UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE CIÊNCIAS E TECNOLOGIA
CURSO DE PÓS-GRADUAÇÃO EM INFORMÁTICA

TRATAMENTO DE EVENTOS EM REDES ELÉTRICAS: UMA FERRAMENTA

ALEXANDRE NÓBREGA DUARTE

Campina Grande

Janeiro - 2003

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE CIÊNCIAS E TECNOLOGIA
CURSO DE PÓS-GRADUAÇÃO EM INFORMÁTICA

TRATAMENTO DE EVENTOS EM REDES ELÉTRICAS: UMA FERRAMENTA

ALEXANDRE NÓBREGA DUARTE

Dissertação submetida à Coordenação de Pós-Graduação em Informática do Centro de Ciências e Tecnologia da Universidade Federal de Campina Grande como requisito parcial para a obtenção do grau de Mestre em Ciências (MSc).

Área de Concentração: Ciências da Computação

Linha de Pesquisa: Redes de Computadores e Sistemas Distribuídos

Orientador: Jacques Philippe Sauvé

Campina Grande

Janeiro – 2003

"A imaginação é mais importante que o conhecimento”.

Albert Einstein

Agradecimentos

Ao meu orientador, Jacques, pela confiança de que poderíamos realizar um bom trabalho;

A toda a minha família, por apoiar e incentivar este trabalho, especialmente a minha mãe, Emeide, pela disponibilidade para discussões sobre a pesquisa em si e pelo auxílio na normalização do trabalho;

À CHESF, por incentivar e patrocinar este projeto, especialmente a Sérgio, Socorro, Marcelo e Madison, pela colaboração e participação ativa no trabalho;

A toda a equipe do projeto *Smart Alarms* (Jacques, Walfredo, Jorge, Marcus e Eloi) por serem um ótimo time e por contribuírem com tudo o que foi preciso para que os objetivos deste trabalho fossem atingidos;

Aos meus amigos e amigas, pelo apoio e imensa torcida;

Aos professores e colegas do DSC, pelos valorosos conhecimentos compartilhados durante nossa convivência.

DUARTE, Alexandre Nóbrega. **Tratamento de Eventos em Redes Elétricas**: uma Ferramenta. Campina Grande: 2003. 140 f. Dissertação (Mestrado em Informática) – Universidade Federal de Campina Grande – 2003.

Resumo

Apresenta uma nova ferramenta para o diagnóstico automático de falhas em redes elétricas. A ferramenta utiliza uma técnica híbrida de correlação de eventos criada especialmente para ser utilizada em redes com constantes modificações de topologia. A técnica híbrida combina o raciocínio baseado em regras com o raciocínio baseado em modelos para eliminar as principais limitações do raciocínio baseado em regras. Com a ferramenta de diagnóstico foi possível validar o conhecimento dos especialistas em sistemas de transmissão de energia elétrica necessário para o diagnóstico de falhas em linhas de transmissão e construir uma base de regras para tal. A ferramenta foi testada no diagnóstico de falhas em linhas de transmissão de um dos cinco centros regionais da Companhia Hidro Elétrica do São Francisco (CHESF) e apresentou resultados satisfatórios de desempenho e precisão.

Palavras-chave: Correlação de Eventos; Processamento de Eventos; Diagnóstico de Causa Raiz, Raciocínio Baseado em Regras, Raciocínio Baseado em Modelos.

DUARTE, Alexandre Nóbrega. **Tratamento de Eventos em Redes Elétricas**: uma Ferramenta. Campina Grande: 2003. 140 f. Dissertação (Mestrado em Informática) – Universidade Federal de Campina Grande – 2003.

Abstract

It presents a new tool for the automatic diagnosis of faults in electric networks. The tool uses a hybrid event correlation technique especially created to be used in networks with constant topological modifications. The hybrid technique combines rule-based reasoning with model-based reasoning to eliminate the main limitations of rule-based reasoning. With the tool it was possible to validate the knowledge acquired from electric energy transmission systems specialists needed for the diagnosis of faults in transmission lines and to construct rules. The tool was tested in the diagnosis of faults in transmission lines of one of the five regional centers of the Companhia Hidro Elétrica do São Francisco (CHESF) and presented satisfactory results in terms of performance and precision.

Keywords: Event Correlation; Event Processing; Root Cause Analysis; Rule-Based Reasoning; Model-Based Reasoning.

Lista de Siglas

API – Application Programming Interface

CEPEL – Centro de Pesquisas de Energia Elétrica

CHESF – Companhia Hidro Elétrica do São Francisco

COS – Centro de Operação do Sistema

COSERN – Companhia Energética do Rio Grande do Norte

CROL – Centro Regional de Operação Leste

CRON – Centro Regional de Operação Norte

CROS – Centro Regional de Operação Sul

CRS – Chronicle Recognition System

DSC – Departamento de Sistemas e Computação

ECS – Event Correlation Services

EMS – Energy Management System

GMCD – Gerenciador de Memória Compartilhada Distribuída

HP – Hewlett-Packard

IMPACT – Intelligent Management Plataforms for Alarm Correlation Tasks

ISO – International Standard Organization

MCD – Memória Compartilhada Distribuída

OSI – Open Systems Interconnection

SAGE – Sistema Aberto de Gerência de Energia

SCADA – Supervisory Control and Data Acquisition

SGBD – Sistema Gerenciador de Banco de Dados

UFCG – Universidade Federal de Campina Grande

UTRs – Unidades Terminais Remotas

XDR – External Data Representation Standard

Lista de Tabelas

Lista de Figuras

Sumário

1 Introdução

Os centros de supervisão e controle das redes modernas de transmissão e distribuição de energia elétrica têm a tarefa complexa de gerenciar redes grandes e geograficamente abrangentes. Felizmente, tais centros adquirem em tempo real uma grande quantidade de dados sobre a rede elétrica, facilitando assim o diagnóstico e a localização de condições de anormalidade no sistema. Nos centros de controle modernos existem sistemas de software responsáveis pelo acompanhamento da carga no sistema, pela análise de contingências, pela análise de curtos-circuitos, dentre outras funções. Entretanto, o conhecimento especialista de operadores humanos ainda é indispensável para supervisionar o sistema e tomar decisões críticas, principalmente em situações de emergência. Em tais situações, são normalmente geradas grandes quantidades de eventos (indicando situações potencialmente anormais), um efeito cascata muitas vezes originado por uma única falha. Existem registros de situações críticas nos centros de controle em que os operadores receberam **mais de 1500 eventos em um único segundo**.[1]

O grande volume de eventos em situações críticas é um problema para a operação do sistema, pois aumenta o tempo de diagnóstico e de reação dos operadores, que têm que "digerir" todo o surto de eventos para identificar os reais problemas no sistema. Dessa forma, nos momentos críticos, quando, devido à gravidade da situação e à quantidade de clientes afetados, o diagnóstico deve ser realizado o mais rápido possível para que ações corretivas possam ser executadas, o tempo levado pelos operadores para descobrir o que está realmente acontecendo na rede elétrica é muito maior. Além disso, humanos estão sujeitos a cometer erros em situações de estresse, e um diagnóstico incorreto pode agravar ainda mais a situação uma vez que uma ação corretiva equivocada pode danificar um equipamento ou propagar os efeitos de uma falha localizada para outras partes do sistema.

---

[1] Esse dado foi obtido no banco de dados de monitoração da rede elétrica da CHESF e é referente ao dia 7 de agosto de 2001, às 06:08:14 da manhã.

Nos sistemas de potência, tempo é dinheiro pois, quanto mais tempo o diagnóstico e as ações corretivas levarem para ser realizados, mais tempo o sistema estará fora de operação, implicando uma interrupção do fornecimento de energia e, conseqüentemente, a interrupção do faturamento da empresa. Além do fator financeiro, as interrupções no fornecimento de energia diminuem a satisfação dos clientes com a empresa.

Uma forma de diminuir a quantidade de informação que os operadores precisam interpretar e, conseqüentemente, diminuir o seu tempo de resposta e probabilidade de erros, é associar os eventos as suas causas raiz[2]; desse modo, os operadores teriam que interpretar apenas as possíveis causas dos problemas na rede elétrica, e não, as suas conseqüências (eventos).

O raciocínio envolvido no diagnóstico de falhas (causa raiz) em sistemas de potência é eminentemente simbólico, o que viabiliza a sua automatização através de software. Esse fato levantou a possibilidade de se utilizarem aplicações baseadas em conhecimento para o processamento automático de eventos, permitindo associar uma série de eventos correlatos com uma única causa raiz (DABBAGHCHI, 1997).

Várias empresas e grupos de pesquisa têm pesquisado e desenvolvido técnicas e aplicações para o diagnóstico de falhas em redes de computadores, redes elétricas e redes de telecomunicações e diversos outros tipos (AZEVEDO, 2001) (BIBAS, 1996) (BIELER, 1994) (BRUNNER, 1993) (CORDIER, 2001) (CRONK, 1998) (GARDNER, 1996) (HP, 1995) (KLIGER, 1995) (SEAGATE, 1996). Aparentemente eles não têm considerado a possibilidade de aplicar uma técnica desenvolvida para um tipo de rede nos demais tipos. Mais precisamente, a maioria das técnicas de diagnóstico e correlação de eventos utilizadas em redes de computadores e redes de telecomunicações não têm sido aplicadas no diagnóstico de falhas em redes elétricas, apesar de não existir nenhum impedimento teórico para isso. Talvez por esse fato exista um razoável número de aplicações comerciais para diagnóstico de falhas em redes de computadores e redes de telecomunicações (SEAGATE, 1996) (HP, 1995) (YEMINI, 1996) (LEWIS, 1999) e um número bastante reduzido de aplicações para redes elétricas (VALE, 1997) (LABORIE, 1997).

A Companhia Hidro Elétrica do São Francisco (CHESF) e o Departamento de Sistemas e Computação da Universidade Federal de Campina Grande (DSC/UFCG) firmaram

[2] A causa raiz é a origem dos problemas na rede. Sabe-se que um problema pode se propagar e gerar outros problemas. Apenas o problema inicial é considerado como causa raiz.

uma parceria para desenvolver o projeto “Smart Alarms: Tratamento Inteligente de Alarmes On-line de Subestações e Centros de Controle”. O projeto *Smart Alarms* conta com a participação de 4 professores do DSC/UFCG, 2 engenheiros da CHESF e 2 alunos do Mestrado em Informática da UFCG e visa ao desenvolvimento e à integração com o sistema de gerenciamento de energia da CHESF de uma ferramenta automática de diagnóstico de falhas em rede elétricas.

Este trabalho de Mestrado é uma parte do projeto *Smart Alarms* e corresponde à pesquisa e ao desenvolvimento realizados no primeiro ano do projeto.

1.1 Objetivos da Dissertação

Esta dissertação tem como objetivo principal analisar a aplicação de algumas técnicas de correlação de eventos ao diagnóstico de falhas em redes elétricas para auxiliar no processo de restabelecimento do fornecimento de energia após uma falha. Dentro deste objetivo, deseja-se propor uma arquitetura baseada em componentes, que possibilite um alto nível de reusabilidade de software, para o diagnóstico de falhas em redes elétricas e implementar uma aplicação que permita a validação da arquitetura proposta e a investigação da aplicabilidade de técnicas de correlação de eventos ao diagnóstico de falhas em redes elétricas.

Além da análise de aplicabilidade de técnicas, um outro objetivo do trabalho é adquirir e validar o conhecimento dos especialistas em operação de sistemas elétricos utilizado para o diagnóstico de falhas na rede elétrica. Não é possível validar esse conhecimento sem um sistema de diagnóstico real que permita comparar seus diagnósticos com os diagnósticos dos especialistas. A validação do conhecimento obtido é mais uma das funções da ferramenta de diagnóstico desenvolvida.

1.2 Estrutura da Dissertação

O assunto tratado nesta dissertação foi dividido em seis capítulos, incluindo esta introdução.

No Capítulo 2, são fornecidos os conceitos da engenharia elétrica e do gerenciamento de sistemas de potência necessários para que um cientista da computação possa desenvolver soluções de software nas áreas de geração, transmissão e distribuição de energia elétrica. Além disso, são apresentados dados sobre a CHESF, o ambiente de estudo do trabalho e o SAGE, o sistema utilizado no gerenciamento da rede elétrica da CHESF.

Os conceitos da inteligência artificial utilizados no diagnóstico automático de falhas são fornecidos no Capítulo 3, através de um resumo sobre técnicas de correlação de eventos e do estabelecimento do vocabulário específico da área.

O levantamento dos requisitos e o projeto de uma aplicação de diagnóstico de falhas em sistemas elétricos são apresentados no Capítulo 4, juntamente com a especificação de uma nova técnica de correlação de eventos que combina duas das técnicas discutidas no Capítulo 3.

No Capítulo 5, a implementação de uma aplicação para o diagnóstico de falhas em sistemas elétricos é detalhada e são discutidos aspectos da metodologia de testes utilizada para validar os resultados da aplicação.

Finalizando, no Capítulo 6, estão as conclusões e propostas para trabalhos futuros, incluindo possíveis extensões e alterações da aplicação desenvolvida.

Além dos capítulos acima apresentados o trabalho também consta de dois apêndices. O Apêndice A contém alguns trechos do arquivo de configuração da aplicação de diagnóstico de falhas em rede elétricas; no Apêndice B, são expostas as regras de diagnóstico criadas para o diagnóstico de falhas nas linhas de transmissão do Centro Regional de Operação Leste (CROL) da CHESF, utilizado como piloto para os testes da aplicação.

2 Geração, Transmissão e Distribuição de Energia Elétrica

Este capítulo tem a função de fornecer informações sobre o ambiente de estudo tratado nesta dissertação e prover os conhecimentos sobre engenharia elétrica e gerenciamento de sistemas de potência necessários para um bom entendimento do trabalho como um todo. Não foram considerados detalhes técnicos, tampouco fundamentações matemáticas ou físicas dos assuntos abordados uma vez que tal conhecimento não foi necessário para o desenvolvimento do trabalho proposto. Ao invés disso, o conhecimento é apresentado de forma intuitiva e, muitas vezes, auto-explicativa.

A seção 2.1 aborda os conceitos da engenharia elétrica referentes ao sistemas de geração, transmissão e distribuição de energia elétrica. A seção 2.2 apresenta os sistemas SCADA (*Supervisory Control and Data Acquisition*) responsáveis pela aquisição de dados e controle dos sistemas elétricos, enquanto a seção 2.3 trata do ambiente de estudo desta dissertação de Mestrado a Companhia Hidro Elétrica do São Francisco (CHESF) e a seção 2.4 expõe detalhes do SAGE (Sistema Aberto de Gerência de Energia), o sistema SCADA utilizado na gerência da rede elétrica da CHESF. Finalizando, na seção 2.5, são apresentadas algumas considerações adicionais sobre o assunto tratado no capítulo.

2.1 Sistemas de Geração, Transmissão e Distribuição de Energia Elétrica

Na Figura 2.1, é possível observar um esboço de um sistema de geração, transmissão e distribuição de energia elétrica, a qual é produzida na usina de geração e transmitida em linhas de alta tensão e de extra-alta tensão, para minimizar as perdas de energia em transmissões de longa distância, até subestações de transformação que reduzem a tensão das linhas para transmissões de curta e média distâncias. Depois da redução de tensão, a energia chega a consumidores industriais de grande porte, que possuem subestações próprias e as

subestações de distribuição. Nas subestações de distribuição, a energia tem sua tensão novamente reduzida e é enviada para os consumidores industriais de pequeno porte, que não possuem subestações próprias de transformação e para o sistema de distribuição que, por sua vez, encarrega-se de levar a energia até os consumidores residenciais e comerciais.

Nesta seção, são discutidas as três principais funções dos sistemas de potência: geração, transmissão e distribuição de energia elétrica, e são apresentados os principais equipamentos utilizados para o seu funcionamento.

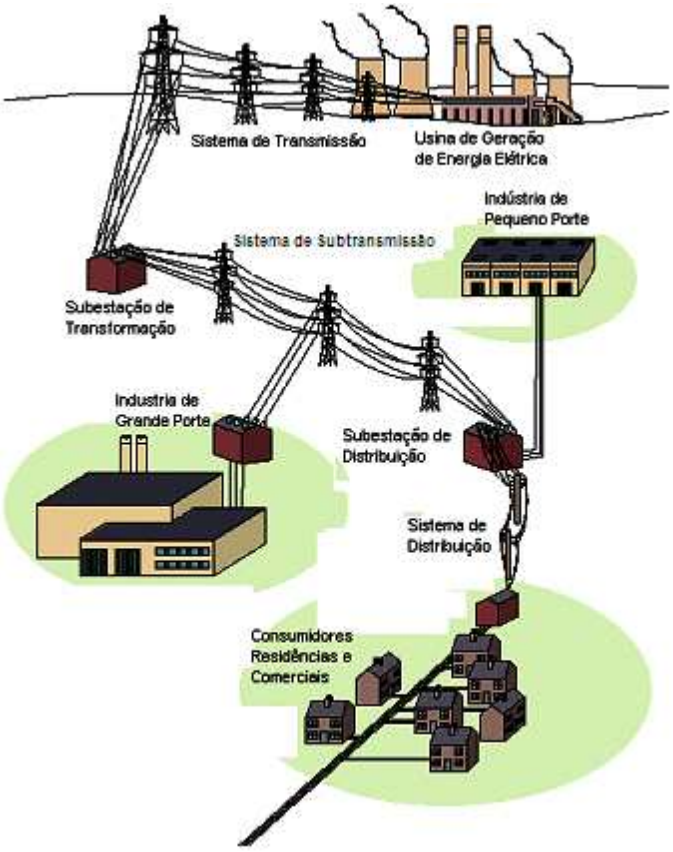

Figura 2.1: Sistema de Geração, Transmissão e Distribuição de Energia Elétrica

2.1.1 Geração

A geração de energia elétrica consiste na transformação de algum tipo de energia em energia mecânica que é utilizada para fazer funcionar os geradores elétricos. A seguir é apresentado o principal método de geração de energia elétrica utilizado no Brasil: geração hidrelétrica.

