordo com o processo de tomada de decisão do operador, pode evitar erros futuros, desde que na ocorrência de uma falha, as informações referentes ao erro estivessem armazenadas em uma ontologia, possibilitando realizar treinamentos de operadores de formas mais eficiente.

Relatório RDFH e a classificação do Erro

O Relatório Desligamento por Falha Humana trata de acidentes nos quais houve "uma mudança de estado de equipamento com perda de sua função associada, causada por uma ação humana indevida" [13].

O uso de ontologia de domínio para classificar o erro humano foi proposto por Serey, durante a análise de 18 relatórios RDFH, que resultou em uma tipologia dos erros analisados.

Tabela 1: Categorização proposta do erro.

Categoria geral	Categoria específica
Observação do Estado do Sistema	Excessiva
	Falsa interpretação
	Incorreta
	Incompleta
	Inapropriada
	Ausente
Escolha de uma hipótese	Inconsistente em relação à observação
	Consistente, mas pouco provável
	Consistente, mas muito custosa
	Funcionalmente não pertinente
Avaliação de uma Hipótese	Incompleta
	Aceitação de uma hipótese errada
	Rejeição de uma hipótese certa
	Ausente
Escolha do procedimento	Incompleto
	Incorreto
	Supérfluo
	Ausente
Execução	Ação omissa
	Ação repetida
	Acréscimo de uma operação
	Operação fora de sequência
	Intervenção em tempo não apropriado

	Posição da operação incorreta
	Execução incompleta
	Ação sem relação e inapropriada

Fonte: Serey [13].

A partir das informações sobre cenário de treinamentos, log de incidentes, a estrutura de ontologias e a classificação do erro, é possível construir uma ferramenta que contemple as características aqui discutidas, e no próximo capítulo discutiremos o processo de implementação do nosso trabalho.

3 DESENVOLVIMENTO

O módulo Gerador automático de cenários, desenvolvido ao longo deste projeto, foi especificado a partir de diagramas em UML - *Unified Modelling Language*. Neste capítulo discutiremos a arquitetura do software, a organização e definição de cada módulo, assim como serão apresentados os diagramas de caso de uso e de classes, ambos na notação UML.

3.1 ARQUITETURA DO SISTEMA

A arquitetura do sistema foi implementada a partir da concepção das funcionalidades do sistema. A especificação dessas funcionalidades resultou dos objetivos do trabalho, e são listadas a seguir, na forma de requisitos funcionais:

- Ler um arquivo de *log* (.aud) e extrair eventos temporais de abertura de disjuntores que configurem uma ocorrência específica – *Sigla Funct1*;
- Ler e salvar informações nas Ontologias Cenário de Treinamento e Cenário de Erro – *Sigla Fucnt2*;
- Exibir uma interface Homem-Máquina capaz de coletar informações referentes ao relatório RDFH, que sejam relevantes à construção das Ontologias citadas acima, e exibir os eventos extraídos do arquivo de *log* – *Sigla Fucnt3*;
- Produzir um arquivo de texto puro capaz de ser importado para o SIMULOP – *Sigla Fucnt4*.

Partindo desses requisitos funcionais, adotamos a arquitetura da figura 8 para representar o sistema Gerador Automático de Cenários.

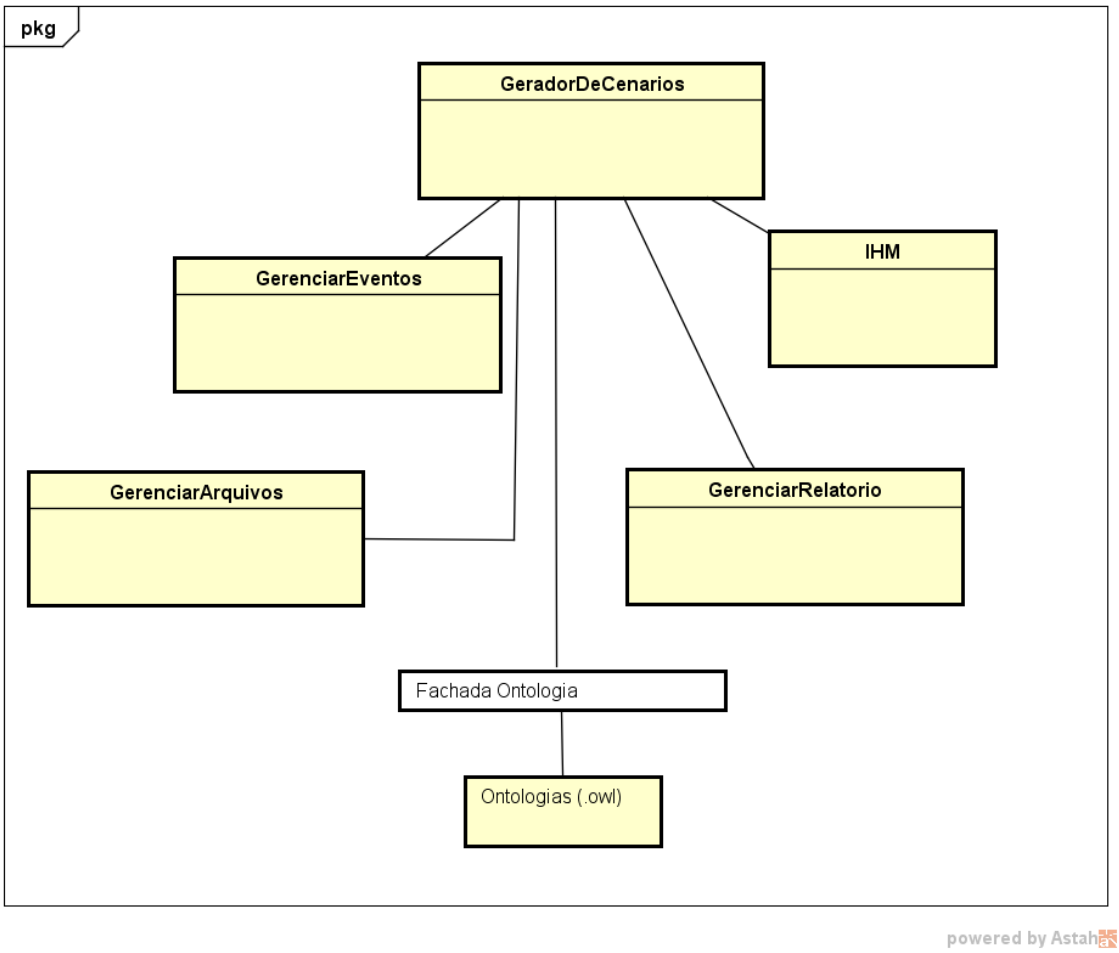


Figura 8: Representação esquemática da arquitetura do sistema gerador automático de cenários (fonte: o Próprio autor).

- GeradorDeCenarios: responsável pelo controle dos demais módulos. Sua principal função é controlar os componentes de interface gráfica, gerenciar a comunicação entre os demais módulos GerenciarEventos, IHM e GerenciarRelatorio com a fachada da ontologia e fornecer ao modulo gerenciarArquivos os objetos necessários para efetuar a leitura dos arquivos do tipo (.aud) e gera o arquivo de texto puro.
- GerenciarEventos: possui uma estrutura de dados para a comunicação com a Ontologia Cenário de Treinamento, com o objetivo de realizar o *parser* dos dados coletados pelo módulo GerenciarArquivos para o padrão contido na Ontologia. Esse módulo fornecerá esses objetos para serem tratados na interface gráfica e salvos na ontologia.

- IHM: este módulo fornece uma interface gráfica para edição de um relatório RDFH, visualização dos eventos contidos no módulo GerenciarEventos e manipulação de arquivos.
- GerenciarRelatorio: estrutura os objetos do relatório RDFH no padrão da Ontologia Cenário de Erro. Seus objetos serão também objetos da IHM, e serão salvos na ontologia.

O diagrama de caso de uso adotado está representado na figura 9. A partir do diagrama de caso de uso, é possível identificar as principais interações entre o usuário e o sistema, assim como a sequência de tarefas que devem ser executadas para atingir o caso de uso específico.

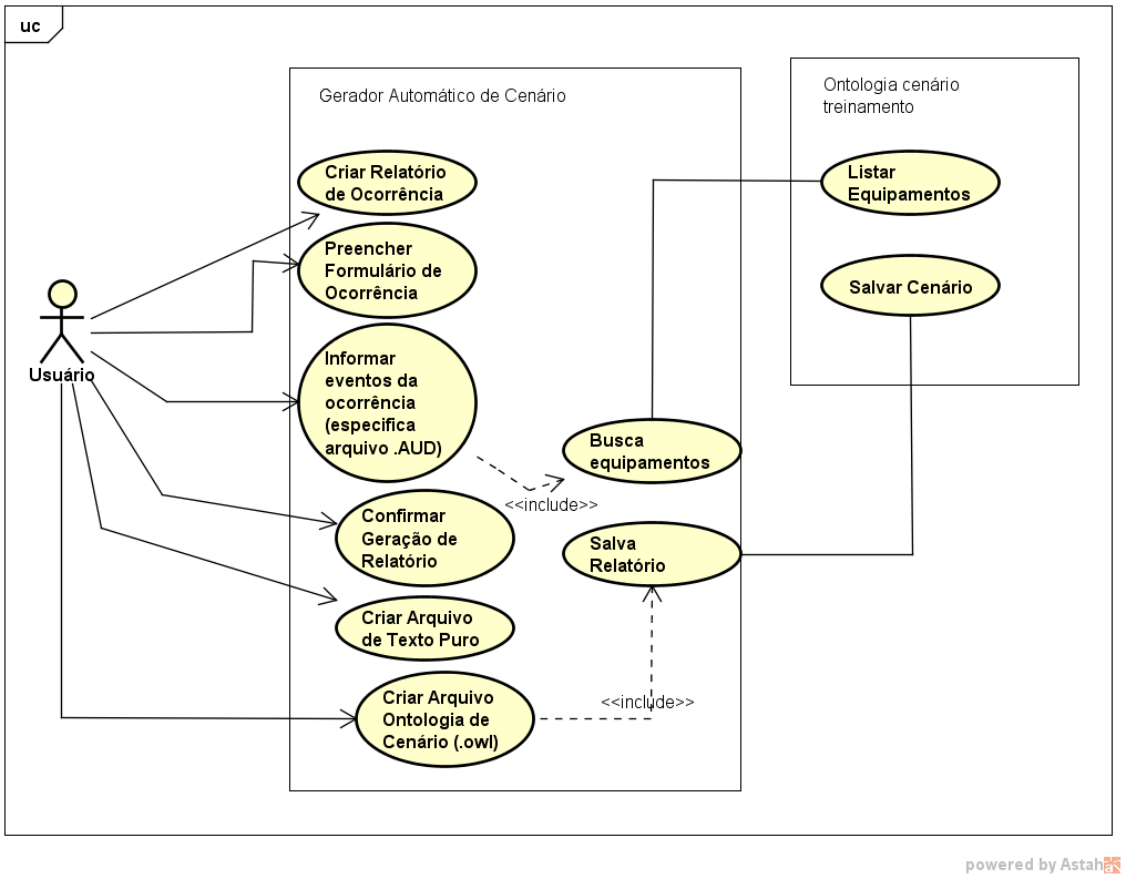


Figura 9: Caso de uso do gerador automático de cenários (Fonte: o própio autor).

3.2 ESPECIFICAÇÃO UML

O diagrama de classes é uma ferramenta importante na estruturação de um software, fornecendo ao implementador a relação entre os objetos que o compõe, modularizando e facilitando a modelagem do sistema.

A partir da arquitetura do sistema e, dos casos de usos, é possível especificar os pacotes e as classes necessárias para a aplicação. A tabela 2 contém uma associação entre os casos de uso e os pacotes de classe utilizados na implementação do software.

Tabela 2: Relação entre os casos de uso e os pacotes do sistema gerador automático de cenários.

Sigla caso de uso	**Caso de Uso**	**Pacote associado**
UserCase1	Criar Relatório de Ocorrência	*geradorautomaticodecenarios*
UserCase2	Preencher Formulário de Ocorrência	*ihm*
UserCase3	Informar eventos da ocorrência	*geradorautomaticodecenarios*
UserCase3.1	Buscar Equipamentos	*ontologia*
UserCase4	Confirmar Geração de Relatório	*ihm*
UserCase5	Criar arquivo de texto puro (SAGE)	*gerenciadordearquivos*
UserCase6	Criar arquivo ontologia de cenário (.owl)	*ontologia*

Fonte: O próprio autor.

A organização dos pacotes e das classes que realizam a implementação dos casos de uso está representada na figura 10. Cada pacote, e a relação entre suas classes, serão discutidos ao longo desta capítulo.

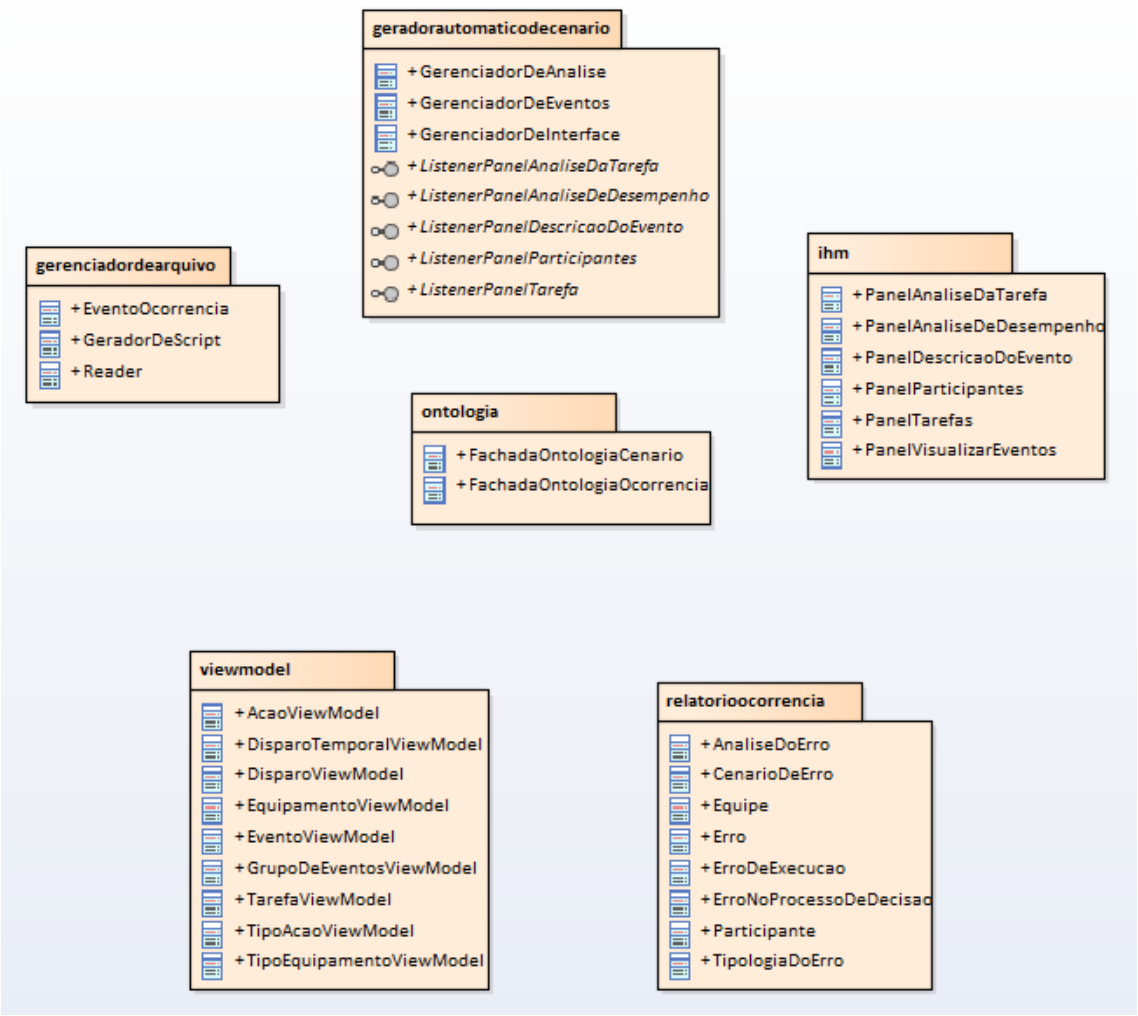


Figura 10: Diagrama de pacotes do sistema gerador automático de cenários (fonte: O próppio autor).

O pacote ***geradorautomaticodecenario*** é o pacote principal do sistema. Ele contém os gerenciadores que fornecerão as aplicações necessárias para manipulação de dados e a interface gráfica do software. Os relacionamentos das classes pertencentes a este grupo são apresentados na figura 11. Uma característica importante deste pacote é a existência de classes do tipo *interface* – esses tipos são *listeners (Ouvinte ou Auditor em português)*. Essas interfaces serão associadas aos elementos de interface gráfica, facilitando a comunicação entre os gerenciadores e os objetos de interface. Os *listeners* serão implementados pelos gerenciadores a fim de modularizar ainda mais o sistema. Eles definem as atividades relacionadas a aplicação (funcionalidades de sistema), tornando os componentes de interface livres de obrigações relativas à aplicação, mantendo-os responsáveis apenas pelas aplicações gráficas.

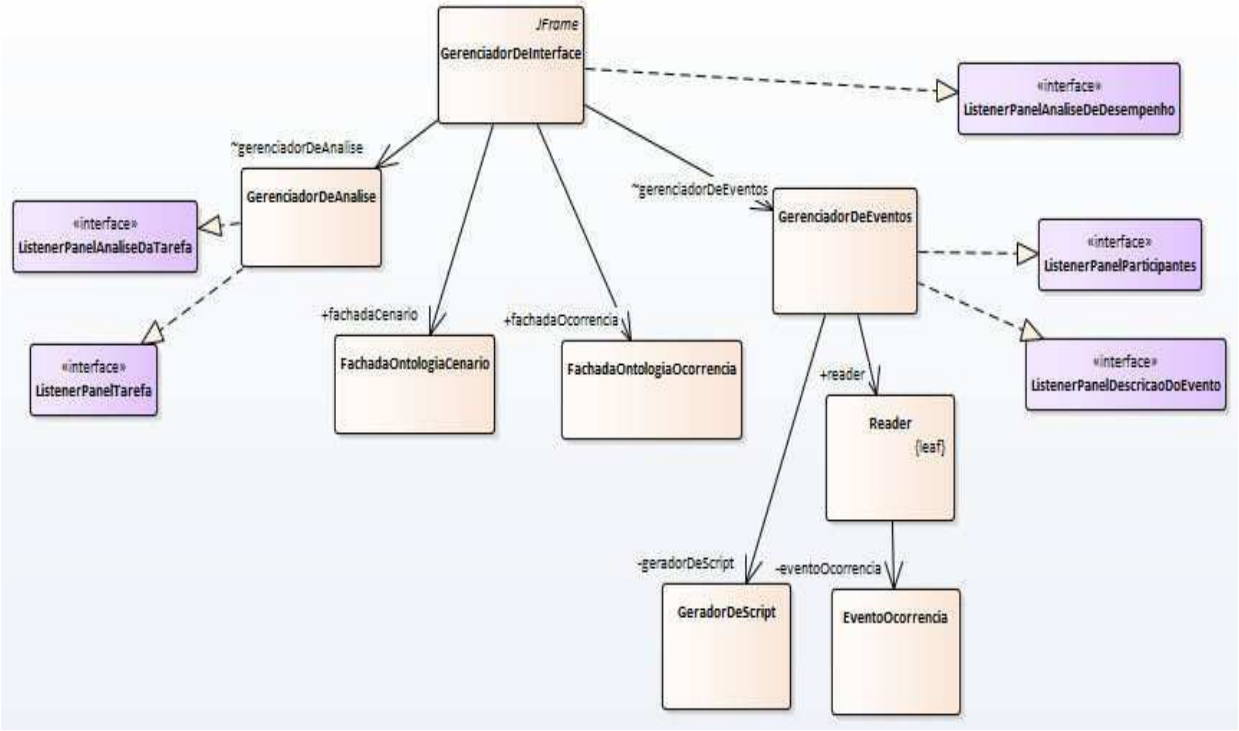


Figura 11: Diagrama de classes simplificado (omite-se os atributos e os métodos de cada classe) do pacote "Gerador Automático de Cenário" (Fonte: o próprio autor).

O pacote ***ihm*** contém os componentes de interface gráfica do sistema, que permitirá a interação do usuário com o software. Seu diagrama de classes é apresentado na figura 12, onde se observa uma dependência entre este pacote e o pacote principal ***geradorautomaticodecenario***.

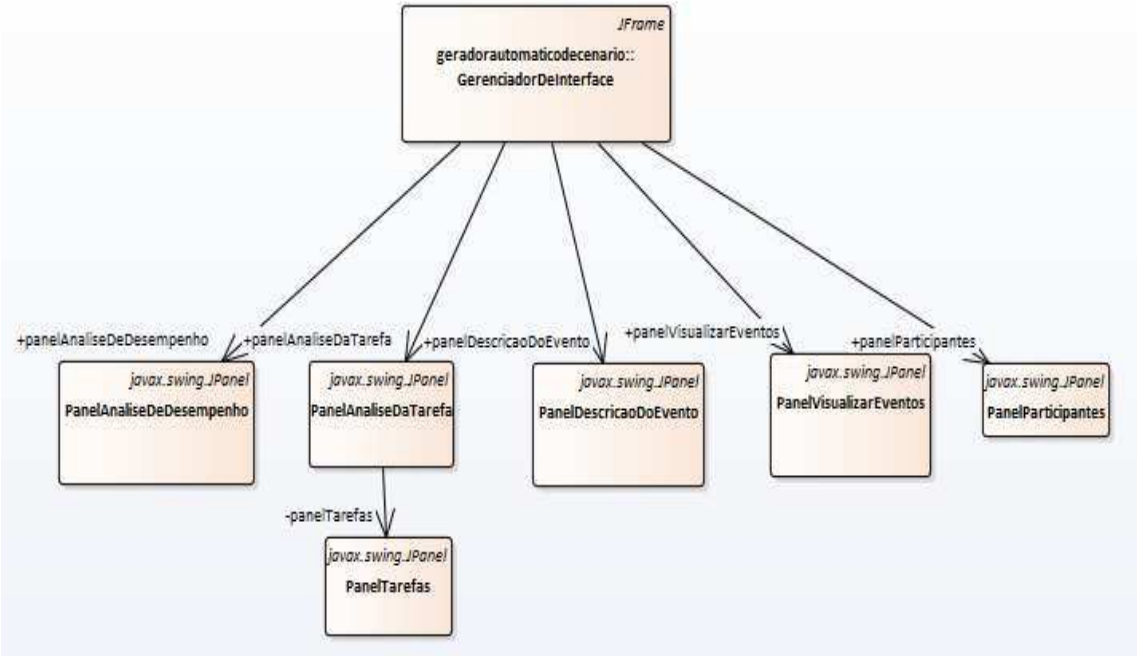


Figura 12: Diagrama de classes simplificado do pacote ihm (Fonte: O próprio autor).

O pacote ***gerenciadordearquivo*** implementa as funcionalidades referentes a leitura e escrita de arquivos de texto. Este pacote contém uma estrutura de dados própria para facilitar a leitura do arquivo de auditagem *(.aud)*, *EventoOcorrencia*, que será tratado pelo *GerenciadorDeEventos* do pacote ***geradorautomaticodecenario***. Seu diagrama de classes é apresentado na figura 13.

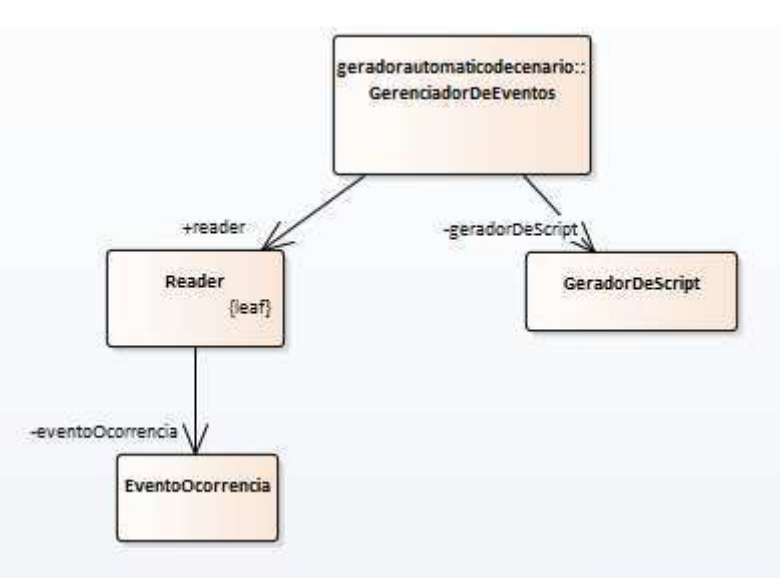


Figura 13: Diagrama de classes simplificado do pacote "Gerenciador de Arquivo" (fonte: o próprio autor).

O pacote ***ontologia*** contém as duas fachadas que se comunicarão com as Ontologias *CenarioDeTreinamento* e *CenarioDeErro*. As informações contidas na interface gráfica serão posteriormente salvas nessas ontologias. As duas fachadas foram representadas no diagrama de classes da figura 11 (geradorautomaticodecenario) por serem fundamentais na realização de suas funcionalidades, porém, elas pertencem ao pacote ***ontologia***, e seu diagrama é apresentado a seguir (Figura 14).