A geração de energia hidrelétrica envolve o armazenamento de um fluido (Figura 2.2), normalmente água de rios, a conversão da energia hidráulica do fluido em energia mecânica, em uma turbina hidráulica, e a conversão da energia mecânica em energia elétrica por um

gerador elétrico. Apesar do alto custo inicial para a construção de usinas hidrelétricas, os custos baixos de manutenção, o alto tempo de serviço e a alta confiabilidade dos equipamentos as tornam uma fonte de energia flexível e com uma relação custo/benefício muito boa.

Figura 2.2: Reservatório da Usina Hidrelétrica Luiz Gonzaga da CHESF

Usinas hidrelétricas são localizadas em áreas nas quais é possível realizar um uso econômico das fontes de energia hidráulica. A energia hidráulica está presente em qualquer lugar onde há um fluxo de um fluído e um desnível. O desnível representa a energia potencial e é a distância vertical percorrida pelo fluido no processo de conversão de energia. A maioria das usinas hidrelétricas utiliza a energia potencial dos rios, porém outros líquidos, como a água do mar e resíduos de esgotos tratados, também têm sido utilizados. A implantação de usinas hidrelétricas requer um estudo minucioso de fatores técnicos, econômicos, ambientais e sociais. Uma parte significante dos recursos gastos para construir uma usina hidrelétrica é utilizada para mitigar os efeitos ambientais em peixes e na vida selvagem e na re-locação da infra-estrutura e da população afetadas pela inundação da região do reservatório da usina (RAMAKUMAR, 1998).

2.1.2 Transmissão e Distribuição

O propósito dos sistemas de transmissão e distribuição de energia elétrica é levar a energia elétrica das usinas de geração até os consumidores. Um sistema de corrente alternada de três fases é utilizado para a maioria das linhas de transmissão de energia.

A Figura 2.3 ilustra os conceitos de um sistema típico de transmissão e distribuição de energia elétrica. As usinas de geração produzem energia com uma tensão entre 15 e 25 kV. Esta tensão relativamente baixa não é adequada para a transmissão de energia por longas distâncias. As usinas de geração possuem um transformador que eleva essa tensão reduzindo a

corrente para diminuir as perdas de energia durante a transmissão e evitar um superaquecimento das linhas de transmissão devido à passagem da corrente elétrica.

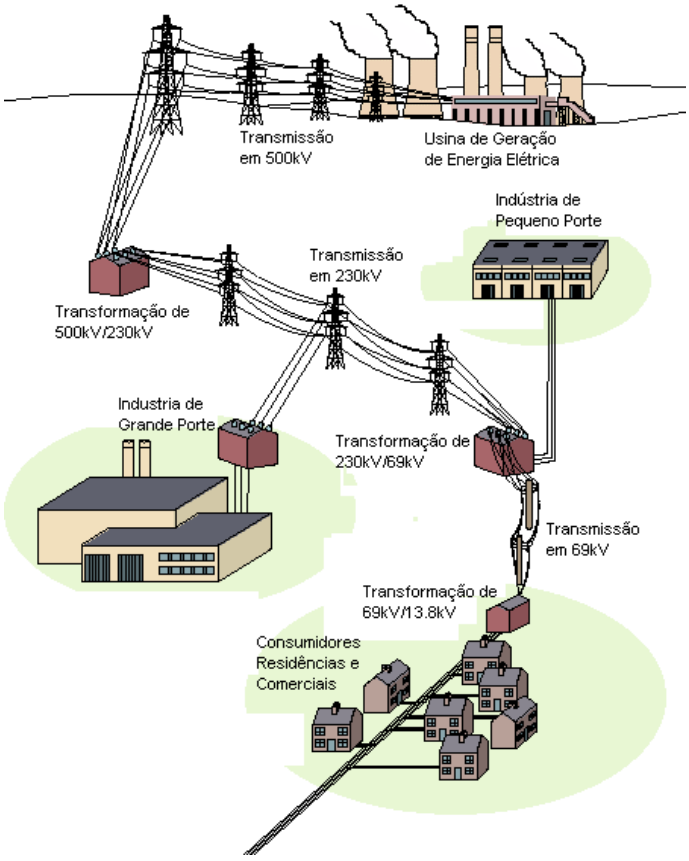

Figura 2.3: Esquema típico de transmissão e distribuição de energia

Na Figura 2.3, a tensão foi elevada para 500kV, e uma linha de transmissão de extra-alta tensão é utilizada para transmitir a energia gerada para uma subestação distante. Nessa subestação, a tensão é reduzida de 500kV para 230kV, e a energia é retransmitida utilizando linhas de alta tensão para subestações de alta tensão próximas das cidades. Nas subestações de alta tensão, a tensão é novamente reduzida, agora, de 230kV para 69kV. Depois disso, linhas de sub-transmissão (média tensão) levam a energia até as subestações de distribuição, onde a tensão é mais uma vez reduzida de 69kV para 13.8kV. Várias linhas de distribuição saem das subestações de distribuição em postes ou através de dutos subterrâneos e levam a energia até as ruas e avenidas. Antes de chegar até os consumidores residenciais, a energia passa por mais uma transformação de tensão de 13.8kV para 230/115 V para só então ser utilizada pelos clientes.

Existem clientes industriais que podem receber a energia em tensões mais altas que os clientes residenciais e comerciais. Geralmente esses clientes possuem as próprias subestações para reduzir a tensão a níveis desejáveis.

O sistema de transmissão precisa ser bastante robusto para permanecer funcionando mesmo que várias linhas de transmissão deixem de transmitir energia por algum motivo. Para tanto, são criadas redundâncias no sistema, fazendo com que as subestações se interconectem por vários caminhos diferentes.

2.1.3 Principais Equipamentos

Nesta seção, são apresentados alguns dos principais equipamentos utilizados nos sistemas de potência.

Chave

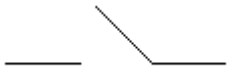

Figura 2.4: Representação gráfica de uma chave em um circuito

Chaves, cuja representação gráfica é exibida na Figura 2.4, são dispositivos mecânicos utilizados para alterar as conexões de um circuito ou isolar um circuito de sua fonte de energia. São normalmente utilizadas em subestações para isolar os equipamentos durante os períodos de manutenção. Tipicamente, uma chave é instalada em cada lado dos equipamentos para fornecer uma confirmação visual de que o circuito está aberto para a segurança do pessoal de manutenção (MCDONALD,1998).

Na Figura 2.5, podem ser observadas algumas chaves em uma subestação.

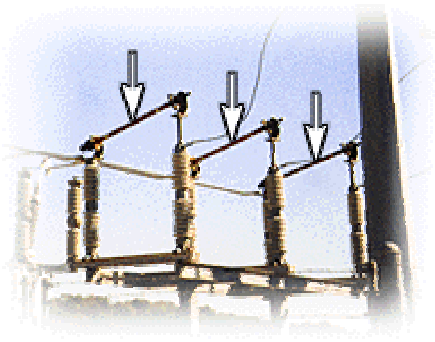

Figura 2.5: Chave seccionadora

Disjuntor

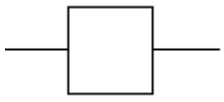

Figura 2.6: Representação gráfica de um disjuntor em um circuito

Disjuntores, cuja representação gráfica é exibida na Figura 2.6, são dispositivos mecânicos capazes de interromper a passagem da corrente em um circuito elétrico energizado, ou seja, durante a operação normal do circuito, impedindo que haja uma descarga elétrica pelo rompimento dielétrico do ar. Os disjuntores são geralmente classificados de acordo com o meio de interrupção utilizado para resfriar e alongar o arco elétrico, permitindo a interrupção. Os principais tipos de disjuntores são: disjuntores a óleo, disjuntores a vácuo e disjuntores a gás SF6 (hexafluorido de enxofre). Um disjuntor pode ser aberto automaticamente pela proteção do sistema elétrico durante uma falha no sistema, como um curto circuito, para evitar que os efeitos do problema se propaguem para o restante da rede elétrica (MCDONALD,1998). Na Figura 2.7, podem ser observados alguns disjuntores a gás instalados em uma subestação.

Figura 2.7: Disjuntores a gás

Linha de Transmissão

Linhas de transmissão são o meio físico utilizado para transportar a energia elétrica das usinas de geração até os consumidores finais. Existem vários tipos de linhas de transmissão, classificados de acordo com a tensão da energia que transportam. Os principais tipos de linhas são (KARADY, 1998)

- Linhas de alta tensão: são utilizadas, juntamente com as linhas de extra-alta tensão, para conectar usinas de geração a subestações e formam uma rede elétrica. A tensão de uma linha de alta tensão pode variar entre 100 e 230 kV; considera-se extra-alta as tensões acima de 230kV. O comprimento máximo de uma linha de alta tensão é de 320 quilômetros, e o de uma linha de extra-alta tensão é de 800 quilômetros.

- Linhas de sub-transmissão: geralmente são utilizadas para interconectar as subestações de alta tensão com as subestações de distribuição dentro de uma cidade. A tensão das linhas de sub-transmissão é sempre inferior a 115kV. O comprimento máximo de uma linha de sub-transmissão é de cerca de 90 quilômetros. A maioria das linhas de sub-transmissão está localizada em ruas ou avenidas.
- Linhas de distribuição: são utilizadas para levar a energia das subestações de distribuição até os transformadores nos postes das cidades. A tensão das linhas de distribuição é inferior a 25kV e podem ter no máximo 40 quilômetros.

Na Figura 2.8, pode ser observada uma torre de sustentação de uma linha de transmissão de alta tensão.

Figura 2.8: Torre de alta tensão

Barramento

Barramentos são linhas de transmissão internas das subestações e são utilizados para interconectar as diversas linhas que trazem a energia para a subestação com os diversos transformadores e linhas que recebem energia da subestação.

Vários fatores afetam a confiabilidade de uma subestação; um deles é o arranjo dos barramentos e dispositivos de interrupção (chaves e disjuntores). Além disso, o arranjo dos barramentos e dispositivos de interrupção afeta a manutenção, a proteção e o custo da subestação. A seguir, são apresentados os tipos de arranjos mais comuns (MCDONALD,1998):

- Barramento simples (Figura 2.9): neste arranjo há um barramento principal com todos os circuitos diretamente conectados a ele, com baixa confiabilidade, já que uma única falha no barramento afeta todo o sistema, porém o custo é muito baixo e necessita de pouco espaço na subestação.

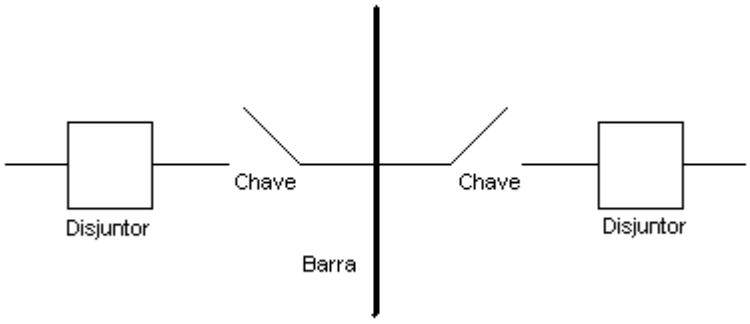

Figura 2.9: Arranjo de barramento simples

- Barramento duplo, disjuntor duplo (Figura 2.10): este arranjo fornece um alto nível de confiabilidade por possuir dois disjuntores distintos para cada circuito. Adicionalmente, com dois barramentos diferentes, falhas em apenas um deles, não compromete todo o sistema. A manutenção de um barramento ou de um disjuntor pode ser realizada sem interromper nenhum dos circuitos, com um custo muito alto e necessita do dobro do espaço de um arranjo com barramento simples.

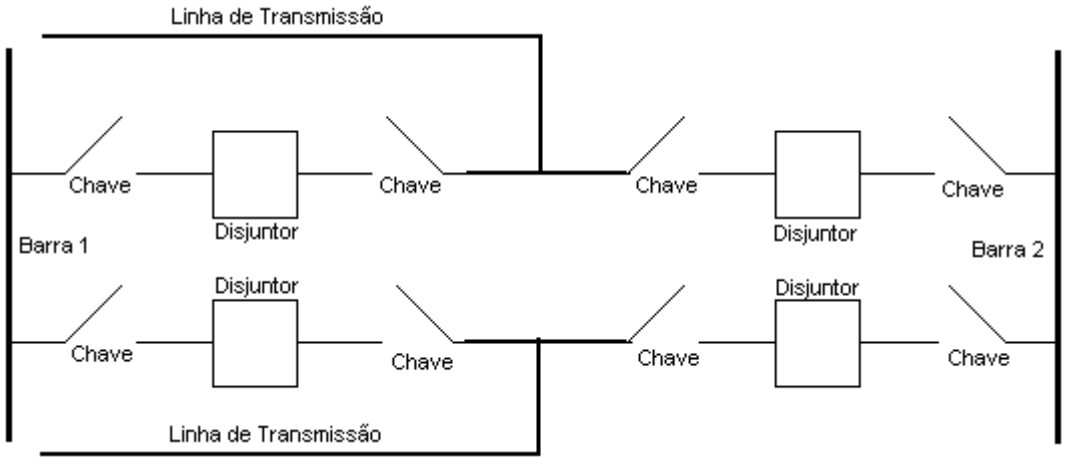

Figura 2.10: Arranjo de barramento duplo, disjuntor duplo

- Barramento principal e barramento de transferência (Figura 2.11): este arranjo apresenta todos os circuitos conectados entre um barramento principal e um barramento de transferência. Já que todos os circuitos estão conectados ao mesmo barramento, este arranjo apresenta a mesma confiabilidade do arranjo

de barramento simples, mas tem a vantagem de permitir que seja realizada a manutenção no barramento sem implicar um desligamento de todos os circuitos; durante a manutenção os circuitos passam a se conectar ao barramento de transferência. Apresenta um custo maior que o arranjo com barramento simples, mas fornece uma maior flexibilidade durante a manutenção.

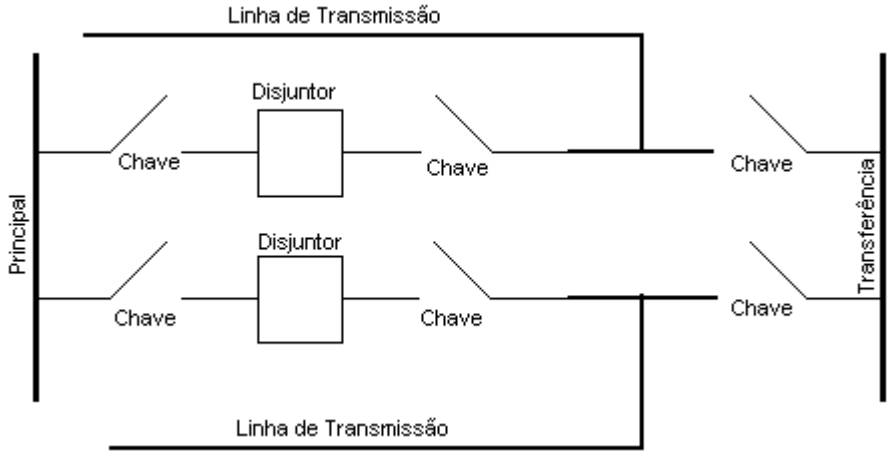

Figura 2.11: Arranjo com barramento principal e de transferência

- Barramento duplo, disjuntor simples (Figura 2.12): Este arranjo possui dois barramentos principais conectados a cada uma das linhas de transmissão e a um disjuntor de desempate. Utilizando o disjuntor de desempate na posição fechada permite a transferência de circuitos de um barramento para o outro através das chaves. Este arranjo permite que os circuitos operem utilizando qualquer um dos barramentos. Uma falha em um barramento não irá afetar o outro mas uma falha no disjuntor de desempate irá causar um desligamento de todo o sistema. Com o disjuntor de desempate operando na posição aberta são perdidas as vantagens de se utilizar dois barramentos principais. Nesse caso o sistema possui dois barramentos simples, descritos anteriormente, como baixo nível de confiabilidade.

- Barramento em Anel (Figura 2.13): Nesse arranjo, como o nome indica, todos os disjuntores são organizados em um anel. Se um circuito falhar, seus dois disjuntores adjacentes abrirão e o restante do sistema não será afetado. Da mesma forma, uma falha em um barramento afetará apenas os disjuntores adjacentes e o restante do sistema permanecerá energizado. A manutenção de

um disjuntor nesse arranjo pode ser realizada sem a interrupção de nenhum circuito.