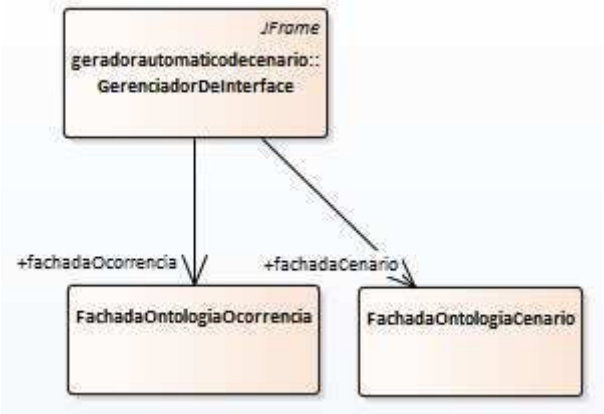


Figura 14: Diagrama de classes simplificado do pacote ontologia, mostrando a relação deste com o pacote "Gerador Automático de Cenário" (fonte: o próprio autor).

O pacote ***viewmodel*** foi reaproveitado do projeto *Gerador de Cenário para Treinamento de Operadores*, por abstrair as informações necessárias para a construção do objeto *Grupo De Eventos*, assim como as informações referentes às *tarefas* do cenário.

O pacote ***relatorioocorrencia*** corresponde aos objetos de interface gráfica referente à ontologia *Cenário De Erro*. Juntamente com o ***viewmodel***, esses dois pacotes contemplam a estrutura de dados do sistema *Gerador Automático de Cenários,* sendo sua estrutura interna discutida ao longo da implementação dos demais módulos.

3.3 IMPLEMENTAÇÃO

Neste capítulo discutiremos a interface gráfica e as principais classes do sistema *Gerador Automático de Cenários*, detalhando seus principais métodos.

3.3.1 PROJETO DA INTERFACE GRÁFICA

Um dos objetivos do *Gerador Automático de Cenários* é fornecer uma interface gráfica de acordo com a *Fucnt3*. Dessa forma, foi levantada uma lista de requisitos para garantir a validação da interface.

i. Tomando como base as Ontologias: *Cenário de Erro* e *Cenário de Treinamento,* e o relatório RDFH representa graficamente os elementos que estejam presentes no relatório RDFH e que sejam relevantes para a construção das ontologias;
ii. Deve ser possível ao usuário visualizar os eventos de ocorrência;
iii. A interface tem que ser autoexplicativa;

Para atender ao primeiro requisito, foi necessário consultar os elementos presentes na ontologia *Cenário de Erro*, proposta por Torres Filho em [6] – ver figura 15 – e mapeá-los com as informações presentes em um relatório RDFH. A figura 15 apresenta as relações da classe *Ontologia_Cenario_De_Erro* com seus objetos. Esta classe possui três subclasses: *Equipe, Erro e Tipo de Turno*. As figuras 16, 17 e 18 apresentam as relações entre as classes: *Equipe, Erro, Tipo de Turno* e seus objetos, respectivamente.

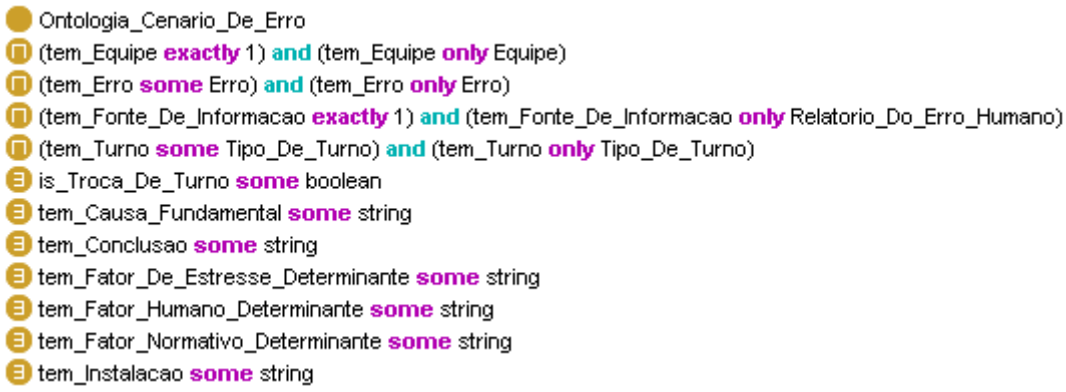


Figura 15: Visualização das Propriedades e expressões da ontologia cenario de erro através do protégé (fonte: O Próprio autor).

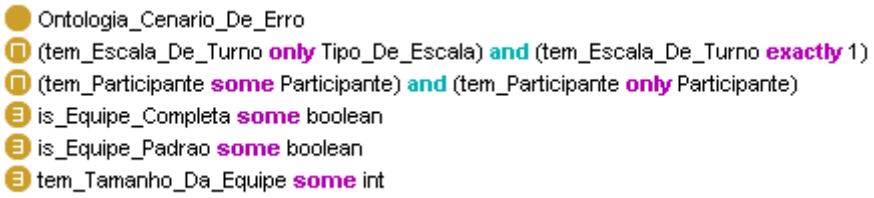


Figura 16: Visualização das propriedades e expressões da subclassse equipe através do protégé (Fonte: O Próprio autor).

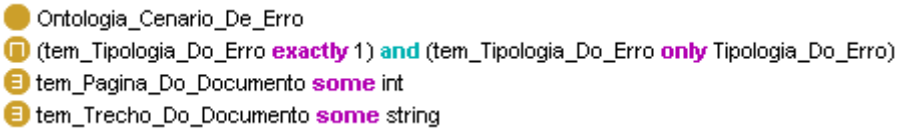


Figura 17: Visualização das propriedades e expressões da subclassse Erro através do protégé (Fonte: O Próprio autor).

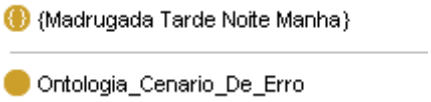


Figura 18: Visualização das propriedades e expressões da subclasse *Tipo de* Turno através do Protégé (Fonte: O próprio autor).

A classe *Tipologia do Erro* é definida por três subclasses, que são ilustradas na figura 19. Cada subclasse é associada a um ou mais tipos de subclasse da *Categoria Do Erro*. A figura 20 apresenta as subclasses desta categoria e a relação destas as quais são apresentadas nas figuras 21, 22 e 23. A subclasse *Categoria Específica* contém os indivíduos que definem todas as subclasses de *Categoria do Erro*.

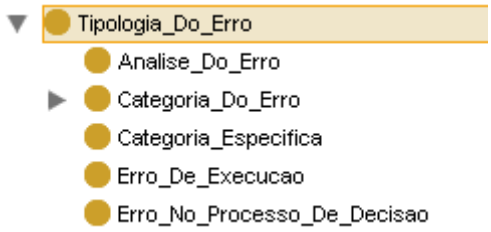


Figura 19: Visualização das subclasses de tipologia do erro através do *Protégé* (Fonte: O Próprio autor).

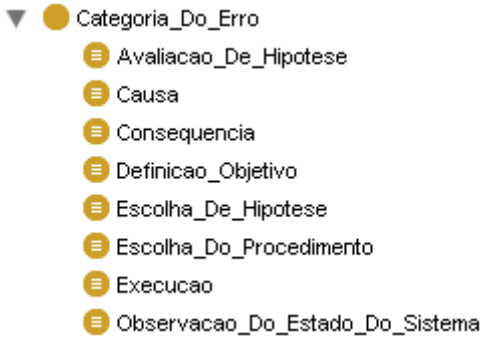


Figura 20: Visualização das subclasses de Categoria do Erro através do *Protégé* (Fonte: O próprio autor).

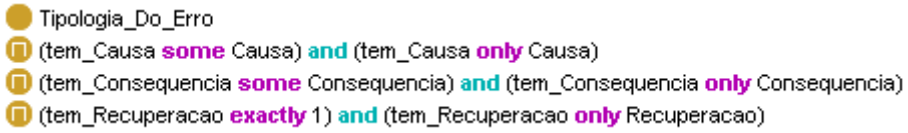


Figura 21: Visualização das propriedades e expressões da subclasse de tipologia do erro, analise do erro, através do *Protégé* (fonte: o próprio autor).

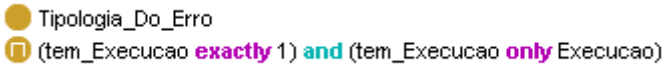


Figura 22: Visualização das propriedades e expressões da subclasse de tipologia do erro, Erro de execução, através do *Protégé* (fonte: o próprio autor).

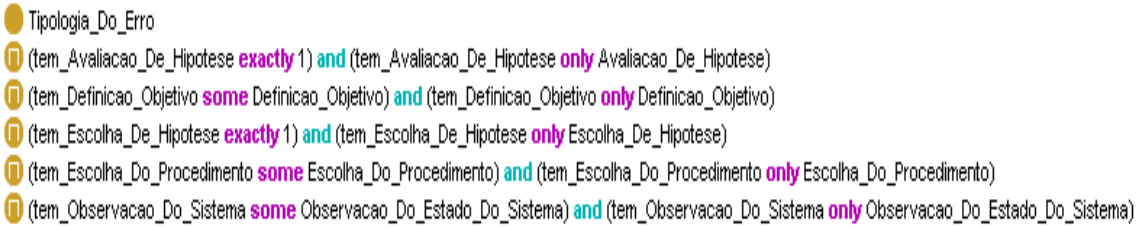


Figura 23: Visualização das propriedades e expressões da subclasse de tipologia do erro, Erro no processo de decisão, através do *Protégé* (fonte: o próprio autor).

O mapeamento de informações é um processo que consiste em analisar o documento RDFH e associar os objetos ali contido com os objetos da ontologia cenário de erro, verificando se todos os objetos relevantes estão presente em ambos. Dessa forma, a tabela 3 apresenta o mapeamento das sessões presentes no relatório RDFH com

os objetos da classe *Ontologia_Cenario_De_Erro*. Para ilustrar o mapeamento, foi selecionado o relatório RDFH RdehGRN042010.

Tabela 3: Tabela de mapeamento entre o relatório rdfh e a ontologia cenário de erro.

Capítulo do RDFH	Objeto (s) da Ontologia	Trecho do relatório
Capítulo 5, *Análise do Desempenho Humano*	Tem causa fundamental	"*Análise incorreta da configuração da instalação após ocorrência e nivelamento de informações deficiente entre operadores – **causa fundamental***"
Capítulo 10, *Conclusões*	Tem conclusão	Não necessário.
Capítulo 4, *Análise da Tarefa*	Tem fator de estresse determinante	*"**Fatores determinantes:** ... **Stressores** – Tarefas Complexas"*
Capítulo 4, *Análise da Tarefa*	Tem fator humano determinante	*"**Fatores determinantes**: ... **Homem** – Treinamento/Prática"*
Capítulo 4, *Análise da Tarefa*	Tem fator normativo determinante	"***Fatores determinantes: Normativo** – Não cumprimento dos procedimentos da IN–OP.01.006 - Reenergização de Equipamentos e Linhas de Transmissão"*
Capítulo 1, *Descrição do Evento*	Tem instalação	"*1. DESCRIÇÃO DO EVENTO **SE: Delmiro Gouveia***"
Capítulo 5, *Análise do Desempenho Humano*	Tem Erro	(a) Causa: "*... não identificaram corretament*

				e a configuração da instalação ...” Consequência: *“... desenergizando o barramento 04B1 às 05h35 ...*” Recuperação: *Não informada.*
	Analise Do Erro (a)	Erro de Execucao (b)	Erro no Processo de Decisão (c)	(b) Execução: “*... ao invés de realizar a recomposição ...*”
				(c) Avaliação de Hipótese: “*...e embora tenha constatado que não havia desligamento geral ... seguindo o guia de reenergização ... ao invés de realizar a recomposição parcial...* ” Definição do Objetivo: *“... recomposição parcial ...”* Escolha de

<table>
<tr><td></td><td></td><td></td><td></td><td>Hipótese: “..., iniciou a recomposição da instalação”
Observação do Sistema: “... e embora tenha constatado que não havia desligamento geral”</td></tr>
<tr><td rowspan="2">Capítulo 3.2, Informações Complementares</td><td colspan="5">Tem Equipe</td><td rowspan="2">“...composto por 02 (dois) operadores ... A escala adotada ... é a 4x2 mista ... experientes com 25 e 12 anos ...”</td></tr>
<tr><td>Escala de Turno</td><td>Tem Participante</td><td>Equipe Completa</td><td>Núm. De Integrantes</td><td>Equipe Padrão</td></tr>
<tr><td>Capítulo 1, Descrição do Evento</td><td colspan="5">Tipo de Turno</td><td>“Horário: 05h35”</td></tr>
</table>

Fonte: O próprio autor.

Após realizar o mapeamento, cada objeto da ontologia foi associado ao componente gráfico correspondente. Essa associação está representada na tabela 4.

Tabela 4: Relação entre objetos da ontologia e componentes gráficos para a ihm.

Objeto da Ontologia	Componente Gráfico
Causa Fundamental	Campo de Texto
Conclusão	Campo de Texto
Fator de Estresse Determinante	Campo de Texto
Fator Humano Determinante	Campo de Texto
Fator Normativo Determinante	Campo de Texto
Instalação	Caixa de Seleção
Erro – Análise do Erro	Caixa de Seleção
Erro – Erro de Execução	Caixa de Seleção
Erro – Erro no processo de Decisão	Caixa de Seleção

Equipe – Escala de turno	Caixa de Seleção
Equipe – Participante	Tabela
Equipe Completa	Caixa de Marcação
Número de Integrantes	Caixa de Texto
Equipe Padrão	Caixa de Marcação

Fonte: O próprio autor.