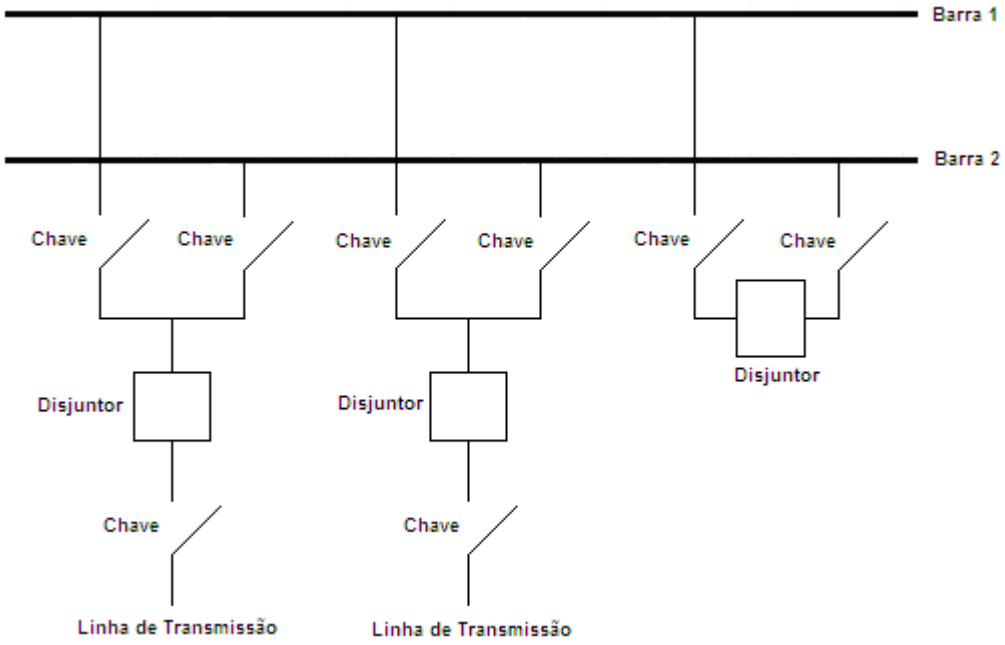

Figura 2.12: Arranjo com barramento duplo e disjuntor simples

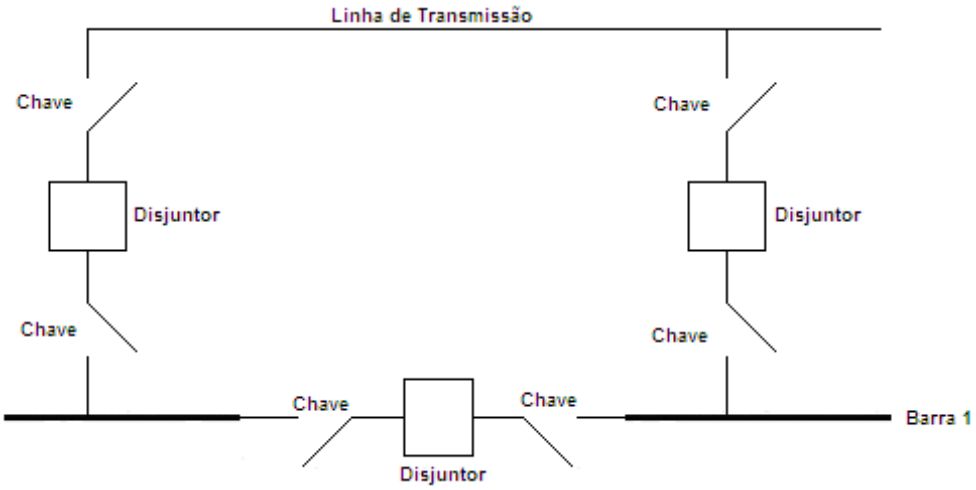

Figura 2.13: Arranjo com barramento em anel

- Disjuntor e meio (Figura 2.14): o arranjo com disjuntor e meio apresenta a característica de que cada circuito está entre dois disjuntores e possui dois barramentos principais. Uma falha em um dos circuitos provocará uma abertura nos dois disjuntores e não interferirá em quaisquer outros circuitos. A manutenção de qualquer disjuntor pode ser realizada sem a interrupção de

nenhum dos circuitos. Este é um dos arranjos com maior confiabilidade, pode ser expandido facilmente e apresenta custo e necessidade de espaço inferiores aos do arranjo com barramento duplo.

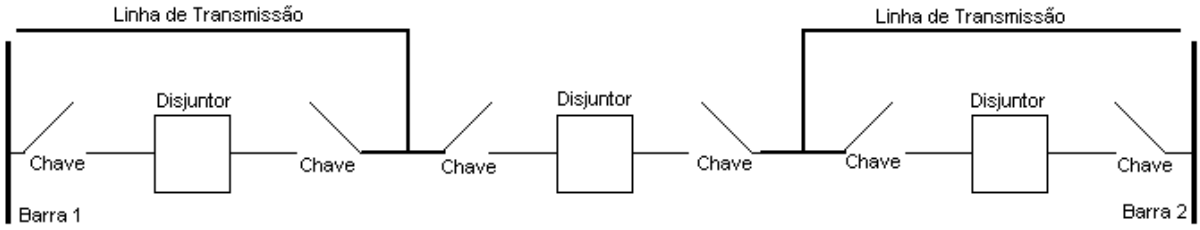

Figura 2.14: Arranjo com disjuntor e meio

A Tabela 2.1 apresenta uma comparação em termos de confiabilidade, custo e área necessária para instalação dos arranjos apresentados.

Tabela 2.1: Comparação dos arranjos de barramentos

Arranjo	Confiabilidade	Custo	Área
Barramento simples	Baixa	Baixo	Pequena
Barramento duplo	Alta	Alto	Grande
Barramento principal e de transferência	Baixa	Médio	Pequena
Barramento duplo, disjuntor simples	Média	Médio	Média
Barramento em Anel	Alta	Médio	Média
Disjuntor e meio	Alta	Médio	Grande

Transformador

Um transformador é definido como um dispositivo elétrico estático, ou seja, sem partes constantemente em movimento, usado em sistema de potência para transferir a potência entre circuitos através do uso de indução eletromagnética. Os sistemas de potência tipicamente dispõem de um grande número de locais de geração, pontos de distribuição e interconexões com o próprio sistema e com outros sistemas. A complexidade do sistema acarreta em uma variedade de tensões de transmissão e distribuição. Os transformadores devem ser utilizados em cada um dos pontos onde há uma transição entre os níveis de tensão, e podem realizar dois tipos de operação: elevar ou diminuir a tensão. A elevação da tensão geralmente só é realizada na usina de geração; já a diminuição da tensão é usada para alimentar os diversos circuitos de transmissão e distribuição (HARLOW, 1998.

A Figura 2.15 exibe um transformador de grande porte instalado em uma subestação.

Figura 2.15: Transformador em uma subestação

Reator

Os reatores são utilizados para prover reatância indutiva aos circuitos de potência para uma grande variedade de propósitos, cujos aspectos técnicos fogem ao escopo deste trabalho. Podem ser utilizados em qualquer nível de tensão (HARLOW, 1998).

Gerador

Geradores elétricos são dispositivos capazes de converter alguma forma de energia em energia elétrica. Os mais comuns são os geradores eletromecânicos que convertem a energia mecânica em energia elétrica. Exemplos de geradores eletromecânicos são os utilizados em usinas hidrelétricas, que convertem a energia mecânica gerada pelas turbinas, devido ao movimento de um fluido, em energia elétrica.

2.2 Sistemas SCADA

O gerenciamento da energia elétrica é realizado em centros de controle por sistemas computadorizados chamados *Energy Management Systems* (EMS). A aquisição de dados e controle remoto é realizada por sistemas SCADA. Um EMS geralmente inclui uma interface para o sistema SCADA através da qual é possível estabelecer comunicação com as usinas de geração, subestações e outros dispositivos remotos. A Figura 2.16 ilustra a camada de aplicação de um EMS moderno; além disso, são exibidas também as camadas sobre as quais o EMS está estruturado: o sistema operacional, um gerenciador de banco de dados e uma camada de serviços e bibliotecas (WOLLENBERG, 1998).

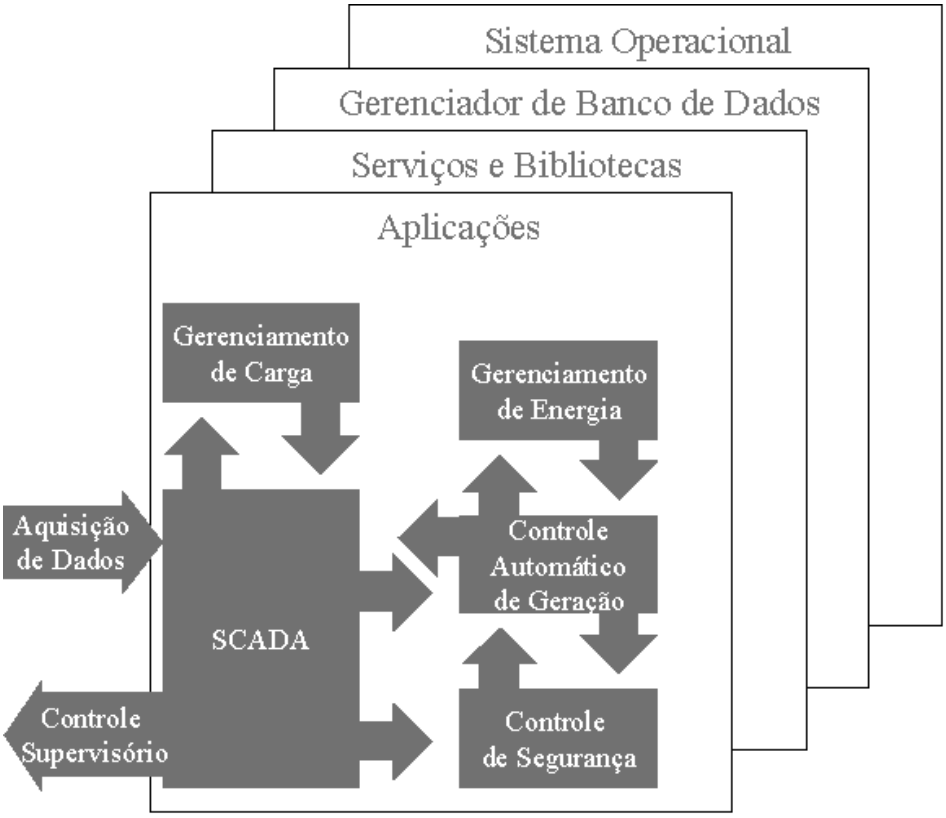

Figura 2.16: Camadas de um EMS moderno

Um sistema SCADA consiste de uma estação-mestre que se comunica com unidades terminais remotas (UTRs) para permitir que os operadores possam monitorar e controlar usinas e subestações. As UTRs transmitem o estado dos dispositivos e valores de medições analógicas para a estação-mestre e recebem comandos de controle da mesma.

As funções tradicionais de um sistema SCADA são (WOLLENBERG, 1998):

- Aquisição de Dados: fornece ao operador dados e medições dos dispositivos da rede elétrica;
- Controle Supervisório: permite que o operador possa controlar remotamente os dispositivos, por exemplo, comandando a abertura de um disjuntor;
- *Tagging*: bloqueia um dispositivo impedindo operação não autorizadas;
- Alarmes: notificam o operador sobre eventos não planejados e condições anormais de operação.

A estação-mestre de um sistema SCADA é crítica para a operação dos sistemas elétricos. Para garantir um melhor desempenho e tolerância a falhas, suas funcionalidades são geralmente distribuídas em mais de um computador. A configuração mais comum é a

utilização de um sistema com dois computadores, um primário e um secundário, executando o sistema ao mesmo tempo.

A seguir, são comentadas as principais aplicações de um EMS moderno (WOLLENBERG, 1998).

2.2.1 Controle Automático de Geração

O Controle Automático de Geração (CAG) possui duas funções principais que operam em tempo real para ajustar a geração de acordo com a carga. As funções principais do CAG são controle de freqüência e despacho econômico, comentadas a seguir.

Controle de Carga-Freqüência

O Controle de Carga-Freqüência precisa atingir três objetivos, descritos a seguir em ordem de prioridade:

- Manter a freqüência de geração no valor específico (50/60Hz);
- Manter o intercâmbio de energia com áreas de controle vizinhas no valor configurado;
- Manter a alocação de energia nas unidades de acordo com valores economicamente desejáveis.

Despacho Econômico

As unidades geradoras têm diferentes custos de geração; é necessário achar o nível de geração de cada uma delas de modo que a potência seja gerada a custo mínimo. O objetivo do Despacho Econômico é definir quanto cada unidade deve gerar para que o custo de geração seja mínimo.

Além do custo de geração existem outros fatores que devem ser levados em consideração para a obtenção de um esquema ótimo de geração de energia. Um deles é que o esquema de geração deve prover margens de reserva adequadas. Isso é feito através do estabelecimento de um limite para o nível máximo de geração inferior à capacidade de geração da unidade. Outro fator a ser considerado é o limite do sistema de transmissão de energia. Em alguns casos, é possível que o esquema mais econômico não seja possível devido aos limites do sistema de transmissão.

2.2.2 Gerenciamento de Carga

Os sistemas SCADA, com suas UTRs instaladas nas subestações de distribuição, podem fornecer o status e as medições dos equipamentos da subestação. Os sistemas de automação de distribuição de energia e gerenciamento de carga podem monitorar dispositivos seccionadores de circuitos (chaves e disjuntores), operar chaves para a reconfiguração de circuitos, controlar a tensão nas linhas, implementar um esquema de tarifação dependente do tempo e alternar a carga entre os diversos consumidores para balancear a carga do sistema de distribuição.

2.2.3 Gerenciamento de Energia

O Despacho Econômico minimiza o custo da produção de energia. O Gerenciamento de Energia atua em um nível mais alto que o Despacho Econômico, sendo o responsável por controlar, em nível global e de forma economicamente vantajosa, a geração e a transmissão de energia. Por exemplo, a água armazenada em um reservatório de uma usina hidrelétrica é um recurso que pode ser mais valioso no futuro e, portanto, não deve ser utilizado mesmo que o custo da geração de energia hidroelétrica seja mais baixo que o custo de geração de energia termoelétrica. As considerações globais são necessárias devido à capacidade de vender e comprar energia de outros sistemas de geração interconectados, pois, em alguns casos, pode ser mais vantajoso comprar que produzir energia.

2.2.4 Controle de Segurança

Os sistemas de geração, transmissão e distribuição de energia são projetados para suportarem as primeiras contingências. Uma contingência é definida como um evento que indica que um ou mais componentes importantes do sistema, como linhas de transmissão, geradores ou transformadores, foram inesperadamente removidos do sistema. Sobrevivência significa que o sistema continua a funcionar em níveis aceitáveis de tensão e freqüência e sem perda de carga. A operação normal do sistema pode sofrer um enorme número de contingências, e muitas delas não podem ser previstas.

O Controle de Segurança analisa modelos topológicos da rede elétrica para determinar as condições do sistema. Para isso são necessárias algumas aplicações, listadas a seguir.

- **Processador de Topologia**: processa as medições do estado dos equipamentos da rede elétrica em tempo real para criar um modelo de conectividade elétrica do sistema;

- **Estimador de Estado**: utiliza as medições do estado dos equipamentos da rede elétrica em tempo real e medições analógicas para determinar a melhor estimativa para o estado do sistema elétrico;
- **Fluxo de Potência Ótimo:** recomenda ações de controle para otimizar objetivos específicos sujeitos a limitações do sistema elétrico como custo de operação do sistema e perdas;
- **Análise de Contingência:** analisa o impacto de um conjunto de contingências no estado do sistema elétrico e identifica contingências potencialmente danosas que podem causar violações dos limites de operação do sistema.

2.3 A Companhia Hidro Elétrica do São Francisco (CHESF)

Com mais de 50 anos de atuação, a Companhia Hidro Elétrica do São Francisco - CHESF - é uma das maiores e mais importantes empresas do setor elétrico brasileiro. Ela atua na geração, transmissão e comercialização de energia elétrica, suprindo, principalmente, oito estados nordestinos: Alagoas, Bahia, Ceará, Paraíba, Pernambuco, Piauí, Rio Grande do Norte e Sergipe. Sua área de abrangência é de 1,2 milhão de quilômetros quadrados, o equivalente a 14,3% do território brasileiro, beneficiando mais de 40 milhões de habitantes.

Desde sua origem, a CHESF tem um papel de extrema relevância no desenvolvimento da região Nordeste. Ela foi criada pelo Decreto-Lei 8.031 de 3 de outubro de 1945, como uma sociedade de economia mista ligada ao Ministério da Agricultura, e teve suas atividades iniciadas, efetivamente, em 15 de março de 1948. Sua criação foi baseada na carência de energia elétrica da região Nordeste, que passou a ser suprida com o aproveitamento do potencial hidrelétrico do rio São Francisco. Seu idealizador foi o engenheiro agrônomo Apolônio Sales, Ministro da Agricultura no governo Getúlio Vargas.

No restante dessa seção são apresentados dados sobre os sistemas de geração e transmissão de energia elétrica da CHESF. Todos os dados apresentados foram coletados no *site web* da CHESF, http://www.chesf.gov.br.

2.3.1 Sistema de Geração

O sistema de geração da CHESF é composto por 14 usinas hidrelétricas, duas termelétricas e uma eólica, totalizando uma potência nominal instalada de 10,7 milhões de kW, a maior do setor elétrico brasileiro. As usinas hidrelétricas representam cerca de 96% da

potência total instalada na CHESF, e a maior parte delas está situada no Rio São Francisco. São 68 geradores, sendo 54 hidráulicas, 10 térmicas e 4 eólicas.

Na Tabela 2.2, são apresentadas cada uma das usinas geradoras de energia da CHESF e suas respectivas capacidades de geração.

Tabela 2.2: Capacidades geradoras das usinas da CHESF

Usinas	Geradores	Potencia Instalada (kW)	Total
Hidrelétricas	54		10.268.328
Paulo Afonso I	3	60.000	180.001
Paulo Afonso II A	3	75.000	215.000
Paulo Afonso II B	3	76.000	228.000
Paulo Afonso III	4	198.550	794.200
Paulo Afonso IV	6	410.000	2.462.400
Sobradinho	6	175.050	1.050.300
Apolônio Sales (Moxotó)	4	100.000	400.000
Boa Esperança I	2	55.000	110.000
Boa Esperança II	2	63.650	127.300
Funil	3	10.000	30.000
Pedra	1	20.007	20.007
Araras	2	2.000	4.000
Coremas	2	1.760	3.800
Piloto	1	2.000	2.000
Luiz Gonzaga (Itaparica)	6	246.600	1.479.600
Xingo	6	527.000	3.162.000
Termelétricas	10		434.970
Camaçari II	5	58.500	292.500
Bongi	5	28.494	142.470
Eólicas	4		1.200
Mucuripe	4	300	1.200
Outras			340
Total	68		10.704.838

Na Figura 2.17, é exibida a localização de cada uma das usinas hidrelétricas e termelétricas da CHESF.