A partir do que já foi discutido sobre eventos no OTS, na capítulo 2.1, decidiu-se organizar a interface dos eventos em uma tabela, na qual os elementos discutidos em 2.1 pudessem ser representados.

A figura 24 representa a organização e a relação entre os objetos da ontologia *Cenário de Treinamento*, proposta por Torres Filho em [6].

Figura 24: Visualização das propriedades e expressões da classe cenário de treinamento, da ontologia cenário de treinamento, através do *Protégé* (fonte: o próprio autor).

A tabela 5 apresenta o mapeamento entre esta ontologia e o relatório RDFH.

Tabela 5: Mapeamento entre o relatório RDFH e a ontologia cenário de treinamento.

Objetos da ontologia cenário de treinamento	Trecho do relatório RDFH
Configuração Depois do Cenário	“Demais equipamentos da subestação encontravam-se na configuração normal”.

Station	“SE: Delmiro Gouveia”
Tarefa	“Tipo de tarefa – Complexa, emergência e rara. Objetivo da tarefa executada – Recomposição da instalação/equipamento. ”

Fonte: O próprio autor.

A configuração do sistema após o cenário é a mesma configuração inicial apresentada na tabela 4. Para representar a tarefa, foi utilizado o componente gráfico do tipo tabela – adequado para representar todos os objetos relacionados à tarefa.

As figuras 25, 26 e 27 ilustram a interface proposta.

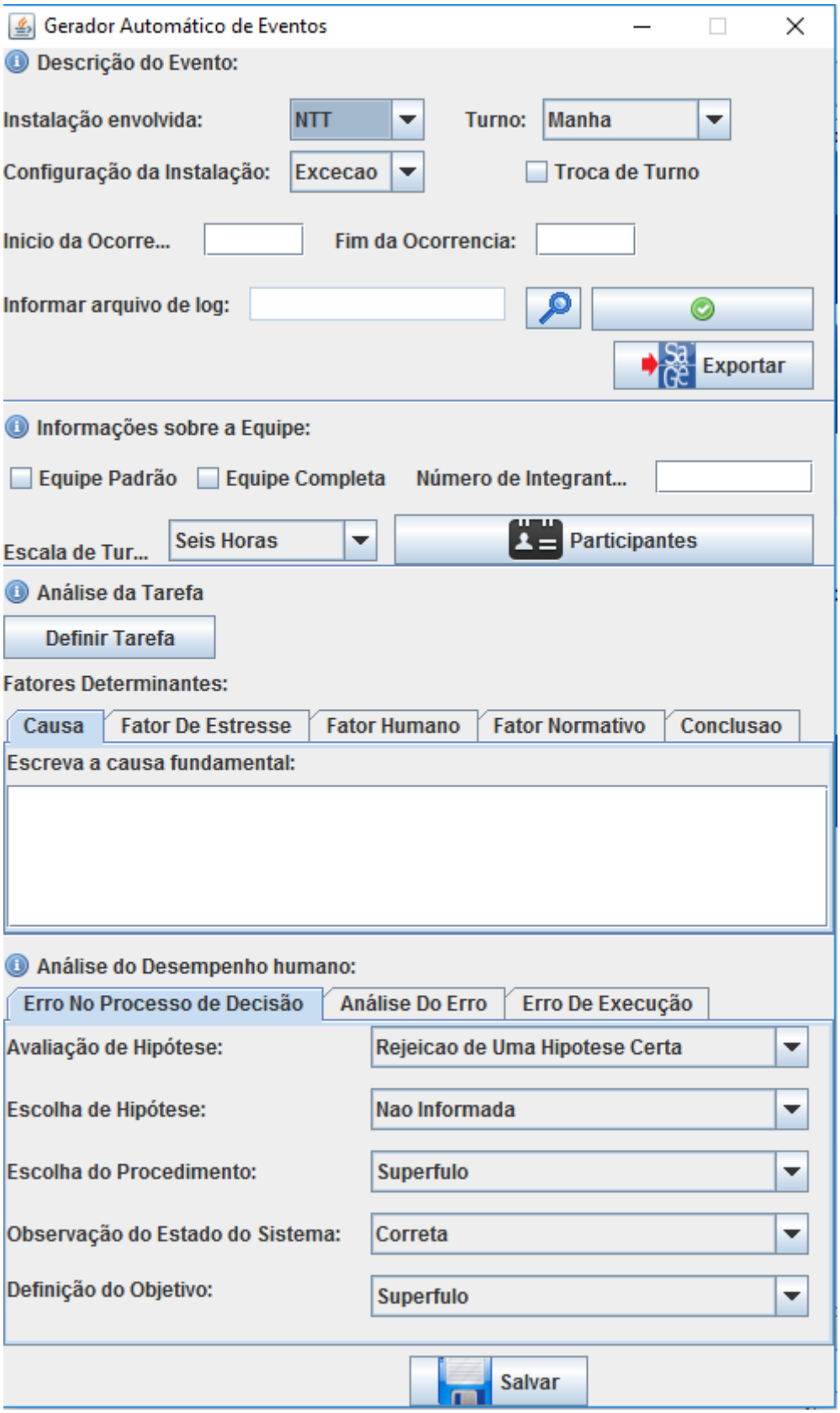


Figura 25: IHM do *Gerador Automático de Cenários* (Fonte: O próprio autor).

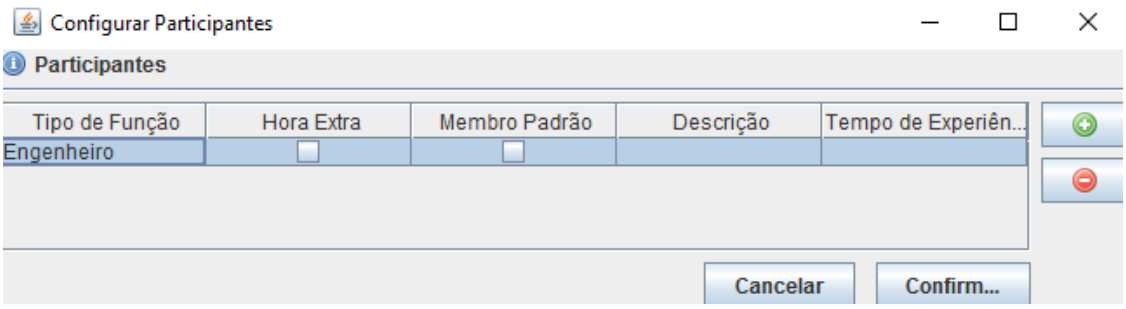


Figura 26: IHM de edição de participantes (fonte: o próprio autor).

Figura 27: IHM de Visualização de eventos (Fonte: O próprio autor).

3.3.2 CLASSE GERENCIADOR DE INTERFACE

Esta é a classe de maior importância para o sistema proposto. Nela, serão instanciadas as interfaces gráficas, construção dos objetos que compõem tanto o cenário de erro quanto o cenário de treinamento. Com o intuito de deixar o sistema o mais modular possível, definimos os objetos de gerência de eventos e análise de relatório nesta classe.

O gerenciador de interface implementa o *listener AnaliseDeDesempenho*, que fornecerá a importante função de salvar um cenário de erro na Ontologia de domínio de aplicação *Cenário De Erro*. Esta classe é do tipo *JFrame.* Em Java, o tipo *JFrame* é utilizado para organizar objetos de interface do tipo *JPanel*¸ e é responsável por inicializar o sistema como um todo (esta classe é executada como uma *main class*).

Seu diagrama de classe completo é apresentado na figura 28. Os métodos principais relacionados às funcionalidades do sistema, aquelas que foram discutidas na capítulo 3.1, serão descritos a seguir.

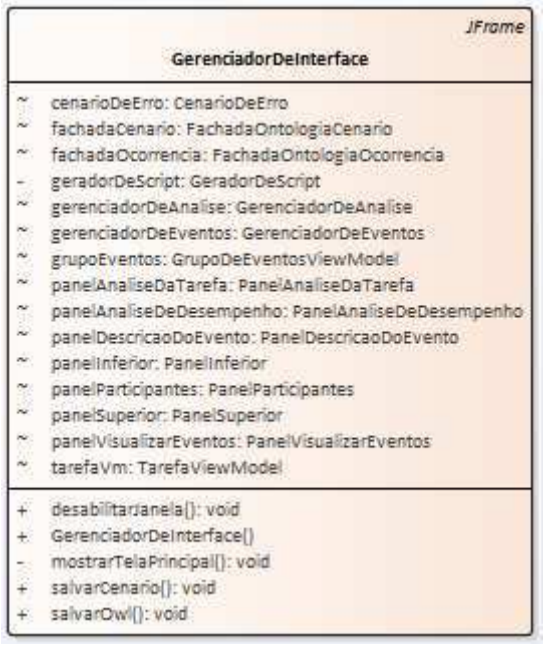


Figura 28: Diagrama de classe do gerenciador de interface (fonte: o próprio autor).

mostrarTelaPrincipal

Método do tipo *private*, só pode ser chamado pelo Gerenciador de Interface. Este método é responsável por organizar os objetos da interface gráfica, mantendo uma estrutura de *layout* que organiza os painéis (*JPanel das classes de interface gráfica*). Este método implementa a funcionalidade *Funct3* do sistema.

salvarCenario e ***salvarOwl***

Estes dois métodos salvam as informações referentes ao relatório de ocorrência. O primeiro coleta as informações nos painéis de interface gráfica referentes ao relatório de ocorrência, enquanto o segundo implementa o método do *listener* do painel de análise do relatório. Este último é chamado quando o usuário decide salvar o relatório de ocorrência e este, por sua vez, aponta para o método ***salvarCenario***.

A figura 29 apresenta o trecho de código do método ***salvarCenario***. Este método é responsável por implementar a funcionalidade *Funct2* do sistema.

```
public void salvarCenario ()
{
    this.cenarioDeErro = new CenarioDeErro();
    this.cenarioDeErro.setCausaFundamental(gerenciadorDeAnalise.getPanelAnaliseDaTarefa().getCausaFundamental());
    this.cenarioDeErro.setFatorDeEstresse(gerenciadorDeAnalise.getPanelAnaliseDaTarefa().getFatorEstresse());
    this.cenarioDeErro.setFatorHumano(gerenciadorDeAnalise.getPanelAnaliseDaTarefa().getFatorHumano());
    this.cenarioDeErro.setFatorNormativo(gerenciadorDeAnalise.getPanelAnaliseDaTarefa().getFatorNormativo());
    this.cenarioDeErro.setConclusao(gerenciadorDeAnalise.getPanelAnaliseDaTarefa().getConclusao());
    Erro erro = new Erro();
    AnaliseDoErro analise = new AnaliseDoErro();
    analise.setTemCausa(gerenciadorDeAnalise.getPanelAnaliseDesempenho().getCausa());
    analise.setTemConsequencia(gerenciadorDeAnalise.getPanelAnaliseDesempenho().getConsequencias());
    analise.setTemRecuperacao(gerenciadorDeAnalise.getPanelAnaliseDesempenho().getRecuperacao());
    ErroNoProcessoDeDecisao erroDecisao = new ErroNoProcessoDeDecisao();
    erroDecisao.setAvaliacaoDeHipotese(gerenciadorDeAnalise.getPanelAnaliseDesempenho().getAvHipotese());
    erroDecisao.setDefObjetivo(gerenciadorDeAnalise.getPanelAnaliseDesempenho().getDefinicaoObjetivo());
    erroDecisao.setEscolhaDeHipotese(gerenciadorDeAnalise.getPanelAnaliseDesempenho().getEscolhaHipotese());
    erroDecisao.setEscolhaDeProcedimento(gerenciadorDeAnalise.getPanelAnaliseDesempenho().getEscolhaProcedimento());
    erroDecisao.setEstadoDoSistema(gerenciadorDeAnalise.getPanelAnaliseDesempenho().getEstadoSistema());
    ErroDeExecucao erroExecucao = new ErroDeExecucao(gerenciadorDeAnalise.getPanelAnaliseDesempenho().getExecucao());
    TipologiaDoErro tipologia = new TipologiaDoErro();
    tipologia.setTemAnaliseDoErro(analise);
    tipologia.setTemErroDeExecucao(erroExecucao);
    tipologia.setTemErroNoProcessoDeDecisao(erroDecisao);
    erro.setTipologiaDoErro(tipologia);
    this.cenarioDeErro.setErro(erro);
    this.cenarioDeErro.setEquipe(gerenciadorDeEventos.getEquipe());
    this.cenarioDeErro.setConfiguracaoInicial(gerenciadorDeEventos.getPanelGeradorDeEventos().getConfiguracaoInicial());
    this.cenarioDeErro.setIsTrocaDeTurno(gerenciadorDeEventos.getPanelGeradorDeEventos().isTrocaDeTurno());
    this.cenarioDeErro.setEscalaDeTurno(gerenciadorDeEventos.getPanelGeradorDeEventos().getTurno());
    this.grupoEventos = gerenciadorDeEventos.getGrupoDeEventos();
    this.tarefaVm = gerenciadorDeAnalise.getTarefaViewModel();
    try {
        fachadaCenario.saveOntology(grupoEventos, tarefaVm,
                "C:\\Users\\Andre GMS\\Documents\\NetBeansProjects\\GeradorAutomaticoDeCenario\\cenarioTrein.owl");
        fachadaOcorrencia.salvarRelatorioOcorrencia(cenarioDeErro,
                "C:\\Users\\Andre GMS\\Documents\\NetBeansProjects\\GeradorAutomaticoDeCenario\\cenErro.owl");
    } catch (Exception ex)
    {

    }
}
```

Figura 29: Trecho de código da classe gerador de interface, do método “Salvar Cenário” (Fonte: O próprio autor).

3.3.3 Classe Gerenciador de Eventos

O gerenciador de eventos implementa dois *listeners* para dois painéis de interface gráfica, sendo estes: o *ListenerPanelDescricaoDoEvento* e o *ListenerPanelParticipantes*, como foi apresentado na figura 11, no início deste capítulo (3).

Seus principais métodos são solicitar ao *GerenciadorDeArquivo* a leitura de um arquivo de auditagem, traduzir a estrutura de dados, dos eventos de ocorrência ali presentes, para a estrutura de dados que se comunicará com a ontologia – o pacote *viewmodel*; solicitar a visualização destes na interface gráfica *PanelVisualisarEventos* e solicitar ao *GerenciadorDeArquivo* a escrita dos eventos de ocorrência em um arquivo de texto puro.

Seu diagrama de classe completo é apresentado na figura 30.

Figura 30: Diagrama de classe do gerenciador de eventos (fonte: o próprio autor).

Os métodos principais desta classe estão relacionados com as *func1, funct3* e *funct4*. O método ***gerarEventosOcorrencia*** é responsável por chamar os métodos do Gerenciador De Arquivos os quais são responsáveis por realizar a operação de leitura de um arquivo de auditagem específico. A figura 31 ilustra o diagrama de classe do *Reader*, uma das classes do *Gerenciador de Arquivos* (ver figura 13). O método ***lerAud*** é responsável por realizar o algoritmo que seleciona os eventos relevantes. Quando o usuário fornece o intervalo de tempo que durou uma ocorrência especifica (ver figura 25), o método seleciona ou não determinado evento a partir do algoritmo descrito na figura 32.

Figura 31: Diagrama de classe do reader (Fonte: o próprio autor).

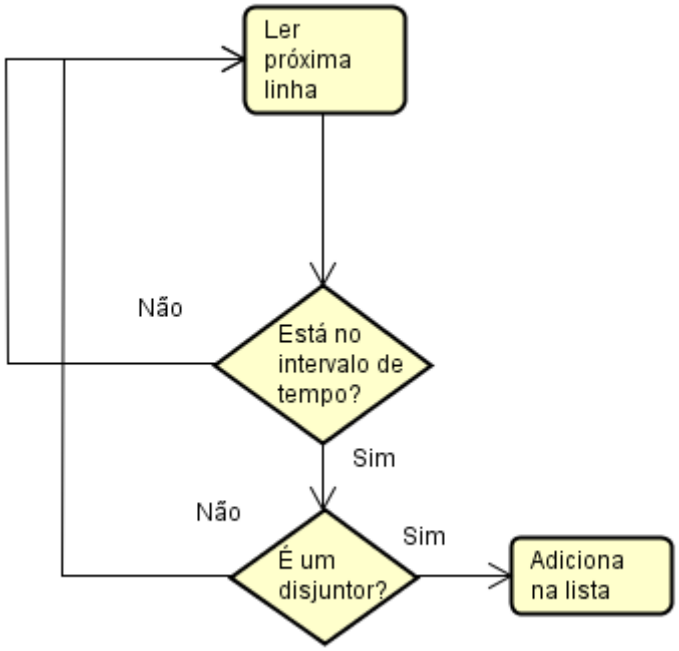


Figura 32: Diagrama representativo do algoritmo do método *lerAud* (Fonte: O próprio autor).

Os métodos ***parserEventosOcorrencia***, ***adicionarEventoTabela*** e ***visualizarEventos,*** em conjunto, realizam uma parte da *funct3*. Enquanto o ***parserEventosOcorrencia*** traduz os eventos lidos pelo ***lerAud*** para o tipo *EventoViewModel*, ***adicionarEventoTabela*** e ***visualizarEventos*** os quais exibem estes eventos em uma tabela.

O método ***gerarScript***, invoca um método de outra Classe importante, a classe *GeradorDeScript* – importada do projeto *Gerador de Cenário para Treinamento de Operadores*. Seu diagrama de classe é ilustrado na figura 33.

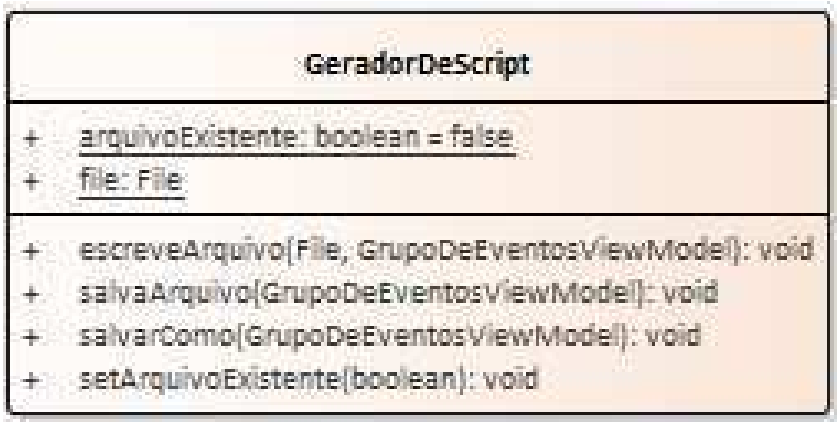


Figura 33: Diagrama de classe do Gerador de Script (Fonte: o próprio autor).

Os métodos responsáveis por implementar a *funct4* são: ***escreveArquivo*** e ***salvarArquivo***. O método ***salvarComo*** não foi utilizado, visto que para os propósitos de validação o método ***salvarArquivo*** foi suficiente.

Com a função ***escreveArquivo***, realiza-se uma escrita em um arquivo de texto seguindo a formatação do OTS. Para isto, utilizamos operações de entrada e saída do Java, através do método ***outputstream***.

```
try {
    //Cabeçário
    outputStream.printf("* Export Event Library   00/00/00     00:00:00");
    outputStream.printf("\n* OTS Model      EXP    Database epriots/db/DBBA");
    outputStream.printf("\n* OTS Date  00/00/0000  TIME  00:00:00");
    outputStream.printf("\n* Maximum Number of Event Groups       20");
    outputStream.printf("\n* Maximum Number of Events Per Group   99");
```

Figura 34: Trecho de código do método ***escreveArquivo*** utilizando a função ***outputStream*** seguindo a formatação do OTS (Fonte: O próprio autor).

3.3.4 Classes FachadaOntologiaCenario e FachadaOntologiaOcorrencia

Essas duas classes são responsáveis por implementar a última funcionalidade do *Gerador Automático de Cenário*. Os diagramas de classe são apresentados nas figuras 35 e 36.

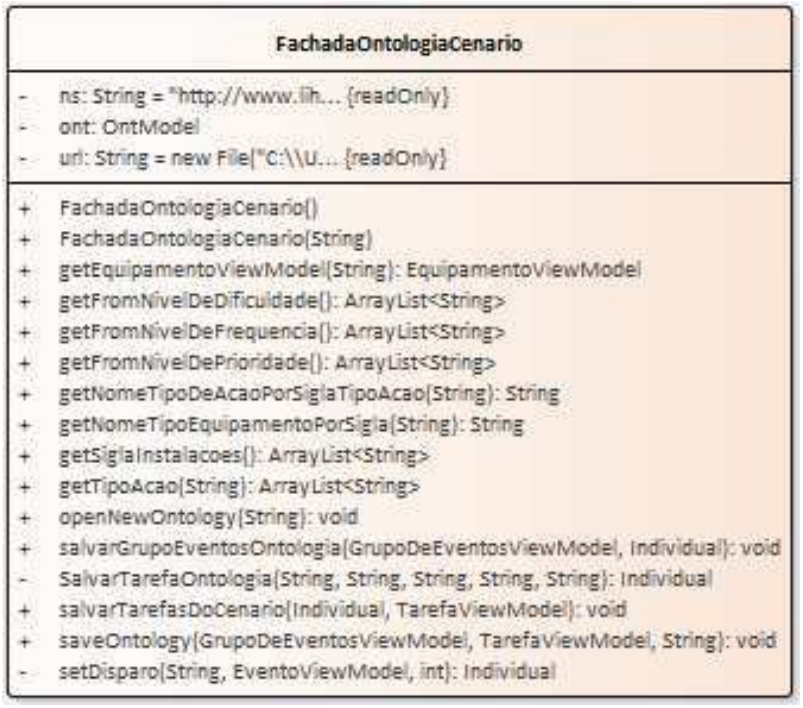


Figura 35: Diagrama de classe da FachadaOntologiaCenario do pacote ontologia (Fonte: O próprio autor).

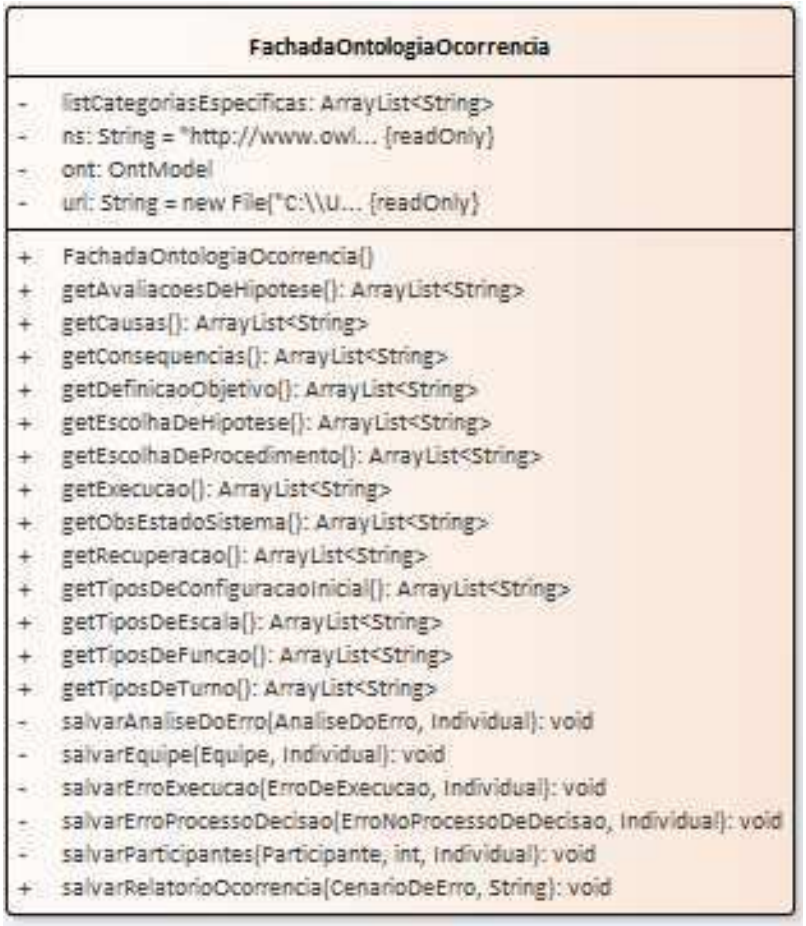


Figura 36: Diagrama de classe da FachadaOntologiaOcorrencia do pacote ontologia (Fonte: O próprio autor).

Todos os métodos do tipo ***get,*** em ambas as classes, são usados para extrair informações das Ontologias visando povoar a interface gráfica do sistema. Os demais métodos do tipo ***salvar*** são responsáveis por escrever os seus objetos específicos nas ontologias. Enquanto os métodos ***saveOntology*** – da *FachadaOntologiaCenario* – e ***salvarRelatorioOcorrencia*** – da *FachadaOntologiaOcorrencia* – são responsáveis por produzir os arquivos OWL referentes às suas ontologias.

Ao finalizar a implementação do *Gerador Automático de Cenários,* iniciamos a discussão sobre o processo de validação do mesmo, uma vez que para alcançarmos os objetivos propostos, precisamos de alguma métrica que defina a aceitabilidade do nosso sistema. Assim, no próximo capítulo apresentaremos o processo de validação da nossa ferramenta.

4 VALIDAÇÃO

Neste capítulo discutiremos o processo de validação e os resultados alcançados com o sistema.