Figura 2.17: Mapa das usinas geradoras da CHESF

2.3.2 Sistema de Transmissão

A CHESF possui o maior sistema de transmissão de energia elétrica do país, com mais de 17 mil quilômetros de linhas de transmissão em 500, 230, 138 e 69 kV. As suas 87 subestações são responsáveis por fornecer a energia produzida pelas usinas às concessionárias e aos grandes complexos industriais da região. Na Figura 2.18, é exibido um mapa do sistema de transmissão da CHESF, onde são mostradas as suas principais linhas de transmissão.

Figura 2.18: Mapa do sistema de transmissão da CHESF

2.4 O SAGE

O SAGE (Sistema Aberto de Gerenciamento de Energia) é um software para controle e supervisão do processo de geração, transmissão e distribuição de energia elétrica. Foi desenvolvido pelo CEPEL (Centro de Pesquisas de Energia Elétrica) e se encontra instalado em várias empresas de energia brasileiras. Em particular, a CHESF vem utilizando com sucesso o SAGE já há alguns anos, embora ainda haja algumas subestações ligadas ao sistema legado Gould[3].

Tendo em vista que o SAGE controla e supervisiona sistemas de geração, transmissão e distribuição de energia elétrica, sua operação está centrada em **dados** que descrevem o estado dos componentes/equipamentos que formam tais sistemas. Deste modo, a forma como o SAGE organiza sua **base de dados** (ou simplesmente **base**) é crucial para o bom entendimento do software. Após a descrição de como o SAGE organiza sua base de dados, serão discutidos os serviços que juntos fornecem a funcionalidade de controle e supervisão de sistemas elétricos.

2.4.1 Base de Dados do SAGE

O SAGE utiliza o modelo relacional para organizar os dados que armazena, no sentido que tabelas representam **entidades, atributos** e **relacionamentos**. Entretanto, ao contrário da maioria dos sistemas baseados no modelo relacional, o módulo de controle e supervisão do SAGE não utiliza um Sistema Gerenciador de Banco de Dados (SGBD) para armazenar seus dados, mas a chamada **Base de Tempo Real**. Para garantir rapidez, a Base de Tempo Real reside integralmente na memória principal dos computadores que rodam o SAGE. Técnicas de replicação ativa são utilizadas para garantir a persistência e disponibilidade dos dados (CEPEL, 2002).

A escolha de um mecanismo próprio para gerência da base se deveu às necessidades de tempo de acesso muito rápido e de tolerância a falhas que o SAGE apresenta. SGBDs utilizam discos para o armazenamento persistente, o que compromete o pior caso do tempo de acesso aos dados (quando os dados requeridos não estão em cache). Além disso, SGBDs tipicamente dependem do funcionamento da uma máquina, o que dificulta/encarece o oferecimento de altos níveis de tolerância a falhas.

---

[3] Gould é o nome do antigo sistema de gerência utilizado nos centros de controle da CHESF e que está sendo substituído pelo SAGE.

A base de dados empregada pelo SAGE contém dados sobre a **configuração** do sistema, bem como sobre sua **operação**. Dados de configuração dizem respeito (CEPEL, 2002):

a) Aos equipamentos que compõem o sistema elétrico e sua organização topológica;

b) Aos mecanismos de transmissão de dados que descrevem o estado de tais equipamentos;

c) Aos computadores usados na execução do próprio SAGE e como estes computadores estão organizados. Como exemplos de dados de configuração, considere: "na subestação ABC, há o disjuntor 15L4-ABC", "o disjuntor 15L4-ABC está ligado à linha de transmissão 05L4-ABC/DEF".

Dados de operação descrevem eventos observados no sistema elétrico ou no sistema computacional que executa o SAGE. Um exemplo de um dado de operação seria "às 14:13:12 do dia 01/02/03, o disjuntor 15L4-ABC abriu". Note que dados de operação são adicionados/alterados ao SAGE com muito mais rapidez que os dados de configuração.

Quando o SAGE inicia sua execução, ele carrega os dados de configuração da **Base de Referência,** a qual é uma imagem em disco dos dados de configuração que estarão contidos na Base de Tempo Real. Já os dados de operação armazenados na Base de Tempo Real serão obtidos durante a execução do SAGE. A Base de Referência é codificada utilizando-se o padrão XDR (*External Data Representation Standard*), o que garante a neutralidade dos dados em relação à representação usada por diferentes arquiteturas de computadores (Intel x86 × Sun Sparc, por exemplo).

Por estar codificada em XDR e ser uma imagem da Base de Tempo Real, a Base de Referência não é de fácil manipulação por seres humanos. Isto cria a necessidade de uma forma mais adequada a humanos para a descrição dos dados de configuração do sistema. Essa necessidade é suprida através da **Base Fonte**, que usa um modelo de dados voltado para descrição da configuração do sistema (topologia e características do sistema elétrico, canais de comunicação de dados e ambiente computacional do SAGE).

O administrador do SAGE define e mantém a configuração do sistema alterando a Base-fonte. Quando concluídas e validadas quaisquer modificações, a Base-fonte é convertida/compilada, gerando a Base de Referência. Quando o SAGE é iniciado, a Base de Referência é então carregada na Base de Tempo Real, que é o repositório de dados efetivamente usado pelo SAGE.

A Figura 2.19 mostra a relação entre as três bases usadas no SAGE, bem como quais programas fazem a conversão entre as bases. É interessante notar que CARGBF e STI_convTxtXDR são programas Windows, ao passo que STI_EqualizaBase e o módulo de controle e supervisão do SAGE executam em Unix (Solaris, Linux, etc.).

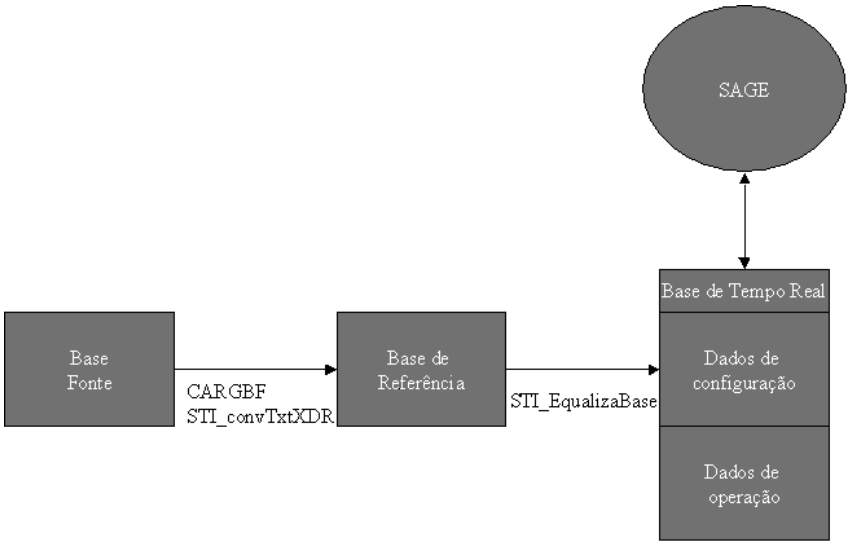

Figura 2.19:Relação entre as diferentes bases utilizadas pelo SAGE

2.4.2 Serviços que Compõem o SAGE

O SAGE pode ser funcionalmente decomposto em cinco serviços principais (CEPEL, 2002):

- Serviço de Difusão Confiável;
- Serviço de Alarmes e Eventos;
- Serviço de Comunicação de Dados;
- Serviço de Aquisição de Dados e Controle Supervisório;
- Serviço de Distribuição de Dados.

O **Serviço de Difusão Confiável** constitui-se no serviço mais básico do SAGE. É ele que implementa a Base de Tempo Real, usando replicação ativa para que o SAGE tolere falhas nas máquinas que compõem o sistema. Mais precisamente, o módulo Gerenciador de Memória Compartilhada Distribuída (GMCD) utiliza o serviço de replicação ativa para fornecer a abstração de Memória Compartilhada Distribuída (MCD). Uma MCD é um segmento de memória compartilhada, cujas alterações são propagadas a todos os computadores que participam do grupo de replicação da MCD. A propagação das alterações é

feita de forma a garantir que cada instância de processo, usando a MCD, veja as alterações na mesma **ordem**. Os dados da Base de Tempo Real são, portanto, armazenados em MCDs.

Além de fornecer a abstração de MCD, o Serviço de Difusão Confiável também exporta os mecanismos de comunicação em grupo usados para garantir replicação ativa. Isto permite que os demais serviços possam também ser replicados, oferecendo assim grande tolerância a falhas.

É importante notar que para o correto funcionamento do Serviço de Difusão Confiável, o administrador do SAGE precisa configurar a Base-fonte corretamente (e proceder com sua conversão em Base de Referência, seguido da carga desta última na Base de Tempo Real). Por exemplo, o administrador precisa definir quais computadores atuarão como nó do sistema SAGE (entidade NOH), quais processos estarão executando (entidade PRO) e quais os nós que replicam um dado processo (entidade INP). A necessidade de configuração também se aplica aos demais serviços que compõem o SAGE.

O **Serviço de Alarmes e Eventos** recebe, processa e armazena alarmes e eventos oriundos do sistema elétrico, sendo supervisionado ou do próprio ambiente computacional que executa o SAGE.

Eventos e alarmes têm um ciclo de vida estabelecido na configuração do SAGE (via alterações na Base-fonte, como discutido acima). Esta definição de ciclo de vida permite estabelecer como e se um alarme pode ser eliminado do sistema, bem como se é necessário que o usuário reconheça o registro de dados (tal reconhecimento é feito através do visor de alarmes).

O **Serviço de Comunicação de Dados** é responsável pela coleta e distribuição das informações (eventos e alarmes) referentes ao sistema elétrico. Uma característica importante do Serviço de Comunicação de Dados é a capacidade de coletar dados diretamente do sistema elétrico ou indiretamente, distribuídos por outros Centros de Supervisão e Controle (que podem ou não estar usando o SAGE). Esta característica permite que o SAGE seja utilizado nos diferentes níveis hierárquicos da supervisão e controle do sistema elétrico (local, regional e central).

Como a vasta maioria dos serviços de redes de computadores, o Serviço de Comunicação de Dados do SAGE se encontra organizado em camadas, ilustradas na Figura 2.20:

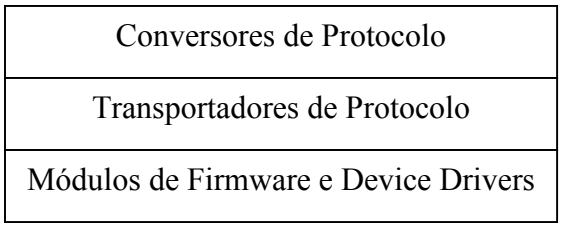

Figura 2.20: Camadas do Serviço de Comunicação de Dados do SAGE

Os Módulos de Firmware e Device Drivers implementam a transferência de bits propriamente dita, correspondendo, portanto, à camada física do modelo de redes ISO/OSI (ISO/IEC, 1984). Os Transportadores de Protocolo permitem que os usuários do Serviço de Comunicação de Dados possam abstrair os protocolos de enlace, rede e transporte usados.

Os Conversores de Protocolo, por sua vez, exportam uma interface única para os serviços de aquisição e distribuição de dados, como também para operações de controle supervisório. São eles que encapsulam os detalhes de como cada equipamento ou centro de controle formata os dados enviados, permitindo que o resto do sistema lide apenas com serviços abstratos de aquisição e distribuição de dados e de controle.

Vale também ressaltar que o SAGE disponibiliza dois utilitários (SIM_TR e SIM_SD) que simulam Conversores de Protocolo, possibilitando a execução do SAGE em modo de teste, sem conexão com um sistema elétrico real.

O **Serviço de Aquisição de Dados e Controle Supervisório** utiliza os serviços dos Conversores de Protocolo para obter, processar e armazenar os registros de dados originados diretamente na rede elétrica, ou distribuídos de outros Centros de Supervisão e Controle. Além disso, este serviço também transporta atuações em equipamentos remotos especificadas pelo operador do sistema. Um aspecto interessante do Serviço de Aquisição de Dados e Controle Supervisório é a possibilidade da definição de Pontos Lógicos Calculados arbitrários, através de codificação na linguagem C. Pontos Lógicos Calculados são componentes virtuais do sistema elétrico, calculados a partir de registros de dados obtidos pelo sistema.

O **Serviço de Distribuição de Dados** também utiliza Conversores de Protocolo para obter dados que devem ser distribuídos (repassados) para outros Centros de Supervisão e Controle. É interessante notar que Conversores de Protocolo também são utilizados no envio dos dados. Portanto, pode-se observar que o Serviço de Distribuição de Dados faz o

roteamento dos dados entre os diversos níveis hierárquicos do sistema de supervisão e controle da empresa.

2.5 Considerações finais

O objetivo deste capítulo foi fornecer o conjunto mínimo de conhecimentos da engenharia elétrica e do gerenciamento de sistemas de potência necessários para que um cientista da computação possa compreender seu funcionamento e desenvolver software para essa área. Além disso, foram apresentados dados sobre a CHESF, parceira da UFCG no desenvolvimento deste trabalho, e sobre o SAGE, sistema com o qual o software desenvolvido durante este trabalho de Mestrado tem uma profunda relação, o que ficará claro mais adiante.

Propositalmente não se incluíram aspectos técnicos dos sistemas elétricos ou provas e fundamentações matemáticas e físicas uma vez que tais informações não foram necessárias para o desenvolvimento do trabalho e, possivelmente, iriam apenas dificultar a compreensão do assunto.

3 Correlação de Eventos

O objetivo deste capítulo é fornecer uma visão geral sobre a correlação de eventos, incluindo o estabelecimento de um conjunto de conceitos básicos (seção 3.1) necessários para o bom entendimento do trabalho desenvolvido. A seção 3.2 oferece uma visão panorâmica das principais técnicas de correlação de eventos encontradas na literatura; a seção 3.3, exibe algumas das principais ferramentas comerciais para o tratamento de eventos, enquanto a seção 3.4 comporta algumas considerações finais sobre o assunto abordado.

3.1 Conceitos Básicos

Uma das áreas mais importantes da gerência de redes, sejam elas de computadores, de telecomunicações ou redes elétricas, consiste na gerência das falhas que ocorrem durante o seu funcionamento. A gerência de falhas compreende a detecção, diagnóstico e remoção das falhas. Qualquer uma dessas funções só pode ser realizada pela adição de valor aos dados coletados durante a monitoração da rede (MANSFIELD, 1993).

3.1.1 Eventos e Alarmes

Um **evento** é uma ocorrência instantânea em um momento no tempo (HASAN, 1999). Tipicamente um evento está relacionado ao objeto no qual ele ocorreu. Os objetos onde ocorrem eventos são chamados objetos gerenciados. Um **objeto gerenciado** é uma entidade com algum estado; para modelar o estado de um objeto gerenciado, ele pode ser tratado como uma coleção de **atributos**, cujos valores podem mudar com o tempo.

Os eventos podem ser classificados como **primitivos** ou **compostos.** Eventos que são gerados diretamente por (ou correspondem a) uma mudança no estado de um objeto gerenciado podem ser considerados como primitivos já que eles são **diretamente observáveis.** Por outro lado, os eventos que não podem ser diretamente detectados por hardware ou software, ou porque o objeto onde os eventos ocorreram não está sendo monitorado ou o evento não foi reportado, são chamados de compostos ou **não-observáveis**.

Um evento não-observável, cuja ocorrência é estabelecida por inferência a partir do padrão de ocorrência dos eventos aos quais ele é correlato, é geralmente uma construção conceitual. Alguns eventos correlatos a um evento não-observável podem ser também não-observáveis. Uma das funções da correlação de eventos é estabelecer eventos não-observáveis a partir dos eventos observáveis que são reportados

Um **alarme** é um evento que é associado a um estado anormal da rede, ou seja, um comportamento da rede ou de um de seus componentes que difere do esperado; alarmes são tipicamente reportados aos operadores do sistema. Esse desvio de comportamento pode ser atribuído a falhas de hardware ou software, a erros humanos, erros de projeto ou a uma combinação de todos eles.

3.1.2 Diagnóstico de Falhas

O **diagnóstico de falhas** é um estágio no processo de gerência de falhas que consiste em encontrar a **causa raiz** (*root cause*) para os sintomas recebidos (representados por eventos). Antes de descobrir a causa raiz, talvez seja necessário formular um conjunto de hipóteses de falha, que precisarão ser validadas por testes (KATZELA, 1995). É desejável que o sistema de diagnóstico de falhas tenha um modelo da configuração gerenciada, processe o fluxo de eventos em tempo real e seja capaz de trabalhar com dados incompletos.

3.1.3 Correlação de Eventos

A **correlação de eventos** consiste na interpretação conceitual de múltiplos eventos, resultando na atribuição de um novo significado aos eventos originais (JAKOBSON, 1993). Como parte do processo de correlação, os dados brutos são interpretados e analisados, levando-se em conta um critério, por exemplo, proximidade temporal ou relações conhecidas de causalidade. Portanto, a correlação adiciona valor aos eventos originais e é um mecanismo importante na gerência de redes. A correlação de eventos reduz o tempo necessário para a identificação das falhas, permitindo uma restauração mais rápida dos serviços afetados.

Um sistema de correlação de eventos pode desempenhar as seguintes funções:

- **Detecção e Diagnóstico das Falhas:** deve-se considerar uma rede elétrica simples onde linhas de transmissão estão conectadas a barramentos através de disjuntores e envolvendo reatores. Nesta rede supervisionada, podem ser gerados eventos associados aos barramentos, como desligamento de barramento (B_i), ou às linhas de transmissão, como desarme de linha de transmissão (L_i); ambos podem ser atribuídos

a uma falha num disjuntor (D_1) ou falha de um reator (R_1). Sem esse conhecimento, definido como **D_1 ou R_1 => B_i ou L_i**, um operador de rede ou um sistema automático não pode, eficiente e efetivamente, apontar a causa real dos alarmes de falha;

- **Filtragem de Eventos:** no exemplo acima, o conhecimento do relacionamento entre os eventos possibilita que um sistema automático de correlação de eventos possa reportar os eventos realmente importantes enquanto suprime os potencialmente numerosos eventos redundantes de falhas;
- **Ajuste de Performance:** quando as causas reais das falhas da rede podem ser isoladas eficiente e efetivamente, as degradações de performance têm vida curta, e os objetos gerenciados relacionados com os eventos de falha podem ser ajustados efetivamente.

3.1.4 Tipos de Correlação

Vários **tipos de correlação** podem ser identificados de acordo com as operações executadas sobre os eventos (JAKOBSON, 1995). Os tipos mais comuns são comentados a seguir:

3.1.4.1 Compressão

A correlação por compressão consiste em detectar, nos eventos recebidos em uma dada janela de tempo, ocorrências múltiplas de um mesmo evento, substituindo os eventos repetidos por um único evento, possivelmente indicando o número de repetições durante o período de observação.

3.1.4.2 Supressão Seletiva

A correlação por supressão seletiva é a inibição temporária da notificação de um dado tipo de evento, de acordo com um critério, avaliado continuamente pelo sistema de correlação, relacionado com o contexto dinâmico do processo de gerência da rede. O critério de supressão é ligado à presença de outros eventos, ao relacionamento temporal entre os eventos ou às prioridades estabelecidas pelos gerentes da rede.

3.1.4.3 Filtragem

A correlação por filtragem consiste em suprimir um dado evento, dependendo dos valores de um conjunto de parâmetros previamente especificados. Na forma mais simples, a filtragem leva em conta apenas os parâmetros dos eventos que estão sendo filtrados. Uma

forma mais elaborada, conhecida como filtragem inteligente, pode compreender vários tipos de operação como compressão e supressão.

3.1.4.4 Contagem

A correlação por contagem consiste na geração de um novo evento cada vez que o número de ocorrências de um determinado tipo de evento ultrapassa um limiar previamente estabelecido.

3.1.4.5 Escalação

A correlação por escalação consiste, dependendo do contexto operacional, na supressão de um evento e criação de outro em seu lugar com um de seus parâmetros (por exemplo, a gravidade), assumindo um valor maior. O contexto operacional inclui, entre outros fatores, a presença de outros eventos, o relacionamento temporal entre os eventos, o número de ocorrências de um evento em uma dada janela de tempo e as prioridades estabelecidas pelos gerentes da rede.

3.1.4.6 Generalização

A correlação por generalização consiste em substituir um evento, dependendo do contexto operacional, por um evento correspondente a sua **super-classe**. Por exemplo, na ocorrência simultânea de eventos sobre diversas falhas associadas a uma linha de transmissão, cada um dos eventos originais pode ser substituído por um outro, indicando um defeito na linha; em seguida, utilizando a correlação por compressão, todos os eventos repetidos podem ser substituídos por um único evento.

Esta operação é baseada no raciocínio indutivo, permitindo a expansão do escopo de conhecimento, ao custo do aumento da complexidade do problema e da introdução de um certo grau de incerteza nos resultados da correlação (MEIRA, 1997).

Existem dois tipos principais de generalização: generalização por simplificação de condições e generalização baseada em instâncias. No primeiro caso, para que os eventos de uma classe inferior possam ser substituídos por eventos de uma classe superior, uma ou mais condições definidas como necessárias para sua identificação são ignoradas ou negligenciadas. No segundo caso, um novo evento pode ser gerado a partir da associação das informações de dois ou mais eventos recebidos.

3.1.4.7 Especialização

A correlação por especialização é o inverso da correlação por generalização e consiste em substituir um evento por um outro correspondente a sua **sub-classe**. Esta operação, baseada em raciocínio dedutivo, não adiciona nenhuma informação nova além das que já estão implicitamente presentes nos eventos originais, mas é útil para evidenciar as conseqüências de um evento.

Por exemplo, um sistema de correlação por especialização pode gerar, quando uma determinada linha de transmissão é desconectada do restante da rede elétrica, um evento para cada problema gerado pela desconexão da linha. Então, através da especialização, as conseqüências da desconexão da linha de transmissão se tornam mais evidentes.

3.1.4.8 Relacionamento Temporal

A correlação por relacionamento temporal depende da ordem ou do momento em que os eventos são recebidos ou gerados. Muitas relações temporais podem ser definidas utilizando conceitos como: ANTES, DURANTE, DEPOIS, INICIO, FIM, COINCIDE etc (JAKOBSON, 1995).

3.2 Técnicas de Correlação de Eventos

Esta seção apresenta uma visão panorâmica das principais técnicas de correlação de eventos encontradas na literatura. Atualmente o número de técnicas de correlação de eventos é muito grande. Algumas abordagens são probabilísticas, outras utilizam paradigmas da inteligência artificial convencional e outras aplicam princípios definidos em lógicas não convencionais. Existem também abordagens que adotam métodos *ad hoc* para lidar com a correlação de eventos.

3.2.1 Raciocínio Baseado em Regras

Muitas das abordagens apresentadas na literatura sobre correlação de eventos são variações da abordagem clássica do raciocínio baseado em regras. Nesta abordagem, o conhecimento geral sobre uma determinada área é representado por um conjunto de regras, e o conhecimento específico, relevante para uma situação em particular, é constituído por fatos, representados por acepções e armazenado em um banco de dados.

Uma regra é formada por duas expressões ligadas por um operador de implicação ($\Rightarrow$) e que trabalham sob uma base de dados global. O lado esquerdo representa os pré-requisitos que precisam ser satisfeitos pela base de dados para que a regra seja aplicável. O lado direito descreve a ação que deve ser realizada caso a regra seja satisfeita. A aplicação de uma regra pode alterar a base de dados.

Todo sistema de raciocínio baseado em regras (ou sistema de produção) tem estratégias de controle que determinam em que ordem as regras aplicáveis serão aplicadas e que interrompe o processo de computação quando uma condição de parada é satisfeita.

Existem dois modos de operação de um sistema baseado em regras: no primeiro, o *forward mode*, a computação começa por um estado inicial e constrói uma seqüência de passos que leva à solução do problema. Em um sistema de correlação de eventos, as regras poderiam ser aplicadas sobre a base de dados que contém todos os eventos recebidos enquanto uma condição de parada envolvendo a falha não fosse encontrada; no segundo modo de operação, denominado *backward mode*, a computação começa a partir da configuração correspondente à solução do problema e então constrói uma seqüência de passos que levam ao estado inicial. Novamente, em um sistema de correlação de eventos, as regras poderiam ser aplicadas sobre a base de dados que contém todas as possíveis falhas, até que uma condição de parada, na qual todos os eventos recebidos deveriam estar presentes, fosse encontrada. O mesmo conjunto de regras pode ser usado para os dois modos de operação (MEIRA, 1997).

Em comparação com os programas “tradicionais” em cujo código, tanto há o conhecimento especializado quanto as informações de controle, o que contribui para torná-los extremamente complexos e difíceis de manter, um sistema de raciocínio baseado em regras é mais simples, mais modularizado e mais fácil de manter; para tanto, ele é organizado em três níveis ilustrados na Figura 3.1 (CRONK, 1998):

- Um mecanismo de inferência que contém as estratégias para resolver uma determinada classe de problemas;
- Uma base de conhecimento contendo o conjunto de regras com o conhecimento sobre a tarefa específica;
- Uma memória operacional com os dados sobre o problema a ser resolvido.

Como exemplo da utilização do raciocínio baseado em regras, deve-se supor que dois eventos de falha de conexão envolvendo o mesmo objeto gerenciado indicam uma falha nesse objeto. Isto pode ser representado pela seguinte regra:

conexão_falhou(X,Y) e conexão_falhou(X, Z) ⇒ objeto_gerenciado_falhou(X)

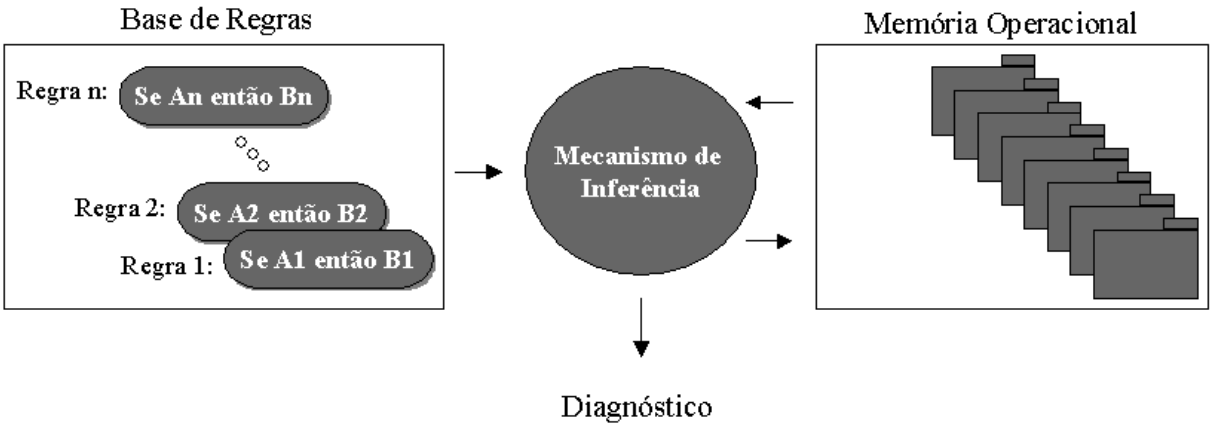

Figura 3.1: Componentes principais de um sistema de raciocínio baseado em regras

Aqui X e Y são variáveis que representam os nomes de dois objetos gerenciados. Um sistema de raciocínio baseado em regras operando no *forward mode* poderia varrer a memória operacional constantemente procurando por fatos de falhas de conexão. Suponha que os seguintes fatos foram adicionados a memória operacional:

conexão_falhou(a, b)

conexão_falhou(a, c)

Neste caso, o sistema poderia deduzir o seguinte fato:

objeto_gerenciado_falhou(a)

Já se o sistema estivesse operando no *backward mode,* o operador poderia requisitar que o sistema determinasse se o objeto gerenciado z falhou. Neste caso, o sistema deveria pesquisar a memória operacional por entradas do tipo conexão_falhou(z, Z), onde Z é um objeto gerenciado arbitrário.

Apesar das vantagens presentes em um sistema de raciocínio baseado em regras frente aos programas "tradicionais", existem também algumas limitações dessa abordagem no tocante a aquisição do conhecimento, que, a princípio, é baseada em entrevistas com especialistas humanos. Esse procedimento pode consumir bastante tempo e é suscetível a erros. Esta limitação tem levado pesquisadores a tentar automatizar e tornar mais rápido este processo através da utilização de técnicas de aprendizagem de máquina.

Uma outra limitação desses sistemas é o fato de que eles não levam em conta em seu processo dedutivo as experiências anteriores, ou seja, não têm memória. Portanto, um sistema puramente baseado em regras, o qual disparou centenas de regras para deduzir a ocorrência de uma falha a partir de um conjunto de eventos irá disparar novamente todas essas regras todas as vezes que receber o mesmo conjunto de eventos, chegando novamente à mesma conclusão. O fato de não levar em conta as experiências anteriores contribui para a degradação da performance desses sistemas.

Mas a principal limitação do raciocínio baseado em regras é o fato de que todo o conhecimento do sistema está limitado às suas regras, portanto ele não pode lidar com situações para as quais as regras não se aplicam; assim, em redes que sofrem modificações constantes de topologia, como é o caso da rede elétrica da CHESF, um sistema baseado em regras tende a se tornar obsoleto rapidamente caso não haja uma constante atualização da base de regras.

3.2.2 Raciocínio Baseado em Casos

Como alternativa para o raciocínio baseado em regras, alguns autores propuseram uma técnica denominada raciocínio baseado em casos. Aqui a unidade básica de conhecimento é um caso, e não, uma regra. Os casos são registros de uma base de casos contendo os aspectos mais relevantes de episódios do passado que são armazenados, recuperados, adaptados e utilizados na solução de novos problemas. A experiência obtida com a solução desses novos problemas constitui novos casos, que são adicionados à base de casos para uso futuro. Portanto, o sistema é capaz de obter conhecimento sozinho, sem necessitar da intervenção de especialistas humanos. A Figura 3.2 apresenta os principais componentes de um sistema de raciocínio baseado em casos.

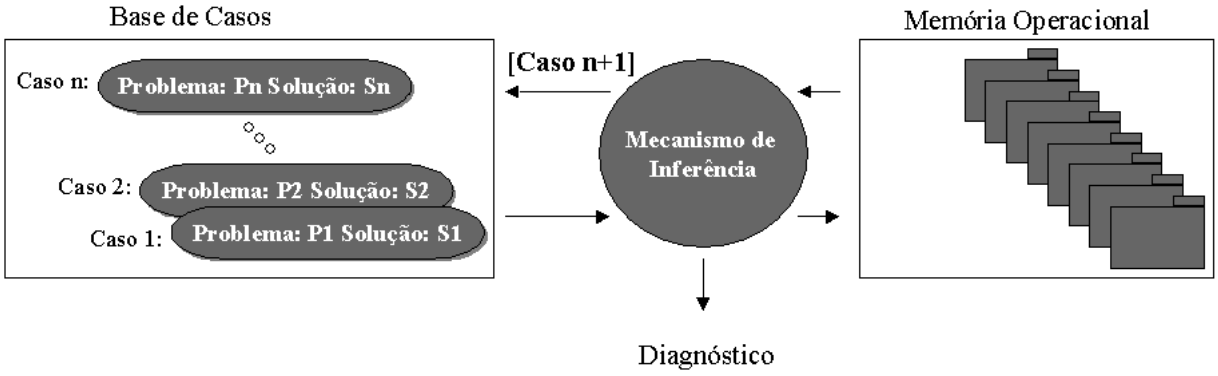

Figura 3.2: Componentes principais de um sistema de raciocínio baseado em casos

Uma outra característica dos sistemas de raciocínio baseado em casos é a sua habilidade de modificar seu comportamento futuro de acordo como os erros cometidos. Além disso, um sistema de raciocínio baseado em casos pode construir soluções para problemas novos, através da adaptação de casos passados para a nova situação.

O desenvolvimento de sistemas de raciocínio baseado em casos começou nos anos 80. Desde essa época, muitos desafios têm estimulado a criatividade dos pesquisadores: Como representar os casos? Como indexá-los para serem recuperados quando necessário? Como adaptar um caso antigo para uma nova situação de forma a obter uma solução original? Como testar uma solução proposta e classificá-la como um sucesso ou uma falha?

É essencial que sistemas de raciocínio baseado em casos detenham uma métrica de similaridade para que casos úteis possam ser recuperados da base de casos. Não é desejável que o sistema recupere o caso que tem o maior número de campos coincidentes com o novo caso, já que alguns dos campos podem ser irrelevantes. Por isso, são necessários critérios de relevância para indicar que tipos de informação devem ser considerados em um dado problema (LEWIS, 1999). É importante notar que critério de relevância não é a mesma coisa que uma regra do raciocínio baseado em regras. Critérios de relevância simplesmente dizem ao sistema quais casos ele deve olhar, mas não diz o que ele deve fazer com os casos.

Então, como estabelecer os critérios de relevância? As pesquisas atuais envolvem a aplicação de técnicas de aprendizagem de máquina em uma base de casos existente. Atualmente, no entanto, uma abordagem mais pragmática seria definir e testar manualmente os critérios de relevância.

Outro desafio do raciocínio baseado em casos é desenvolver as técnicas de adaptação pelas quais o sistema pode modificar uma solução antiga para resolver um novo problema. (LEWIS, 1993-2) descreve uma técnica chamada **adaptação parametrizada**, que é baseada na existência, em um *trouble ticket*[4], de um certo relacionamento entre as variáveis que descrevem um problema e as variáveis que especificam a solução correspondente. O sistema de raciocínio baseado em casos pode levar em conta os parâmetros dessa relação na proposição de uma solução para o novo caso em análise.

[4] *Trouble tickets* são fichas de ocorrências onde são anotados os problemas, suas causas, suas conseqüências e a forma como os problemas foram resolvidos.

Um exemplo do raciocínio baseado em casos é encontrado em (LEWIS, 1993-2), supondo-se que o caso em questão envolva baixo desempenho na transmissão de arquivos em uma rede de computadores:

Problema: Desempenho da transferência de arquivos: 5 unidades

Dois casos similares foram recuperados da base de dados:

- **Problema**: Desempenho da transferência de arquivos: 3 unidades;

  Solução: Ajuste da carga da rede para 20.

- **Problema**: Desempenho da transferência de arquivos: 7 unidades;

  Solução: Ajuste da carga da rede para 30.

Os casos recuperados podem ser aplicados para o novo caso através da adaptação parametrizada. Assumindo que o ajuste da carga da rede é uma função do desempenho da transferência de arquivos, chamada F. Então F(3) = 20 e F(7) = 30, e o que se deseja é descobrir o valor de F(5). Um método para determinar esse valor é o da interpolação simples: F(5) = 25. Então a carga da rede deveria ser ajustada para 25. Se este ajuste resolver o problema, então um novo caso pode ser adicionado à base de casos:

Problema: Desempenho da transferência de arquivos: 5 unidades;

Solução: Ajuste da carga da rede para 25.

3.2.3 Raciocínio Baseado em Modelos

O raciocínio baseado em modelos é um paradigma da inteligência artificial que tem muitas aplicações na correlação de eventos. Consiste em representar o sistema através de um modelo estrutural e de um modelo funcional, em contraste com a abordagem baseada em regras, as quais representam associações empíricas. No caso dos sistemas de gerência de redes, o modelo estrutural poderia conter uma descrição dos elementos da rede e da topologia. O modelo funcional descreve os processos de propagação e correlação de eventos (MEIRA, 1997).