Durante a fundamentação deste trabalho, foram apresentadas duas ontologias de domínio: *Cenário de Treinamento* e *Cenário de Erro*. A fim de validarmos o *Gerador Automático de Cenário,* foram realizados os testes a seguir:

i. Teste 01: A partir de um roteiro de evento simulado – cedido pela CHESF – o qual consiste em um documento com a descrição do roteiro e um arquivo de auditagem, utilizando a ferramenta *Gerador Automático de Cenário,* foi produzido um arquivo de texto puro a ser exportado para o SAGE. O resultado obtido foi comparado com o documento que descreve o roteiro de evento simulado, gerado interativamente pelo tutor, e verificado se os eventos produzidos pela ferramenta de geração automática são equivalentes àqueles presentes no documento roteiro de evento simulado.

ii. Teste 02: A partir do relatório RDFH citado na capítulo 3.3.1, a ferramenta *Gerador Automático de Cenário* foi alimentada com as informações presentes no RDFH. Em seguida, utilizando os arquivos OWL referentes às ontologias: *Cenário de Erro* e *Cenário de Treinamento*, foram comparadas as informações contidas nas ontologias com aquelas contidas no RDFH.

A figura 37 ilustra o processo de validação.

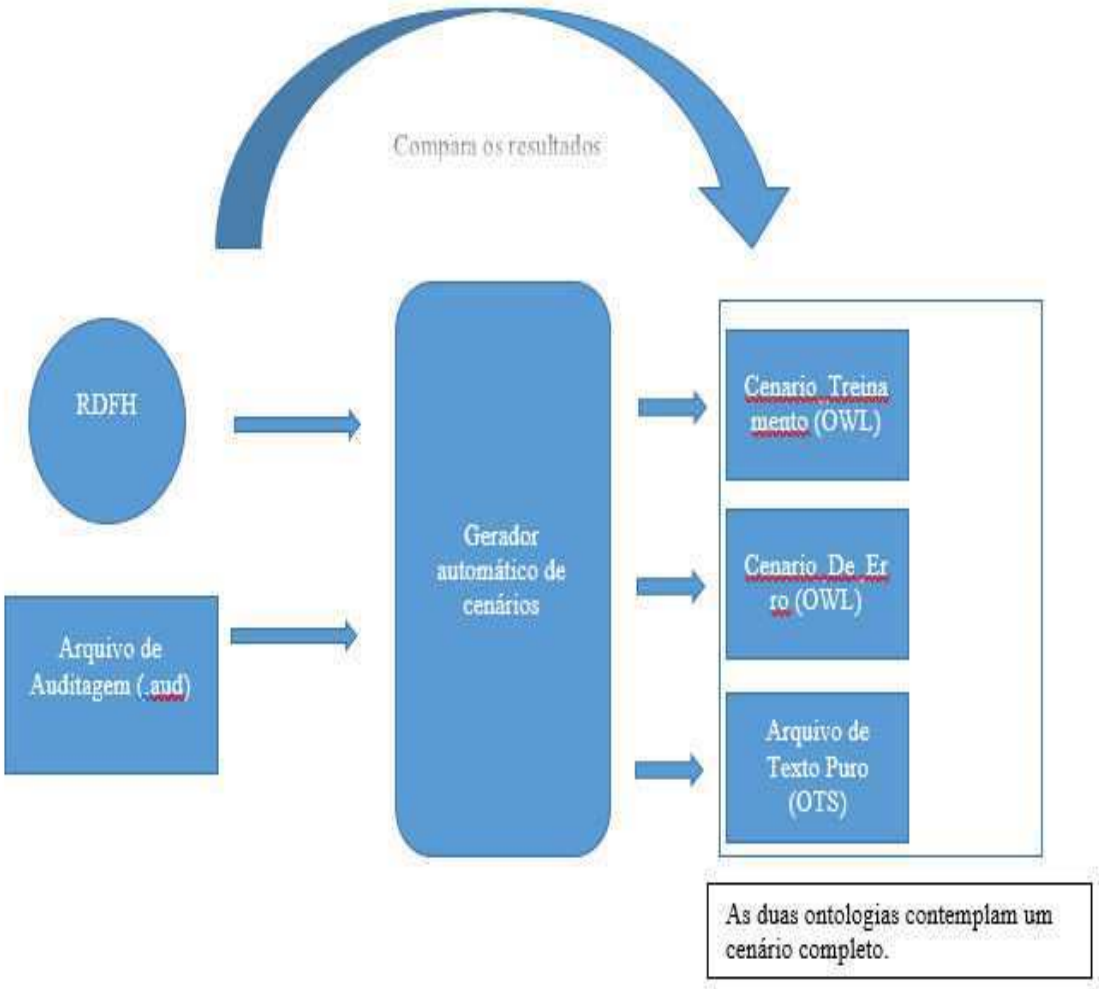


Figura 37: Representação do processo de validação da ferramenta *Gerador automático de cenários* (Fonte: O próprio autor).

4.1 RESULTADOS

Inicialmente, alimentamos a ferramenta com o arquivo *Sessao1.aud* referente ao roteiro de evento simulado de outubro de 2014 cedido pela CHESF. Deste roteiro, pudemos extrair as informações sobre o comportamento especificado para os equipamentos envolvidos na simulação durante o treinamento:

Disjuntores Abertos: 14T7, 12T4, 12T5, 12T6, 12T7, 12F1, 12F2, 12F3 e *12F4*.

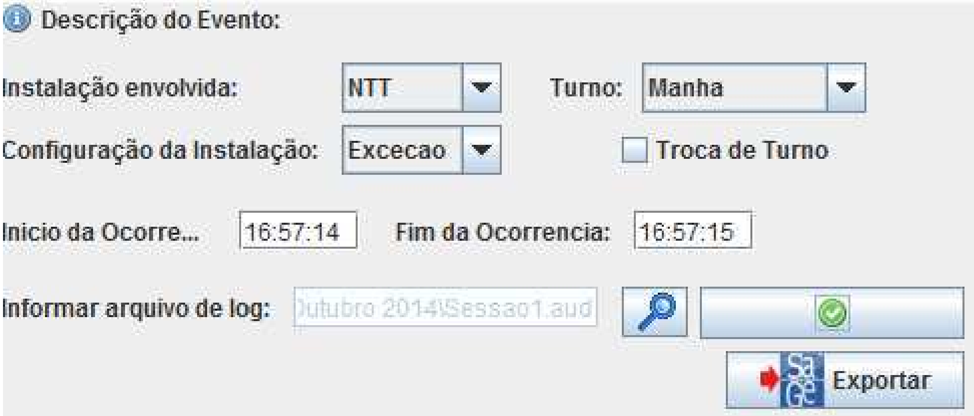
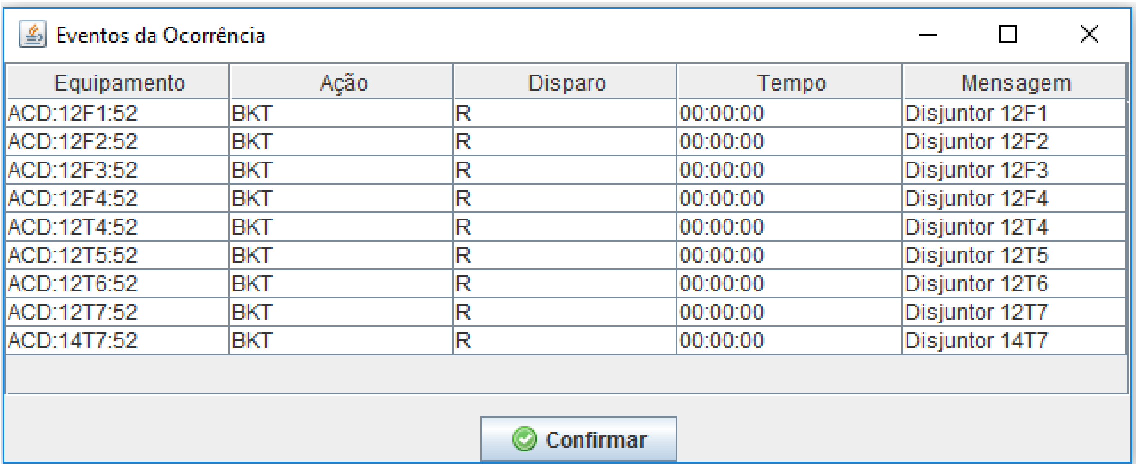


Figura 38: Seleção do arquivo de auditagem (.aud) utilizando a ferramenta *Gerador Automático de cenários* (Fonte: O próprio autor).

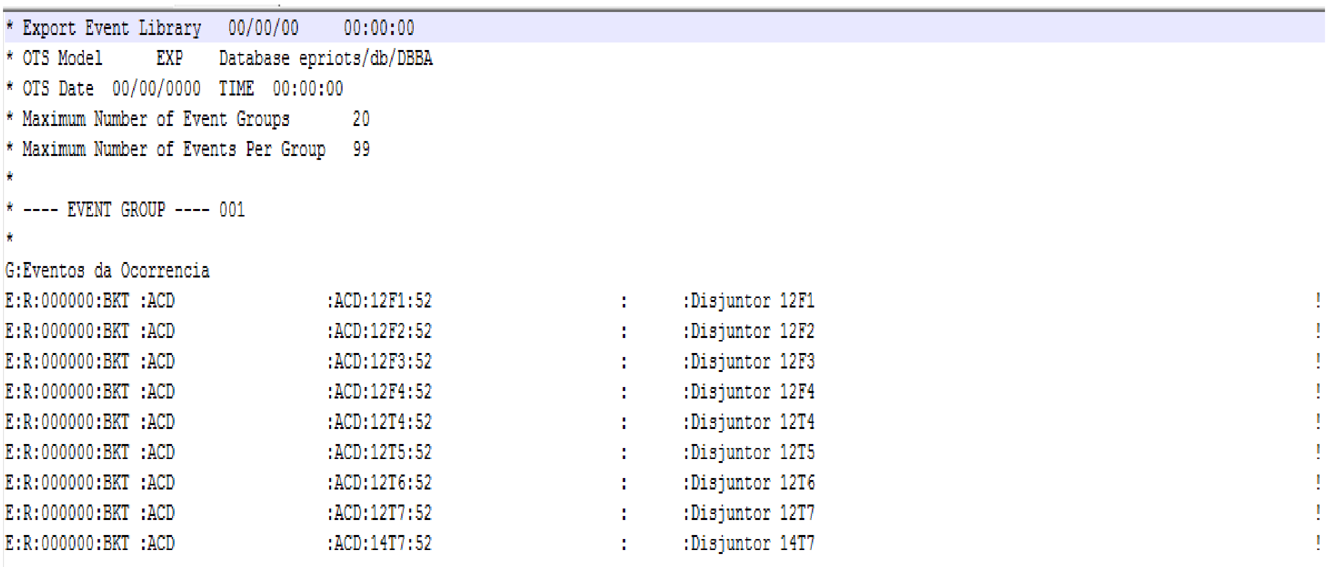


Eventos da Ocorrência

Equipamento	Ação	Disparo	Tempo	Mensagem
ACD:12F1:52	BKT	R	00:00:00	Disjuntor 12F1
ACD:12F2:52	BKT	R	00:00:00	Disjuntor 12F2
ACD:12F3:52	BKT	R	00:00:00	Disjuntor 12F3
ACD:12F4:52	BKT	R	00:00:00	Disjuntor 12F4
ACD:12T4:52	BKT	R	00:00:00	Disjuntor 12T4
ACD:12T5:52	BKT	R	00:00:00	Disjuntor 12T5
ACD:12T6:52	BKT	R	00:00:00	Disjuntor 12T6
ACD:12T7:52	BKT	R	00:00:00	Disjuntor 12T7
ACD:14T7:52	BKT	R	00:00:00	Disjuntor 14T7

Confirmar

Figura 39: Visualização dos eventos da ocorrência (Fonte: O próprio autor).



```
* Export Event Library   00/00/00     00:00:00
* OTS Model      EXP    Database epriots/db/DBBA
* OTS Date   00/00/0000  TIME  00:00:00
* Maximum Number of Event Groups       20
* Maximum Number of Events Per Group   99
*
* ---- EVENT GROUP ---- 001
*
G:Eventos da Ocorrencia
E:R:000000:BKT :ACD            :ACD:12F1:52            :      :Disjuntor 12F1                                      !
E:R:000000:BKT :ACD            :ACD:12F2:52            :      :Disjuntor 12F2                                      !
E:R:000000:BKT :ACD            :ACD:12F3:52            :      :Disjuntor 12F3                                      !
E:R:000000:BKT :ACD            :ACD:12F4:52            :      :Disjuntor 12F4                                      !
E:R:000000:BKT :ACD            :ACD:12T4:52            :      :Disjuntor 12T4                                      !
E:R:000000:BKT :ACD            :ACD:12T5:52            :      :Disjuntor 12T5                                      !
E:R:000000:BKT :ACD            :ACD:12T6:52            :      :Disjuntor 12T6                                      !
E:R:000000:BKT :ACD            :ACD:12T7:52            :      :Disjuntor 12T7                                      !
E:R:000000:BKT :ACD            :ACD:14T7:52            :      :Disjuntor 14T7                                      !
```

Figura 40: Arquivo de texto puro produzido pela ferramenta *Gerador Automático de Cenários* (Fonte: o próprio autor).