Uma aplicação do raciocínio baseado em modelos é rastrear o estado da rede. À medida que mensagens são observadas o modelo é usado para atualizar a estimativa do estado corrente da rede. Quando um evento ocorre, as informações sobre o estado da rede podem ser utilizadas para ajudar a encontrar a falha.

Supondo que exista uma subestação de uma rede elétrica com uma linha de transmissão ligada ao restante da rede através de 4 disjuntores, conforme ilustrado na Figura 3.3.

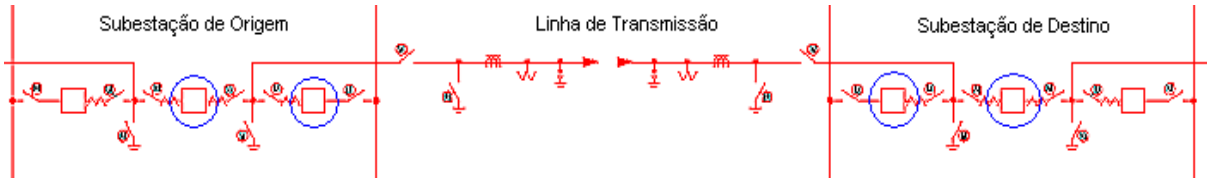

Figura 3.3: Ilustração de uma linha de transmissão

Um modelo da rede elétrica possuirá entidades para representar cada um destes quatro disjuntores e a linha de transmissão.

Cada entidade do modelo deve poder se comunicar de alguma forma com o objeto gerenciado por ela representado; dessa forma, as entidades podem saber qual o estado real dos objetos gerenciados na rede.

Usando este modelo, é possível realizar o diagnóstico de um problema de desconexão da linha de transmissão através da interação entre as entidades que representam os disjuntores e a entidade que representa a linha de transmissão, a qual pode se comunicar com as entidades que representam seus disjuntores para descobrir se ela ainda está conectada ao restante da rede elétrica; caso todas respondam que não, então é diagnosticado o problema de desconexão total da linha de transmissão; se alguns respondem que não e outros que sim, é diagnosticado o problema de desconexão parcial da linha de transmissão e se todos respondem que sim, não existe problema algum com as conexões da linha de transmissão.

Este exemplo mostra que a correlação de eventos no raciocínio baseado em modelos é um esforço colaborativo entre modelos virtuais inteligentes, que são representações em software das entidades reais da rede.

Uma descrição informal de um sistema de raciocínio baseado em modelos é dada a seguir (LEWIS, 1999):

- Um sistema de raciocínio baseado em modelos representa cada componente da rede como um modelo;
- Um modelo é uma representação de uma entidade física ou de uma entidade lógica da rede;

- Um modelo que representa uma entidade física está em comunicação direta com esta entidade.

3.2.4 Codebooks

A idéia da abordagem por *codebooks* é simples: cada problema apresenta seus sintomas por meio de vários eventos. Esses eventos podem ser eventos locais do objeto gerenciado de onde o problema se originou e eventos propagados para outros objetos relacionados. O conjunto de eventos causados por um problema é tratado com um **código** que identifica o problema. A correlação é simplesmente o processo de decodificar o conjunto de eventos observados determinando que problema os tem como código (YEMINI, 1996).

Nessa abordagem, a maioria do processamento necessário para a correlação de eventos é feita a priori, originando um banco de dados denominado *codebook* (KLIGER, 1995). O *codebook* pode ser visto como uma matriz, onde cada linha corresponde a um sintoma (evento), e cada coluna corresponde a um problema. Se *n* sintomas distintos são apresentados no *codebook*, cada elemento do vetor $p_i=(s_1, s_2,..., s_n)$ contém uma medição da causalidade do problema p_i, para o sintoma correspondente. Portanto, se no vetor p_1, $s_1=0$, o sintoma s_1 nunca irá ocorrer como conseqüência do problema p_1; por outro lado, se $s_1=1$, o sintoma s_1 sempre ocorre como conseqüência do problema p_1 (YEMINI, 1996).

Considere um pequeno domínio de interesse no qual existem quatro tipos de eventos, E_1, E_2, E_3 e E_4; e dois tipos de problemas, P_1 e P_2. Agora se são conhecidos os conjuntos de eventos que causam cada alarme, essa informação pode ser organizada em uma matriz como a mostrada na Figura 3.4.

	P_1	P_2
E_1	1	1
E_2	0	0
E_3	0	1
E_4	0	0

a) Matriz de correlação

	P_1	P_2
E_1	1	1
E_3	0	1

b) *Codebook*

Figura 3.4: Uma matriz de correlação e um codebook equivalente

A matriz de correlação indica que uma ocorrência do evento E_1 caracteriza o problema P_1 e que uma ocorrência conjunta dos eventos E_1 e E_3 caracteriza o problema P_2.

A codificação transforma a matriz em uma matriz comprimida, chamada *codebook*. Devido à redundância de sintomas, a matriz acima pode ser comprimida já que os eventos (sintomas) E_1 e E_3 são suficientes para distinguir entre P_1 e P_2. É possível observar que cada problema tem uma **assinatura única de eventos,** a qual é o código do problema e consiste

dos sintomas que o problema causa. É importante que haja uma única combinação de sintomas para cada problema de forma a diferenciar todos os problemas inequivocamente. De posse desta informação, o algoritmo de correlação é simples: examinam-se os sintomas e identifica-se o problema com assinatura *mais próxima* dos sintomas. A proximidade é medida pela distância Hamming entre códigos (o número de bits diferentes).

Como exemplo adicional, considere a matriz e os *codebooks* da Figura 3.5, que mostra que pode existir mais de um *codebook* para uma determinada matriz de correlação. Na construção de um *codebook*, a única restrição é que haja unicidade de assinatura para cada problema. Porém, um código pode ser mais resiliente a ruídos do que outro. Definindo o **raio** do código como sendo metade da distância Hamming mínima entre quaisquer dois códigos no *codebook*, pode-se ver que, com código de raio *r*, é possível detectar *r* erros e corrigir *r-1* erros, onde "erro" significa um evento não detectado ou gerado espuriamente.

Como a maior parte dos passos, até a geração do *codebook* é realizada *off-line*; apenas as operações mais simples são realizadas em tempo de execução (o mecanismo de inferência basicamente casa sintomas a assinaturas). Isto permite que a abordagem baseada em *codebooks*, em termo de eventos processados por segundo, tenha uma performance com ordem de magnitude de duas a quatro vezes maior que as outras abordagens encontradas na literatura (YEMINI, 1996).

	P_1	P_2	P_3	P_4	P_5	P_6
E_1	1	0	0	1	0	1
E_2	1	1	1	1	0	0
E_3	1	1	0	1	0	0
E_4	1	0	1	0	1	0
E_5	1	0	1	1	1	0
E_6	1	1	1	0	0	1
E_7	1	0	1	0	0	0
E_8	1	0	0	1	1	1
E_9	0	1	0	0	1	1
E_{10}	0	1	1	1	0	0
E_{11}	0	0	0	1	1	0
E_{12}	0	1	0	1	0	0
E_{13}	0	1	0	1	1	1
E_{14}	0	0	0	0	0	1
E_{15}	0	0	1	0	1	1
E_{16}	0	1	1	0	0	1
E_{17}	0	1	0	1	1	0
E_{18}	0	1	1	1	0	0
E_{19}	0	1	1	0	1	0
E_{20}	0	0	0	0	1	1

a) Matriz de Correlação

	P_1	P_2	P_3	P_4	P_5	P_6
E_1	1	0	0	1	0	1
E_2	1	1	1	1	0	0
E_4	1	0	1	0	1	0

b) *Codebook* de raio 5

	P_1	P_2	P_3	P_4	P_5	P_6
E_1	1	0	0	1	0	1
E_3	1	1	0	1	0	0
E_4	1	0	1	0	1	0
E_6	1	1	1	0	0	1
E_9	0	1	0	0	1	1
E_{18}	0	1	1	1	0	0

c) *Codebook* de raio 1.5

Figura 3.5: Dois codebooks para a mesma matriz

Os pontos fortes da abordagem baseada em *codebooks* são: performance, robustez (já que eventos podem ser perdidos ou eventos espúrios podem existir), computação automática das regras de correlação e versatilidade na adaptação do sistema a mudanças ocorridas na topologia da rede (YEMINI, 1996).

Apesar de ser uma ótima alternativa do ponto de vista da performance e da robustez, a abordagem baseada em *codebooks* demanda, de acordo com alguns autores, um grande esforço na modelagem da rede, fazendo assim com que ela seja pouco recomendada para redes complexas (MEIRA, 1997). Outra desvantagem dessa abordagem, para sua utilização prática, é o fato de ser protegida por patentes pertencentes a SMARTS (*System Management Arts*) (YEMINI).

3.2.5 Redes Neurais Artificiais

Uma rede neural artificial é um sistema constituído por elementos (neurônios) interconectados de acordo com um modelo que tenta reproduzir a rede neural existente no cérebro humano. Conceitualmente, cada neurônio pode ser considerado como uma unidade autônoma de processamento, possuindo memória local e canais unidirecionais de comunicação com outros neurônios. O funcionamento de um canal de entrada em uma rede neural artificial é inspirado na operação de um dendrito nos neurônios biológicos (Figura 3.6). De maneira análoga, um canal de saída tem o axônio como modelo. Um neurônio possui um axônio, mas pode ter um número arbitrário de dendritos (em um neurônio biológico existem cerca de dez mil dendritos). O sinal de saída de um neurônio pode ser utilizado como entrada para um numero arbitrário de outros neurônios (MEIRA, 1997).

Na forma mais simples, o processamento realizado em um neurônio consiste em efetuar a soma ponderada dos sinais presentes nas suas entradas e gerar um sinal de saída se o resultado da soma ultrapassar um limiar predefinido. No caso mais geral, o processamento pode incluir qualquer tipo de operação matemática nos sinais de entrada, inclusive levando em consideração os valores armazenados na memória local do neurônio.

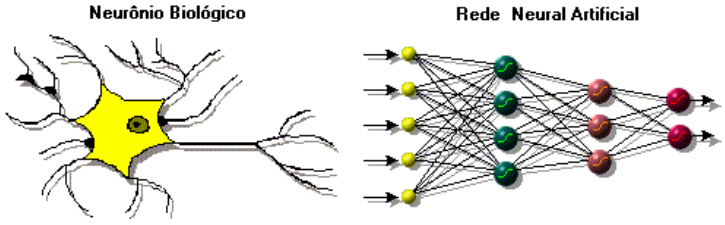

Figura 3.6: Neurônio biológico e rede neural artificial

Uma das motivações do desenvolvimento das redes neurais artificiais é a utilização de computadores para lidar com classes de problemas que são facilmente resolvidos pelo cérebro humano, mas que não são fáceis de tratar em computadores utilizando os paradigmas convencionais de programação.

Uma rede neural artificial precisa ser treinada. Para este propósito, uma série de padrões precisa ser construída; cada padrão, no caso da correlação de eventos, consiste dos eventos gerados por uma falha. Existe um certo número de eventos que podem ser gerados devido a uma falha na rede, os quais são marcados com 1s e os outros com 0s. Isto significa que cada falha é representada com uma seqüência de 0s e 1s. Cada falha possui o próprio vetor, também chamado de padrão (BIELER, 1994).

Na prática, é difícil haver eventos suficientes sobre uma falha porque os sistemas de monitoração não armazenam todos. Outro problema é a disponibilidade de casos da falha. Não é possível ter todos os casos de falha registrados em um sistema de monitoração. Estes problemas podem ser contornados com a utilização de um simulador.

Devido à frustração da grande expectativa sobre as pesquisas de modelagem do sistema nervoso nos anos 40, o interesse nas redes neurais artificiais foi significativamente reduzido no final dos anos 60 quando estudos teóricos mostraram fortes limitações deste paradigma. O interesse na área renasceu no começo dos anos 80 quando a performance dos computadores começou a permitir implementações práticas, que têm um alto custo computacional. A descoberta de novas aplicações das redes neurais artificiais também contribui para esta renascença.

O controle distribuído, o armazenamento de dados e o paralelismo são características das redes neurais artificiais. Além disso, não precisam ter um conhecimento prévio das relações matemáticas entre as entradas e saídas, as quais podem ser “aprendidas” automaticamente durante a operação normal do sistema. Isto as torna, à primeira vista, uma boa alternativa para aplicações como correlação de eventos e diagnóstico de falhas onde as relações entre eventos e falhas nem sempre são bem definidas e entendidas e onde os dados disponíveis são, muitas vezes, ambíguos e inconsistentes.

3.2.6 Reconhecimento de Crônicas

Uma crônica é um conjunto de eventos interligados por restrições temporais que representa as possíveis evoluções de um sistema observado (CORDIER, 2000) (MAGDA, 1999). Um conjunto de crônicas forma uma base de crônicas.

Uma crônica pode ser representada por um grafo dirigido, em que os nodos são eventos, e os arcos dirigidos indicam restrições temporais interligando os eventos (Figura 3.7).

Para ilustrar, considere o seguinte cenário de evolução dos eventos na ocorrência de um problema de fase/terra em uma linha de transmissão:

- Disjuntor desarma (**e1**);
- Disjuntor volta a ser armado pelo sistema de auto-proteção (**e2**). Esse evento pode levar até três segundos;
- Depois disso, no intervalo entre um e dois segundos, três componentes efetuam medição e enviam mensagem (**e3**, **e4**, **e5**).

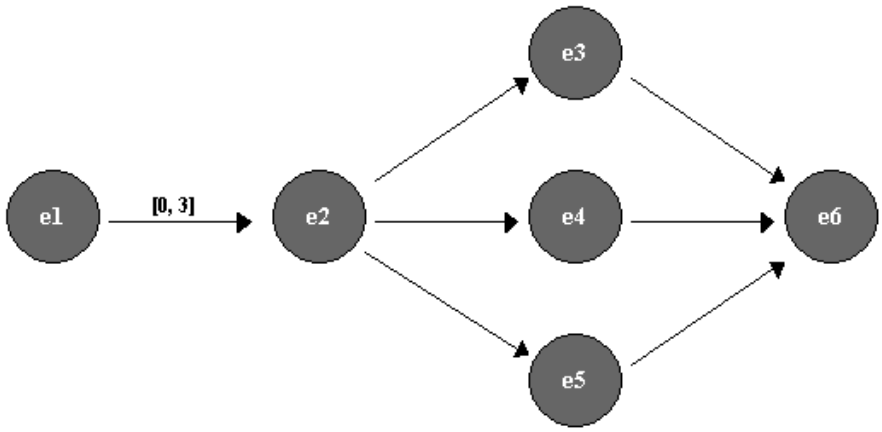

Figura 3.7: Uma crônica

O grafo da Figura 3.7 mostra a dependência temporal do evento **e2** em relação ao evento **e1**. O tempo é representado por um conjunto de pontos linearmente ordenado. Uma restrição temporal entre dois pontos de tempo t_1 e t_2 é indicada pelo intervalo $[I^-,I^+]$, que corresponde aos limites mínimos e máximos da distância temporal entre t_1 e t_2. Eventos ocorrem em pontos do tempo, sendo, portanto, instantâneos.

Existem duas formas de representação de crônicas: a primeira utiliza máquinas de estados finitos. Esta forma de representação tem sido mais adequada para modelar situações

em que a ordem dos eventos é total. Entretanto, é muito comum em casos reais que a ordem dos eventos seja parcial, limitando bastante o emprego da técnica.

Uma segunda forma de representação, bem menos restritiva, utiliza uma linguagem de representação CRS — "Chronicle Recognition System" —, desenvolvida no Centro de Pesquisa da *France Télécom* (DOUSSON, 1993) (GHALLAB, 1996).

Uma crônica em CRS é composta por quatro partes:

- Um conjunto de eventos que representa as mudanças relevantes no estado de um sistema;
- Um conjunto de assertivas que define o contexto da ocorrência dos eventos;
- Um conjunto de restrições temporais que relacionam os eventos, determinando o tempo mínimo e máximo entre a ocorrência de dois eventos;
- Um conjunto de ações a ser processado com o reconhecimento de uma crônica.

O reconhecimento de crônicas é um formalismo adequado para representar seqüências de eventos onde o fator *tempo* é um ponto fundamental.

A geração e a manutenção de bases de crônicas são indubitavelmente os maiores problemas dessa técnica. As soluções existentes são semi-automáticas, exigindo a participação efetiva de especialistas nos diversos domínios de aplicação, o que as torna bastante onerosas.

3.2.7 Lógica Nebulosa

Devido à complexidade das redes gerenciadas, nem sempre é possível construir um modelo preciso dessas redes, em que todas as situações nas quais a ocorrência de um dado conjunto de eventos indicam uma falha em um determinado objeto gerenciado são evidenciadas.

O conhecimento das relações de causa e efeito entre eventos e falhas é geralmente incompleto. Além disso, alguns dos eventos gerados por falhas freqüentemente estão indisponíveis para o sistema de correlação de eventos, devido a perdas ou atrasos. Finalmente, devido ao fato de que existem mudanças freqüentes na topologia da rede, quanto mais preciso o modelo, mas rápido ele fica desatualizado.

A imprecisão das informações fornecidas pelos especialistas freqüentemente causa grandes dificuldades. A lógica nebulosa é uma alternativa para lidar com incertezas e imprecisões que caracterizam algumas aplicações da gerência de redes.

O conceito básico por trás da lógica nebulosa é o conceito de conjuntos nebulosos. Na lógica clássica, um conjunto A possui a propriedade de que, para um dado elemento X, a expressão, "X está contido em A", sempre assume um de dois possíveis valores, verdadeiro ou falso. Nos conjuntos nebulosos, cada elemento X possui, em relação ao conjunto, um grau de associação, que pode assumir qualquer valor entre 0 (o elemento definitivamente não está contido no conjunto) e 1 (o elemento definitivamente está contido no conjunto) (MEIRA, 1997).