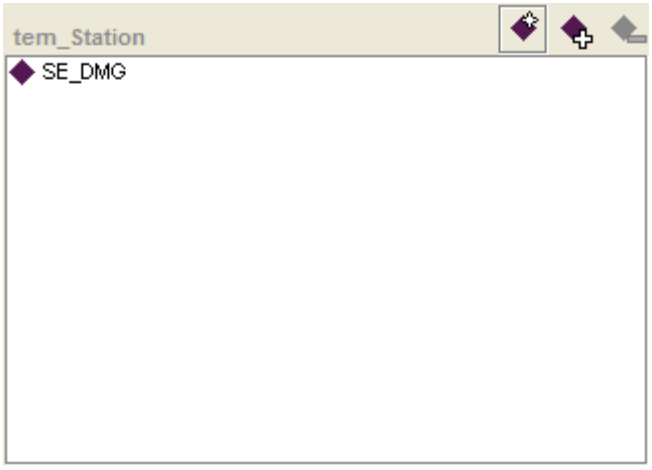


Figura 41: Objeto Instalação da ontologia *Cenário de Treinamento* - visualização no *Protégé* (Fonte: O próprio autor).

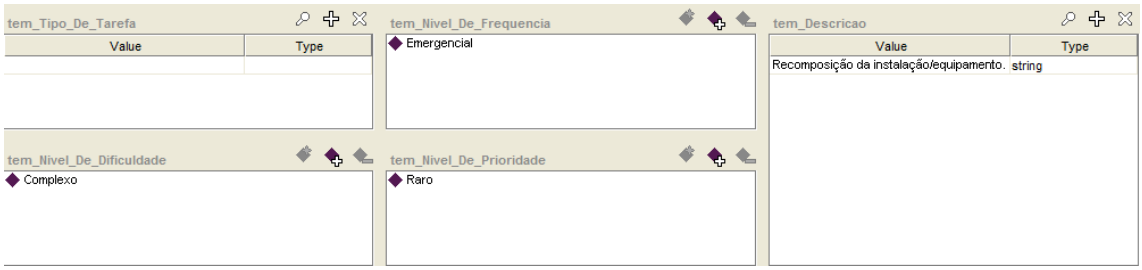


Figura 42: Tarefa salva na ontologia *Cenário de Treinamento - visualização no* Protégé *(Fonte: O próprio autor).*

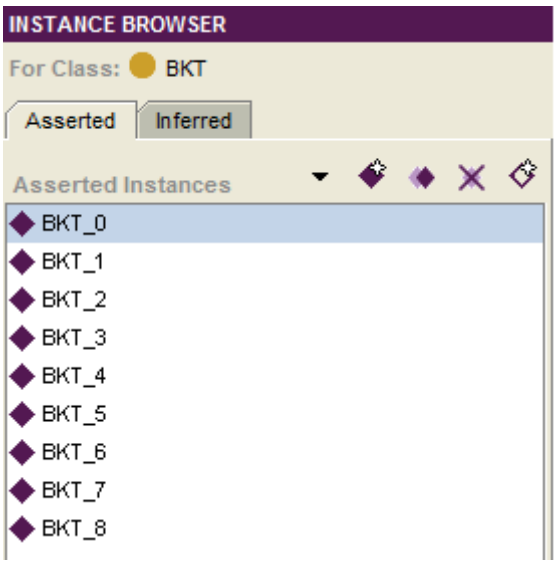


Figura 43: Eventos programados salvos na ontologia *Cenário de Treinamento* - visualização no *Protégé* (Fonte: O próprio autor).

Observou-se que os eventos de ocorrência são equivalentes ao proposto no roteiro de evento programado, assim como arquivo OWL referente à ontologia *Cenário de treinamento*, cujos objetos estão representados nas figuras 41, 42 e 43, foi produzido, concluindo o teste 1.

Dando continuidade à validação, a ferramenta construída foi utilizada para salvar as informações referentes ao relatório RDFH. Na figura 44, podemos visualizar o conteúdo salvo na ontologia *Cenário de Erro* através da ferramenta *Protégé*. A figura 45 apresenta os detalhes da subclasse *Equipe* da mesma ontologia. Como existem dois participantes, apresentamos os detalhes dos participantes nas figuras 46 e 47. As figuras subsequentes, 48, 49 e 50 expõem a tipologia do erro, também salva na ontologia citada.

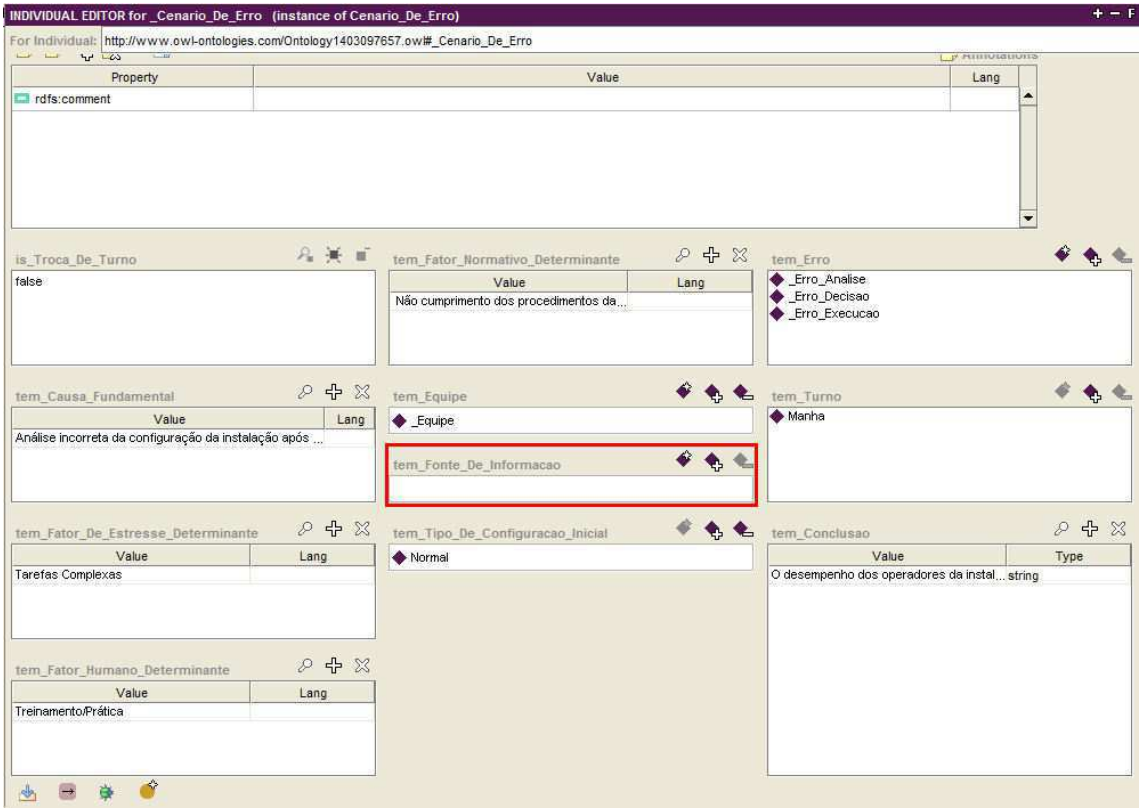


Figura 44: Objetos salvos na ontologia *Cenário de Erro* - visualização através do *Protégé* (Fonte: O próprio autor).

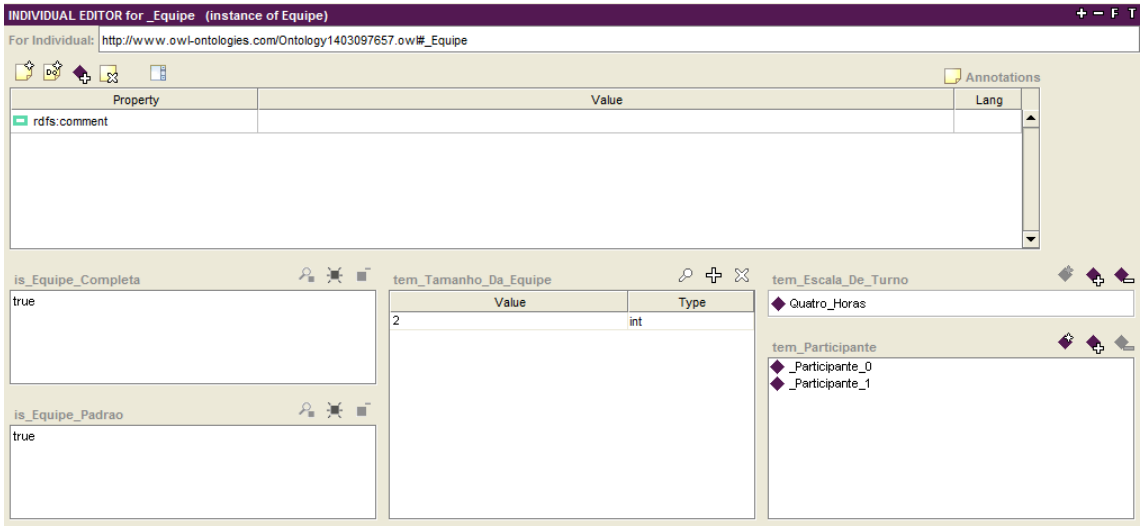


Figura 45: Detalhes da equipe (ontologia *Cenário de Erro*) - visualização através do Protégé (Fonte: O próprio autor).

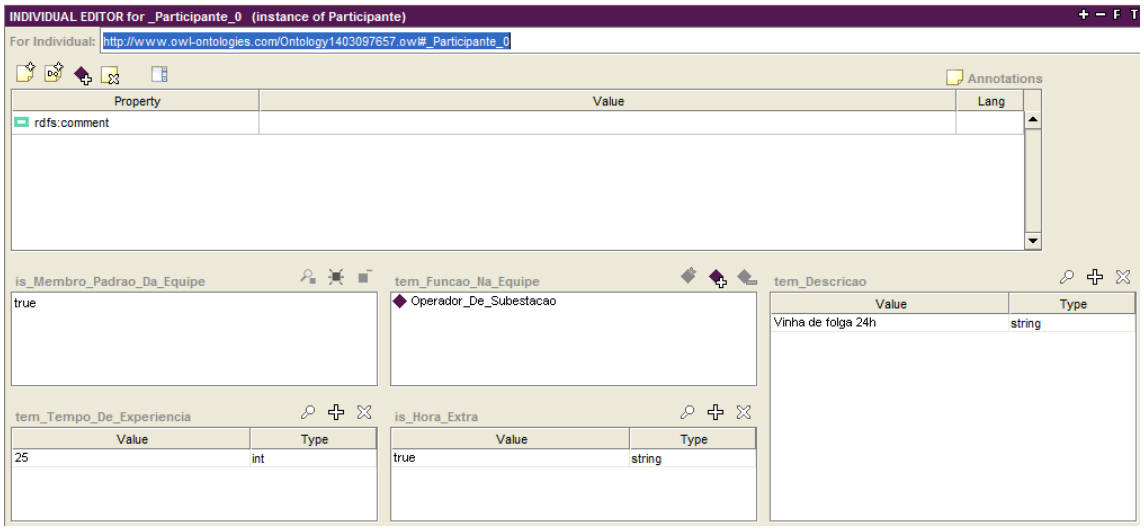


Figura 46: Detalhes do primeiro participante (ontologia *Cenário de Erro*) - visualização através do *Protégé* (Fonte: O próprio autor).

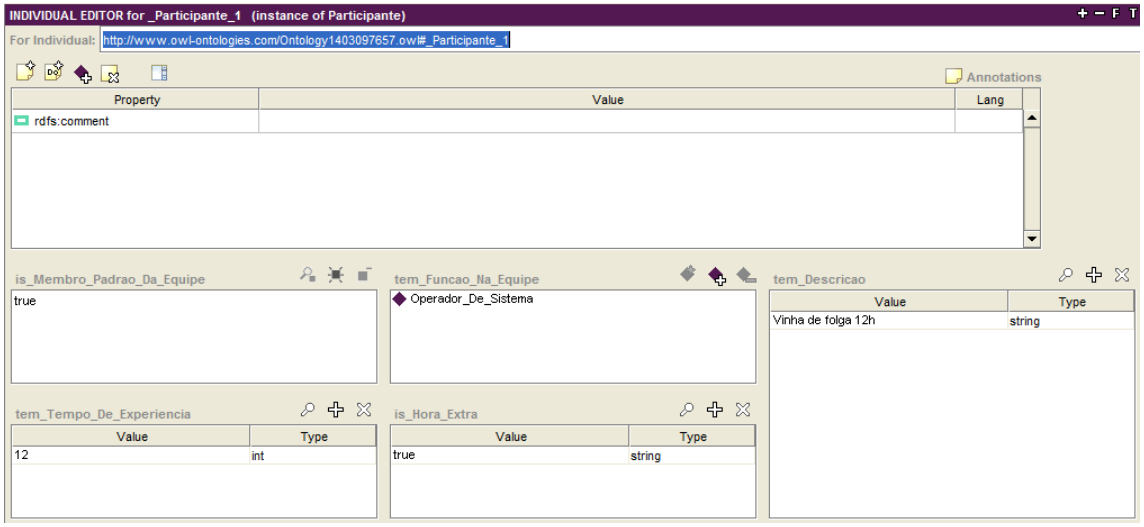


Figura 47: Detalhes do segundo participante (ontologia *Cenário de Erro*) - visualização através do *Protégé* (Fonte: O próprio autor).

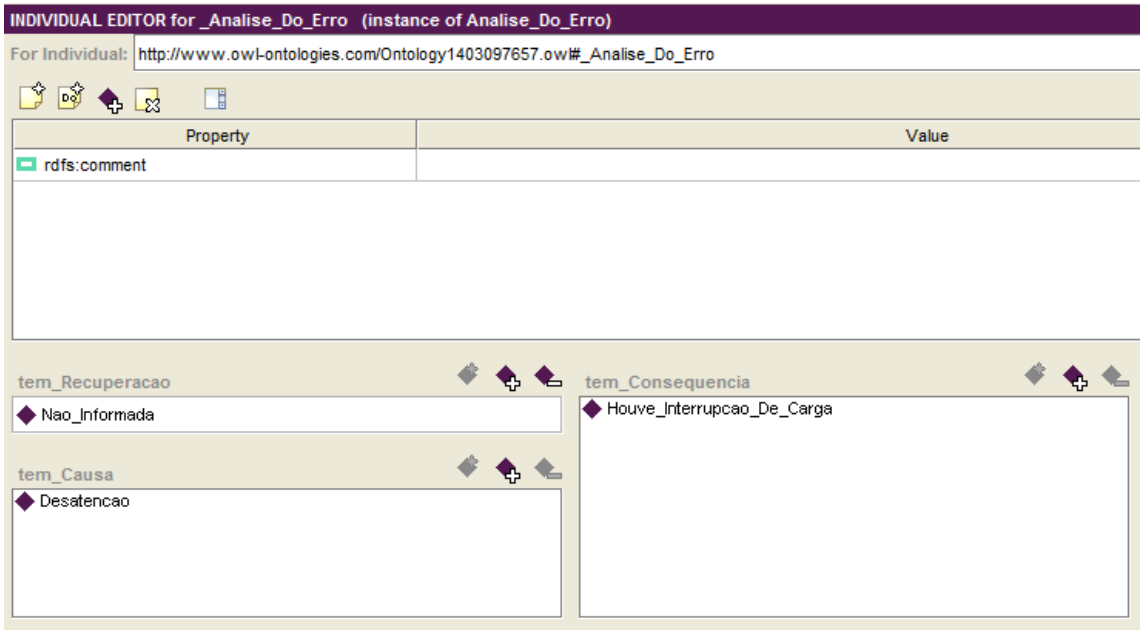


Figura 48: Visualização da análise do erro (ontologia Cenário de Erro) (Fonte: O próprio autor).

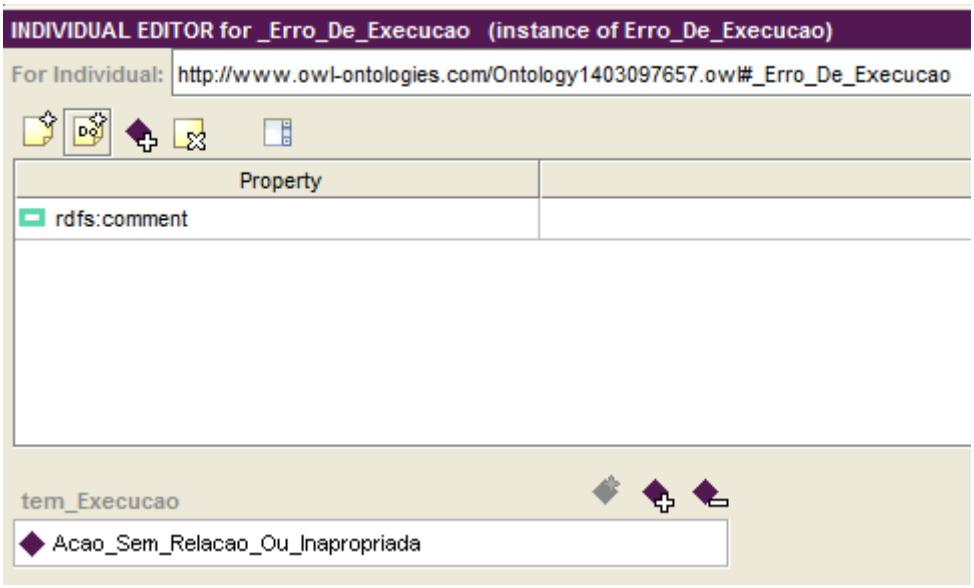


Figura 49: Visualização do Erro de Execução (ontologia Cenário de Erro) (Fonte: O próprio autor).

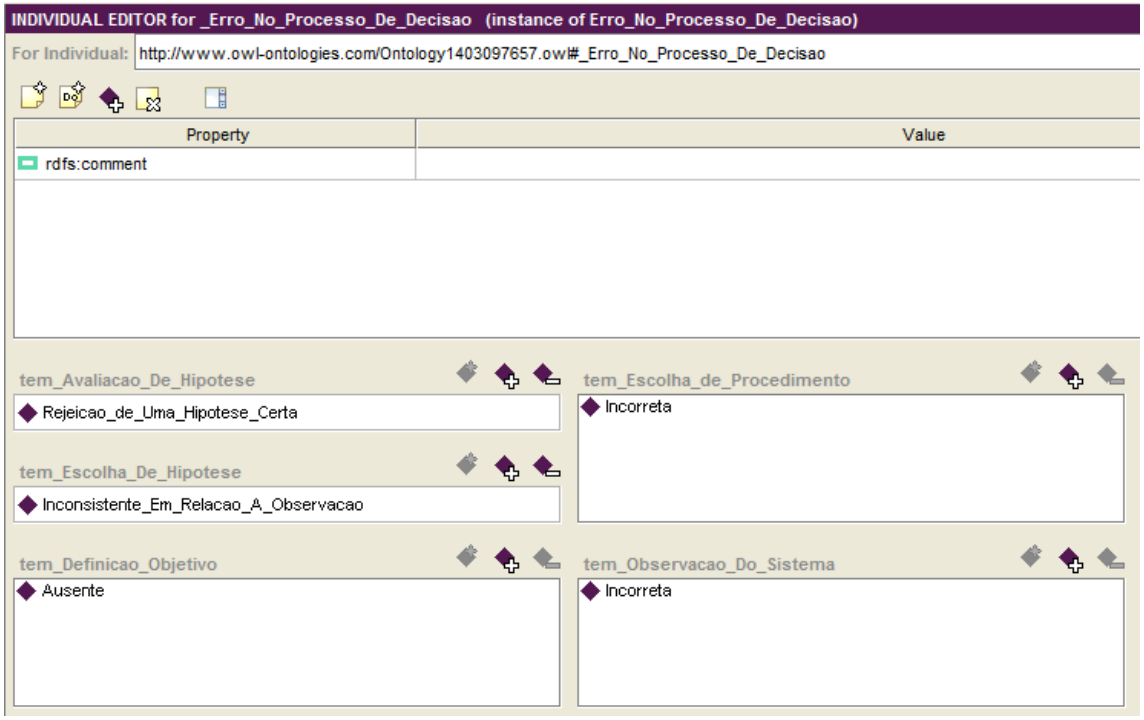


Figura 50: Visualização do Erro no Processo de Decisão (ontologia Cenário de Erro) (Fonte: O próprio autor).

Após a verificação da existência dos 3 arquivos, os dois OWL referentes às ontologias *Cenário de Treinamento* e *Cenário de Erro*, concluímos o teste 2. A discussão sobre estes resultados será abordada no capítulo 5, onde são apresentadas as conclusões deste relatório.

5 CONCLUSÕES

Neste capítulo são discutidos os resultados obtidos a partir deste projeto.

Os resultados obtidos no capítulo 4 ilustram que os objetivos propostos neste trabalho foram alcançados. Com a ferramenta *Gerador Automático de Cenários* e os documentos relatório RDFH e o *log* de incidentes fomos capazes de produzir duas ontologias que descrevem um cenário de treinamento, assim como um arquivo de texto puro o qual pôde ser interpretado pelo OTS.

Durante o desenvolvimento, encontramos algumas dificuldades de acesso à informação – a saber, quais eventos caracterizam fielmente uma ocorrência. Porém, contornamos esta situação através de um algoritmo simples, capaz de identificar diretamente o comportamento de disjuntores ao longo de uma ocorrência – os disjuntores são os principais equipamentos que sofrem mudanças de estado durante uma ocorrência.

Desta forma, concluímos que esta ferramenta é capaz de automatizar a criação de cenários para treinamento de operadores, com ênfase em incidentes ocorridos na CHESF, minimizando o esforço na criação manual destes cenários.

Com as duas ontologias produzidas, é possível alimentar uma base de dados que contemple cenários de treinamentos construídos a partir de ocorrências reais, descrevendo e classificando os eventos de acordo com a análise do erro humano. Isto torna possível a organização de treinamentos, filtrando características importantes, tais como: o tipo de frequência da ocorrência, os tipos de erro relacionados às tomadas de decisão, avaliação de hipóteses, procedimento de tomada de decisão, etc.

5.1 CONSIDERAÇÕES SOBRE TRABALHOS FUTUROS

Como discutido no início deste capítulo, os resultados alcançados satisfizeram os objetivos propostos, porém nem todo o potencial da ferramenta foi contemplado no escopo deste trabalho.

Os testes relacionados à usabilidade da ferramenta não foram contemplados aqui. Desta forma, futuramente poderão ser examinadas interfaces gráficas mais ergonômicas que facilitem o uso do nosso sistema.

Ainda com relação à classificação do erro humano, constatamos que o uso de ontologias neste domínio pode aperfeiçoar o treinamento dos operadores de sistemas elétricos. Logo, uma vez construída uma base de dados com diversos cenários gerados pela ferramenta, seria interessante a proposição de uma ferramenta que estimasse as causas mais frequentes dos erros humanos, direcionando o uso destes cenários de forma mais específica, que atendessem as necessidades individuais dos operadores em treinamento.

REFERÊNCIAS

[1] P. Hirsch, “User Guide for PowerSimulator with EPRI OTS,” EPRI, Palo Alto, California, February 2004.

[2] S. Lee, “Instructor Guidelines for Use of an Operator Training Simulator (OTS),” October 2001.

[3] Manual do usuário para os aplicativos de suporte desenvolvidos ou personalizados, Rio de Janeiro: CEPEL, Dezembro de 2007.

[4] TORRES FILHO, F.; VIEIRA, M. d. F. Q. Processo para o desenvolvimento de cenários de treinamento para ambientes virtuais 3D. Anais do XI Simpósio Brasileiro de Automação Inteligente, 2013.

[5] OLIVEIRA, J. J. R., PEREIRA, L. A. C., LIMA, L. C., SOLLERO, R. B., LEITE, C. R. R., MUNIZ, R. B., COSTA, C. A. B., CAVALCANTE, M. S., CARMO, U. A. C., ARAUJO, A. S., MEIRELLES,L.C., Treinamento e Certificação de Operadores no Sistema SAGE Empregando o, Simulador EPRI/OTS, Grupo de Estudos de Operação de Sistemas Elétricos, XVIII SNPTEE, Curitiba-Paraná, outubro 2005.

[6] TORRES FILHO, F, “ABORDAGEM ONTOLÓGICA PARA MODELAGEM DO TREINAMENTO DE OPERADORES DE SISTEMAS ELÉTRICOS", Campina Grande, 2015.

[7] NASCIMENTO NETO, J. A., “Processo para concepção de estratégias para prevenção do erro na operação de sistemas elétricos,” Campina Grande, 2010.

[8] OMENA, R. A. L. V., “Ferramenta para Edição de Cenários Gerados,” Campina Grande, 2011.

[9] JENA API. Jena2 ontology API. Available at: http://jena.sourceforge.net/ontology/index.html. Acesso em 21 de Julho de 2016.

[10] PEREIRA, L. A. C., LIMA, L. C., SILVA, A. J. R., MACHADO, P. A., AMORIM, M. F. P., FILHO, A. L. O., AZEVEDO, G. P., LAMBERT, N., ZANUR, P. D., TAVARES, V. V., HUANG, J. L. C., COSTA, M. R., VIDAL, C. G., IENCARELLI, C. E. “SAGE- Um Sistema Aberto para a Evolução”.

[11] Julita Bermejo-Alonso, Ricardo Sanz, Manuel Rodríguez, and Carlos Hernández. An ontological framework for autonomous systems modelling. *International Journal on Advances in Intelligent Systems,* 3 (3 and 4): 211-215, 2011.

[12] RANGEL, B. G., “Descrição e armazenamento de cenários de treinamento de operadores a partir de uma ontologia de domínio”, Campina Grande, 2015.

[13] GUERRERO, S. V. C., “Modelo conceitual de cenários de acidentes causados pelo erro humano em sistemas industriais críticos com o foco na concepção de interfaces ergonômicas”, Campina Grande, maio de 2015.

[14] AMALBERTI, R. La conduite de systèmes à risques. Collection Le Travail Humain, Presses Universitaire de France.