3.3 Ferramentas de Correlação de Eventos

Esta seção apresenta algumas das principais ferramentas comerciais de correlação de eventos encontradas na literatura.

Um grande número de aplicações de inteligência artificial, mais precisamente, sistemas especialistas, foi desenvolvido com o objetivo de auxiliar na gerência de redes através da correlação de eventos, no entanto, apenas alguns poucos são capazes de realizar a correlação de eventos em tempo real.

Esta lista de aplicações não pretende ser exaustiva, mas representativa da aplicação das técnicas discutidas na seção 3.2.

3.3.1 Event Correlation Services (HP)

A HP (Hewlett-Packard) desenvolveu, como parte da sua plataforma **OpenView,** um produto denominado ECS ("Evento Correlation Services") (HP, 1995), que se propõe a processar centenas de eventos por segundo, com o objetivo de lidar como o fenômeno conhecido por **tempestade de eventos**[5] em redes de computadores e redes de telecomunicação.

O ECS é um sistema de raciocínio baseado em regras, as quais são agrupadas em elementos de processamento denominados **nós**, cada nó é responsável por uma certa funcionalidade durante o processo de correlação.

---

[5] Ocasião em que centenas ou milhares de eventos são gerados em um curto período de tempo em conseqüência de uma única falha.

Os nós são interconectados por “circuitos”, através dos quais os eventos fluem. Um nó composto pode ser definido a partir de um segmento de um circuito e adicionado à biblioteca de nós para contemplar uma funcionalidade definida pelo usuário. Sua arquitetura distribuída permite uma diminuição do tráfego de informações de gerência através da distribuição da correlação de eventos.

3.3.2 NerveCenter Pro

O NerveCenter Pro da Seagate (SEAGATE, 1996) utiliza o raciocínio baseado em modelos para realizar a correlação de eventos em redes de computadores. As informações recebidas são continuamente monitoradas para determinar a ocorrências de situações anormais, definidas previamente. Se uma dessas situações ocorre a entidade correspondente do modelo muda de estado e realiza as notificações necessárias. O modelo é atualizado a cada mensagem recebida até que a situação seja resolvida.

3.3.3 InCharge

O InCharge é um sistema completamente automático para o diagnóstico de falhas em redes de telecomunicação e redes de computadores. O sistema foi desenvolvido pela companhia norte-americana SMARTS e utiliza a correlação por *codebooks* por ela patenteada (KLIGER, 1995).

O processo de identificação dos problemas é dividido em duas partes; no primeiro estágio, realizado no ambiente de desenvolvimento, os códigos dos problemas são gerados; no segundo estágio, em tempo de execução, os problemas são decodificados.

A Motorola utilizou o InCharge no projeto Iridium, que foi um sistema de telecomunicação baseado em uma constelação de satélites de órbita baixa. Inicialmente eles utilizaram o NerveCenter da Seagate mas, depois de três anos, mudaram para o InCharge devido a defeitos e outras deficiências do NerveCenter.

3.3.4 CRITTER

O CRITTER (LEWIS, 1993) é um sistema de *trouble ticket* que utiliza o raciocínio baseado em casos para o diagnóstico de falhas em redes de computadores. Aqui, cada *trouble ticket* fechado constitui um caso. O componente de raciocínio baseado em casos do CRITTER provê mecanismos para recuperar um caso útil, adaptá-lo e adicionar um novo caso a base de casos.

Para recuperar um caso útil, é utilizado um conjunto de informações sobre as classes de problemas e um conjunto de atributos armazenados no *trouble ticket*. Pode acontecer de a solução para o caso recuperado ser diretamente aplicável ao problema em questão. No caso mais geral, para resolver o problema, o sistema precisa adaptar o caso recuperado.

3.3.5 IMPACT

IMPACT (*Intelligent Management Plataforms for Alarm Correlation Tasks*) (JAKOBSON, 1993) é um sistema desenvolvido pela companhia americana Verizon e que utiliza o raciocínio baseado em modelos em redes de telecomunicação celular. Ele contém módulo que permite que os modelos possam ser criados pelos próprios usuários do sistema permitindo assim que a modelagem do conhecimento possa ser feita pelos próprios especialistas.

3.3.6 SCOUT

SCOUT é um sistema desenvolvido pelo *AT&T Bell Laboratories* para automatizar o diagnóstico de problemas de transmissão em redes de telecomunicação (SASISEKHARAN, 1994). Um dos objetivos do produto é detectar e prever a ocorrência de problemas crônicos nos sistemas de transmissão, através do uso de técnicas de aprendizado de máquina (como as redes neurais artificiais na seção 3.2.5), permitindo a manutenção pró-ativa do sistema.

3.3.7 SPARSE

SPARSE é um sistema de raciocínio baseado em regras, cujo principal objetivo é auxiliar os operadores dos centros de controle de subestações de energia elétrica a interpretar rapidamente a enorme quantidade de eventos que os sistemas SCADA podem receber durante incidentes (por exemplo, um curto circuito em uma linha de transmissão),e, assim, minimizar o tempo de interrupção de serviço para o clientes (VALE, 1997). Esse sistema foi utilizado pela subsidiária de energia elétrica de Portugal.

3.3.8 SISPRO (CEPEL)

O SISPRO é um sistema de tratamento de eventos que está sendo desenvolvido pelo CEPEL, em parceria com a COSERN (Companhia Energética do Rio Grande do Norte). O produto principal do projeto será o desenvolvimento de um sistema inteligente de tratamento de alarmes integrado ao SAGE, nas instalações do COS (Centro de Operação do Sistema) da COSERN, em Natal.

Este produto poderá ser utilizado tanto para o auxílio ao operador durante uma falta severa no sistema elétrico, quando é mais difícil de se obter um diagnóstico com rapidez, dada a grande quantidade de eventos gerados, como para treinamento de novos operadores do sistema elétrico. O sistema utiliza o raciocínio baseado em regras para diagnosticar falhas na rede elétrica e apresenta uma alteração em relação ao raciocínio baseado em regras convencional que é o de detectar regras parciais, apresentado a porcentagem de casamento com uma regra.

3.4 Considerações Finais

Este capítulo forneceu os conhecimentos básicos associados à correlação de eventos, em seguida, apresentou as principais técnicas de correlação de eventos encontradas na literatura e finalizou com uma visão panorâmica sobre algumas ferramentas de correlação de eventos existentes na literatura.

Devido à importância do assunto, muito ainda precisa ser desenvolvido na área de correlação de eventos para atender a todas as necessidades da gerência de redes, principalmente de redes elétricas.

Apesar de todas as técnicas apresentadas neste capítulo poderem ser utilizadas para realizar correlação de eventos em qualquer tipo de rede, seja de computadores, de telecomunicação ou elétrica, na prática, isso não acontece. Existem grupos de pesquisadores de técnicas de correlação de eventos para cada tipo de rede, mas há técnicas que só são estudadas para um determinado tipo de rede mesmo sem existir nada que impeça sua aplicação aos demais tipos. Isso se agrava ainda mais para as redes elétricas já que a maioria dos estudos em correlação de eventos é voltada para as redes de computadores e de telecomunicações, sendo essa uma das causas para a escassez de ferramentas especificas para redes elétricas.

Nos próximos capítulos, apresenta-se uma nova ferramenta para o tratamento de eventos em redes elétricas, a qual utiliza uma combinação de duas das técnicas de correlação de eventos apresentadas neste capítulo.

4 Uma Ferramenta de Tratamento de Eventos em Redes Elétricas: Requisitos, Técnica de Correlação e Projeto

Este capítulo aborda uma nova ferramenta de tratamento automático de eventos em redes elétricas, chamada *Smart Alarms,* que está sendo desenvolvida para ser utilizada no diagnóstico de alarmes na rede elétrica da CHESF. O *Smart Alarms* é capaz de processar, em tempo real, o fluxo de eventos do SAGE e realizar o diagnóstico de alarmes na rede elétrica. Além de auxiliar os operadores nos centros de operação da CHESF, o *Smart Alarms* é uma poderosa ferramenta de treinamento para novos operadores. Com ele, é possível simular situações críticas nos centros de controle através da repetição dos eventos recebidos pelo SAGE, nos momentos em que ocorreram falhas na rede elétrica, e construir cenários para simular falhas que não ocorrem com freqüência.

Este capítulo traz, ainda, os requisitos do sistema (seção 4.1), a técnica de correlação de eventos utilizada para reduzir a quantidade de alarmes (seção 4.2), o projeto arquitetural da ferramenta (seção 4.3) e o projeto detalhado da mesma (seção 4.4), finalizando com algumas considerações finais sobre o assunto tratado (seção 4.5).

4.1 Levantamento de Requisitos

O levantamento de requisitos é um dos grandes desafios do projeto *Smart Alarms*. O problema é que o cliente não sabe exatamente o que o sistema precisa fazer para satisfazê-lo plenamente, por desconhecimento das possibilidades existentes.

A primeira versão funcional do *Smart Alarms,* desenvolvida durante esse trabalho de Mestrado e cujo processo de desenvolvimento será detalhado nesta dissertação, recebeu o nome de *Smart One* e tem as importantes funções de permitir o levantamento minucioso dos requisitos do *Smart Alarms*, verificar a aplicabilidade de técnicas de correlação de eventos a redes elétricas, testar desempenho das técnicas, validar as regras de diagnóstico e permitir conversas mais produtivas com o cliente.

Os requisitos básicos do *Smart One* foram levantados junto ao cliente, mas decisões especificas do projeto foram tomadas pela equipe de desenvolvimento devido à ausência do cliente e à natureza exploratória do protótipo.

A seguir, são descritos os requisitos funcionais e os requisitos não funcionais do *Smart One*.

4.1.1 Requisitos Funcionais

A lista de requisitos funcionais descreve as funcionalidades que devem estar disponíveis para o usuário nas próximas versões da ferramenta:

a) **Eventos SAGE**

- O sistema deve receber como entrada seqüências de eventos semelhantes às seqüências de eventos do SAGE;
- Deve ser possível escolher e alternar entre as diversas seqüências de eventos disponíveis na base histórica de dados, possibilitando a simulação da monitoração da rede elétrica realizada em diferentes datas.

b) **Velocidade de Simulação**

- O sistema deve simular em tempo real a geração de eventos do SAGE e permitir que a velocidade de simulação seja alterada. Alterando a velocidade da simulação, é possível realizar testes de forma mais eficiente, simulando, por exemplo, a monitoração de um dia inteiro em poucos segundos.

c) **Diagnóstico**

- O sistema deve reconhecer todas as regras de linhas de transmissão da CHESF e realizar o diagnóstico dos alarmes;
- Deve ser fácil inserir novas regras de correlação no programa. As regras de correlação podem estar embutidas no código, mas deve haver uma maneira de inserir novas regras sem que seja preciso realizar grandes modificações no programa;

- As informações topológicas devem estar fora do código do programa; elas devem ser lidas de arquivos do SAGE ou de arquivos criados especialmente para tal.

d) **Momento da Análise**

- Deve ser possível escolher em que momento a análise dos eventos precisa ser realizada. Basicamente. existem duas opções:
    - Janelas de amostragem: nesse caso, o sistema possui um *buffer* para armazenar os eventos recebidos e aguarda um intervalo de tempo pré-definido antes de realizar a análise. Dessa forma, é possível evitar que sejam realizados diagnósticos redundantes durante uma tempestade de eventos, pois o sistema aguardará por um momento de estabilidade na rede antes de realizar o diagnóstico.
    - Análise instantânea: a análise é realizada à medida que os eventos vão chegando e é bastante útil durante os testes da aplicação, pois permite observar, passo a passo, as mudanças que ocorrem na topologia da rede elétrica, devido à chegada dos eventos.

e) **Log**

- Deve ser possível definir pontos de *log* de informações em várias etapas do processo de tratamento de eventos. Por exemplo, *logar* todos os eventos gerados, *logar* todos os exibidos pelo SAGE e *logar* todos os alarmes exibidos pelo *Smart One*. Com isso se tem um histórico do processamento, o que permite uma localização mais fácil de defeitos de software.

f) **Subconjuntos de Eventos**

- Deve ser possível selecionar graficamente subconjuntos de uma seqüência de eventos para simulação, permitindo que o usuário possa simular apenas uma parte da monitoração realizada em uma determinada data, por exemplo, o intervalo de tempo em que ocorreram falhas graves na rede elétrica, e não, toda a monitoração realizada naquele dia.

g) **Interface padrão**

- O sistema deve apresentar uma saída não gráfica para o usuário. Essa interface permite que as informações geradas pelo sistema sejam exibidas

sem a necessidade da construção de uma interface gráfica. Com isso, é possível implementar grande parte do sistema antes de criar a interface gráfica.

h) **Interface gráfica**

- O sistema deve possuir uma interface gráfica, onde estarão disponíveis todas as opções de configuração, e que exiba o diagnóstico dos problemas.

i) **Conversor de Topologia**

- O SAGE armazena as informações topológicas em arquivos com um formato proprietário. Deve existir uma forma de converter as informações topológicas contidas nesses arquivos para um formato compatível com o protótipo.

j) **Navegador de Eventos**

- O protótipo possui um banco de dados temporal, onde são armazenadas as seqüências de eventos obtidas com a monitoração da rede elétrica. Deve ser possível localizar seqüências de eventos, utilizando-se uma ou mais das seguintes informações como chave:
    - Tipo de equipamento
    - Tipo de evento
    - Equipamento específico
    - Evento de um equipamento específico
    - Janela de tempo (todos os eventos entre quaisquer duas datas).

k) **Editor de Cenários**

- Deve existir um editor de cenários que permita modificar e criar seqüências de eventos para serem reproduzidas.

4.1.2 Requisitos Não Funcionais

Os seguintes requisitos não funcionais também devem ser atendidos pela ferramenta:

a) O diagnóstico de um problema deve ser realizado em, no máximo, 100 ms, para que a duração do processamento possa ser imperceptível ao operador;

b) As telas do sistema devem emular o mais fielmente possível as telas do SAGE, incluindo *layout*, cor e *timestamp* de eventos, para permitir um melhor conforto visual para os operadores do sistema, acostumados com as telas do SAGE;

c) O sistema deve exibir as informações que seriam exibidas pelo SAGE e as informações obtidas com o diagnóstico, permitindo uma observação fácil dos resultados do processamento.

4.2 Uma técnica híbrida de correlação de eventos

Um fator crucial para o sucesso de um sistema de correlação de eventos é a escolha de uma técnica de correlação de eventos que seja adequada ao ambiente no qual o sistema vai realizar seus diagnósticos. Um fator fundamental na escolha da técnica a ser utilizada no *Smart One* foi o tempo de implementação. Era preciso ter o sistema funcionando o mais rápido possível para poder começar a interagir com o cliente de forma produtiva na validação do sistema.

Como a CHESF já possuía uma base de conhecimento parcial na forma de regras de correlação, criada para um projeto anterior não completado, escolheu-se a técnica mais fácil de ser implementada de acordo com a situação: **o raciocínio baseado em regras**. Outra vantagem dessa escolha é o fato de o cliente já ter uma experiência anterior na confecção das regras, o que facilitaria a complementação da base de conhecimento. Porém essa escolha resultou em dois grandes problemas:

- **O grande número de regras necessárias para modelar os problemas da rede** - Para modelar apenas as falhas das linhas de transmissão supervisionadas pelo CROL (Centro Regional Leste), um dos cinco centros regionais da CHESF, são necessárias 1.334 regras. Supondo que cada um dos centros regionais tenha aproximadamente o mesmo número de equipamentos e que as regras de linhas de transmissão representem metade do total de regras de cada centro, pode-se estimar que seriam necessárias mais de 10.000 regras para modelar todos os problemas de todos os equipamentos da rede de transmissão da CHESF inteira.

- **As constantes modificações na topologia da rede da CHESF -** Somente no mês de outubro de 2002, foram realizadas oito alterações na topologia da rede. Usando o raciocínio baseado em regras convencional, seria necessário reescrever ou atualizar a base de regras sempre que ocorre uma modificação na topologia da rede. Com uma

base com 10.000 regras, essa tarefa se torna bastante árdua, impossibilitando, na verdade, que a técnica possa ser empregada.

Nesta seção, exibe-se uma técnica híbrida de correlação de eventos que une o raciocínio baseado em regras com o raciocínio baseado em modelos e que facilita muito a construção da base de conhecimento, além de praticamente eliminar a necessidade de atualização devido a mudanças na topologia da rede. A técnica híbrida consegue reduzir o número de regras necessárias para modelar os problemas na rede elétrica ao permitir que uma mesma regra possa ser reutilizada por vários equipamentos. Com isso foi possível diminuir o número de regras necessárias para modelar os problemas nas linhas de transmissão do CROL, de 1.334 para 51. Fazendo os mesmos cálculos de antes, é possível estimar que seriam necessárias aproximadamente 100 regras para modelar todos os problemas da rede elétrica da CHESF, já que as regras criadas para um centro regional podem ser utilizadas para os demais centros. A maior vantagem da técnica híbrida, no entanto, é evitar que seja necessário alterar a base de regras devido a alterações na topologia da rede. Com isso, o esforço para manter a base de regras atualizada é mínimo, consistindo apenas na inserção de novas regras nas ocasiões em que novos tipos de problemas são detectados na rede.

No restante desta seção, são apresentados os conceitos de primitivas topológicas e regras genéricas, sua utilização na técnica híbrida de correlação de eventos, e discutido o processo de obtenção das regras.

4.2.1 Primitivas Topológicas

Primitivas topológicas são construções conceituais que permitem retirar das regras de correlação de eventos todas as referências a elementos da topologia da rede elétrica. Com primitivas topológicas, é possível isolar, no diagnóstico de um problema, as informações referentes à topologia associada ao equipamento em questão; dessa forma, a regra não precisa ser alterada caso haja alguma modificação na topologia da rede.

As primitivas topológicas são baseadas na análise da conectividade de um grafo que representa todas as conexões entre os diferentes equipamentos presentes na rede elétrica. Este grafo é a contribuição do raciocínio baseado em modelos para a técnica híbrida. Além de modelar as conexões entre os equipamentos da rede, o grafo mantém também o estado de cada equipamento através do processamento dos eventos recebidos pelo sistema de diagnóstico. A Figura 4.1 ilustra o fluxo de informação do sistema; nela observa-se que a

abertura de um disjuntor (D1) em uma subestação monitorada gera um evento, o qual é coletado pelas estações remotas de monitoração e enviado para o SAGE; o *gateway*[6] repassa essa informação do SAGE para o *Smart One,* que, por sua vez, atualiza o grafo de topologia com a remoção das conexões em vermelho.

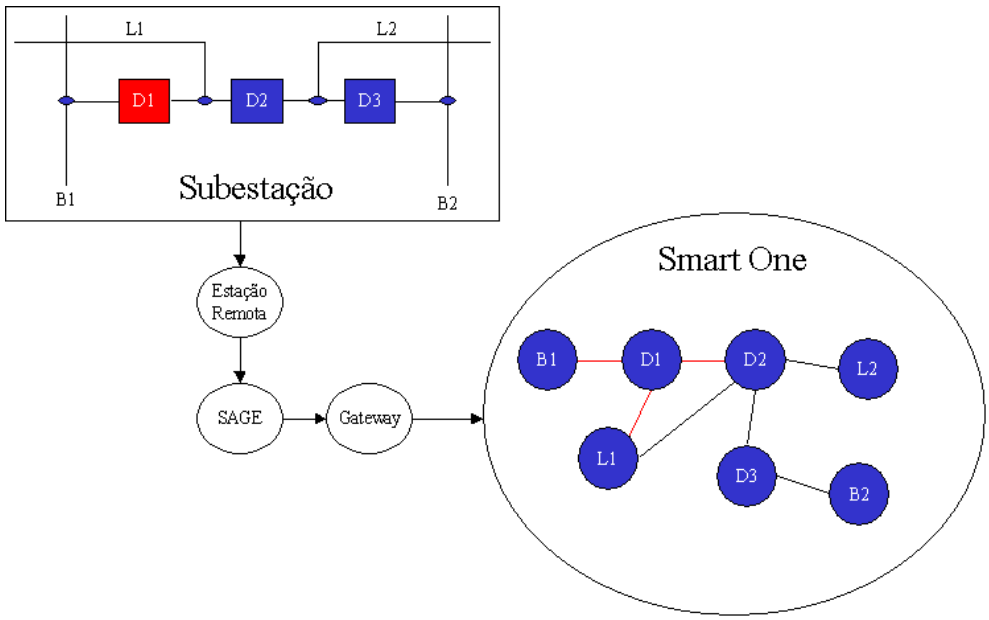

Figura 4.1: Fluxo de informação do sistema

Foram definidas as seguintes primitivas topológicas para as regras de linhas de transmissão:

- **Conectado (Linha, Barra)**: informa se uma determinada linha está conectada a uma determinada barra;
- **Desligamento parcial lado de(Linha)**: informa se a linha não está conectada a nenhuma das barras da subestação de origem, mas a alguma das barras da subestação de destino;
- **Desligamento parcial lado para(Linha)**: informa se a linha não está conectada a nenhuma das barras da subestação de destino, mas a alguma das barras da subestação de origem;
- **Desligamento total(Linha)**: a linha não está conectada a nenhuma barra.

[6] O Gateway e a arquitetura geral do sistema serão discutidos em detalhes mais adiante.

Como exemplo, considere o conhecimento em forma de regra que correlaciona os eventos gerados quando do “desligamento total da linha de transmissão 05L8-AGD/RCD (Angelim II / Recife II)”, exibido na Figura 4.2. A topologia da linha 05L8-AGD/RCD está ilustrada na Figura 4.3.

```
(DJ(15L8, Angelim II) OR CV(35L8-4, Angelim II) OR CV(35L8-5, Angelim II)) AND
(DJ(15D2, Angelim II) OR CV(35D2-1, Angelim II) OR CV(35D2-2, Angelim II)) AND
(DJ(15L8, Recife II) OR CV(35L8-4, Recife II) OR CV(35L8-5, Recife II))  AND
(DJ(15D1, Recife II) OR CV(35D1-1, Recife II) OR CV(35D1-2, Recife II)) AND
 NOT(ATPR(05L8,Angelim II)) AND NOT(ATPR(05L8,Recife II)) AND
 KV(05L8, Angelim II, 0) AND KV(05L8, Recife II, 0) AND
 MW(05L8, Angelim II, 0) AND MW(05L8, Recife II, 0) AND
 MVAR(05L8, Angelim II, 0) AND MVAR(05L8, Recife II, 0)
```

Figura 4.2: Regra para desligamento total da linha 05L8-Angelim II / Recife II

Na regra, o termo DJ significa ‘disjuntor aberto’, o termo CV é ‘chave aberta’, ATPR significa ‘atuação da proteção’, e KV, MW e MVAR referem-se a valores de tensão, potência ativa e potência reativa, respectivamente.

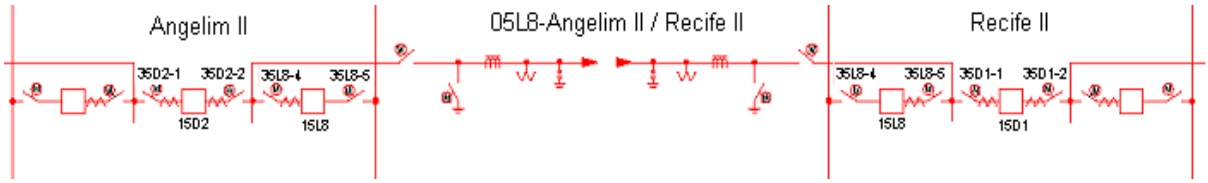

Figura 4.3: Linha 05L8-Angelim II / Recife II

A regra na Figura 4.2 indica que houve um desligamento total da linha de transmissão 05L8 entre Angelim II e Recife II sem a atuação de nenhuma das proteções da linha. Ela pode ser dividida em 6 partes:

- Desconexão da linha com a barra 1 de Angelim II

  (DJ(15L8, Angelim II) OR CV(35L8-4, Angelim II) OR CV(35L8-5, Angelim II));

- Desconexão da linha com a barra 2 de Angelim II

  (DJ(15D2, Angelim II) OR CV(35D2-1, Angelim II) OR CV(35D2-2, Angelim II));

- Desconexão da linha com a barra 1 de Recife II

  (DJ(15L8, Recife II) OR CV(35L8-4, Recife II) OR CV(35L8-5, Recife II));

- Desconexão da linha com a barra 2 de Recife II

  (DJ(15D1, Recife II) OR CV(35D1-1, Recife II) OR CV(35D1-2, Recife II));

- Verificação da não atuação das proteções da linha

  NOT(ATPR(05L8,Angelim II)) AND NOT(ATPR(05L8,Recife II));

- Verificação dos valores das grandezas elétricas da linha

KV(05L8, Angelim II, 0) AND KV(05L8, Recife II, 0) AND

MW(05L8, Angelim II, 0) AND MW(05L8, Recife II, 0) AND

MVAR(05L8, Angelim II, 0) AND MVAR(05L8, Recife II, 0).

Utilizando primitivas topológicas, é possível simplificar a regra da Figura 4.2, reduzindo-a para a regra na Figura 4.4.

```
DESLIGAMENTO_TOTAL(05L8-Angelim II/Recife II) AND
NOT(ATPR(05L8,Angelim II)) AND NOT(ATPR(05L8,Recife II)) AND
KV(05L8, Angelim II, 0) AND KV(05L8, Recife II, 0) AND
MW(05L8, Angelim II, 0) AND MW(05L8, Recife II, 0) AND
MVAR(05L8, Angelim II, 0) AND MVAR(05L8, Recife II, 0)
```

Figura 4.4: Regra com primitivas topológicas para desligamento total da linha 05L8-Angelim II / Recife II

É importante notar que a nova regra, escrita utilizando-se primitivas topológicas, só resolve um dos problemas do raciocínio baseado em regras: a dificuldade de manter a base de regras atualizada quando a rede sofre freqüentes alterações.

Apenas com a inserção de primitivas topológicas não é possível reutilizar uma regra previamente escrita para outros equipamentos do mesmo tipo. Para que as regras possam ser reutilizadas, elas precisam ser parametrizadas.

4.2.2 Regras Genéricas

Regras genéricas são as que, uma vez escritas para um determinado equipamento, podem ser reutilizadas para todos os equipamentos do mesmo tipo. Para tornar uma regra genérica, ela precisa ser parametrizada para remover todas as referências aos componentes específicos relacionados com o equipamento no qual está sendo realizado o diagnóstico. A parametrização da regra se dá através da criação de parâmetros topológicos que correspondem a cada um dos equipamentos da linha na qual está sendo realizado o diagnóstico. Para as linhas do CROL, foram criados os seguintes parâmetros topológicos:

- **DJ1**: corresponde ao disjuntor que interrompe a ligação entre a linha e o primeiro barramento ao qual a linha está conectada (disjuntor de linha);
- **DJ2**: corresponde ao disjuntor que interrompe a ligação entre a linha e o segundo barramento ao qual a linha está conectada, caso ele exista (disjuntor central);
- **DJR**: corresponde ao disjuntor do reator da linha, caso ele exista;

- **R**: corresponde ao reator da linha, caso ele exista;
- **B1**: Primeiro barramento ao qual a linha está conectada;
- **B2**: Segundo barramento ao qual a linha está conectada, caso ele exista.

Dessa forma, uma linha pode ser representada inequivocamente por seus parâmetros topológicos. Os parâmetros da linha 05L8- Angelim II / Recife II são descritos na Tabela 4.1:

Tabela 4.1: Parâmetros topológicos da linha 05L8-Angelim II / Recife II

	Lado de Angelim II	Lado De Recife II
DJ1	15L8-AGD	15L8-RCD
DJ2	15D2-AGD	15D1-RCD
DJR	Não existe	Não existe
R	Não existe	Não existe
B1	05B1-AGD	05B1-RCD
B2	05B2-AGD	05B2-RCD

Utilizando-se estes parâmetros topológicos, foi possível remover da regra qualquer referência a componentes específicos da linha de transmissão.

Apesar da grande facilidade de criação e do reuso de regras proporcionados pelos parâmetros topológicos, eles geram um trabalho adicional bastante grande, que é o de parametrizar as linhas. Toda vez que uma nova linha é inserida na topologia, ela precisa ser parametrizada para que o *Smart One* possa realizar corretamente os seus diagnósticos. Para resolver este problema, o professor Jacques Philippe Sauvé criou o *Topogiggio*, que é uma aplicação capaz de extrair os parâmetros topológicos das linhas de transmissão a partir dos arquivos de topologia do SAGE. Essa aplicação automatiza o processo de parametrização das novas linhas criadas.

Considere novamente a regra "desligamento total da linha de transmissão 05L8-AGD/RCD (Angelim II / Recife II)" na Figura 4.2. Introduzindo primitivas topológicas e parametrizando a regra, é possível reduzi-la para a regra da Figura 4.5.

```
DESLIGAMENTO_TOTAL(linha,de,para) AND
NOT(ATPR(linha, de)) AND NOT(ATPR(linha, para)) AND
KV(linha, de, 0) AND KV(linha, para, 0) AND
MW(linha, de, 0) AND MW(linha, para, 0) AND
MVAR(linha, de, 0) AND MVAR(linha, para, 0)
```

Figura 4.5: Regra genérica para desligamento total da linha 05L8-Angelim II / Recife II

Esta regra possui três parâmetros:

- **linha**: é o código da linha;

- **de**: é o código da subestação de origem da linha;
- **para**: é o código da subestação de destino da linha.

Note que, agora, diferentemente da regra gerada apenas com a introdução de primitivas topológicas, a regra não faz referência a nenhum equipamento específico, e, dessa forma, pode ser utilizada para diagnosticar um desligamento total em qualquer linha, não só na 05L8-AGD/RCD.

Para que essa regra seja genérica, basta que sejam adicionadas as condições de existência, as quais devem ser satisfeitas para que uma regra genérica possa ser aplicada a uma determinada linha de transmissão. Por exemplo, existem regras que fazem menção ao reator da linha de transmissão, portanto uma das condições de existência dessas regras é a de que a linha deve possuir um reator. Assim, essas regras não podem ser aplicadas a linhas que não possuem um reator associado.

Foi justamente a parametrização das regras que possibilitou a redução de 1.334 para 51 regras para linhas de transmissão do CROL. Ao invés de escrever uma regra para cada problema de cada uma das linhas, foi escrita uma regra para cada tipo de problema de linhas de transmissão. Desde a elaboração das 51 regras, exibidas no Apêndice B, várias novas linhas de transmissão foram adicionadas à topologia do CROL, sem que houvesse necessidade de efetuar qualquer tipo de alteração na base de conhecimento.

4.2.3 Processo de obtenção das regras

O processo de obtenção das regras de diagnóstico é um processo árduo e sujeito a falhas uma vez que depende inteiramente do conhecimento dos especialistas. Para tentar minimizar as falhas, o processo foi quebrado em várias etapas; a cada etapa eram levantadas as regras para linhas de transmissão de uma determinada tensão e depois se iniciava um *loop* de validação das regras. Nesse *loop,* as regras levantadas foram implementadas no *Smart One* e refinadas, através de simulações com o banco de dados de eventos, até que os diagnósticos emitidos pela regras estivessem corretos, ou seja, fossem idênticos ao diagnóstico realizado por um especialista humano para o mesmo conjunto de eventos.

Esse processo foi realizado, no decorrer de oito meses, por duas equipes: uma encarregada de levantar as regras juntamente com os especialistas da CHESF e outra encarregada de implementar as regras no protótipo e validá-las juntamente com os especialistas da CHESF.

A equipe que realizou o levantamento das regras era formada por dois especialistas da CHESF (Antônio Sérgio de Araújo e Socorro Melo), por três professores do DSC-UFCG (Marcus Costa Sampaio, Jacques Philippe Sauvé e Jorge César Abrantes de Figueiredo) e por dois alunos do Mestrado em Informática da UFCG (Alexandre Nóbrega Duarte e Eloi Rocha Neto).

Já a equipe que realizou a implementação e validação das regras era formada por um especialista da CHESF (Antônio Sérgio de Araújo), por um professor do DSC-UFCG (Jorge César Abrantes de Figueiredo) e por um aluno do Mestrado em Informática da UFCG (Alexandre Nóbrega Duarte).

No Apêndice B, são descritas as 51 regras genéricas para as linhas de transmissão das 5 tensões da rede elétrica da CHESF que foram levantadas e validadas pelas duas equipes acima descritas.

4.3 Projeto Arquitetural da Ferramenta de Tratamento de Eventos

Nesta seção, serão listados alguns dos problemas a serem solucionados na construção de uma ferramenta de tratamento de eventos e as possibilidades tecnológicas para cada problema. Essa é uma abordagem *risk confronting*, um processo que auxilia no estabelecimento da arquitetura base da solução.

Buscando facilitar o trabalho de desenvolvimento, bem como garantir o cumprimento dos requisitos propostos para a solução, as tecnologias escolhidas devem atender principalmente às seguintes características:

- **Portabilidade** – A ferramenta deve funcionar em máquinas com diferentes sistemas operacionais como computadores pessoais com Microsoft Windows e Linux. Assim, as ferramentas devem ser bastante flexíveis, facilitando a portabilidade da aplicação para diferentes plataformas;
- **Voltada para redes –** O sistema de monitoração da rede elétrica com o qual a ferramenta de correlação interagirá é um sistema distribuído; dessa forma, a aplicação precisa ser capaz de se comunicar em rede.

Existem poucas soluções disponíveis, respeitando simultaneamente essas duas características. Uma delas é a tecnologia Java, que oferece uma solução independente de plataforma. Além da portabilidade, essa tecnologia possui outras características importantes

para o desenvolvimento de aplicações distribuídas: interface para acesso a bancos de dados, capacidade de multi-linhas de execução e suporte à programação em rede.

O padrão Java vem despontando como uma importante tecnologia para desenvolvimento de aplicações avançadas, permitindo a utilização de uma abordagem baseada em componentes de software. Componentes de software representam um importante passo no sentido de sistematizar a produção de software, ao prover reusabilidade num alto nível de abstração através da utilização apropriada de técnicas de orientação a objetos. Ferramentas visuais também podem ajudar o programador a "compor" aplicações através da conexão e da configuração dos componentes.

Todas essas características permitem o desenvolvimento de arquiteturas altamente flexíveis, facilitando a adição de novas funcionalidades e diminuindo o tempo e os custos envolvidos no desenvolvimento. No restante desta seção, será descrito como parte desse conjunto de novas e avançadas tecnologias pode ser empregada para atender aos requisitos do sistema de correlação de eventos.

4.3.1 Estrutura da aplicação

Inicialmente será analisada a arquitetura sobre a qual será estruturada a aplicação. O sistema de correlação de eventos será um módulo adicional do sistema de monitoração e aquisição de dados da CHESF, o SAGE. A Figura 4.6 ilustra os diferentes componentes da estrutura da aplicação. No primeiro nível, estão as subestações monitoradas pelas estações remotas de aquisição de dados. São as estações remotas que coletam os diversos dados disponíveis nas subestações e os enviam para o SAGE.

O SAGE é um sistema distribuído de tempo real que funciona em uma rede própria. Para evitar que o *Smart Alarms* interfira no escalonamento de tempo real do SAGE e que defeitos de software possam comprometer o seu bom funcionamento, foi desenvolvido pelo CEPEL um *gateway* de comunicação que provê ao *Smart Alarms* todas as informações necessárias para a realização do diagnóstico dos alarmes e um meio para que ele possa inserir no SAGE os novos alarmes obtidos com o diagnóstico dos eventos recebidos.

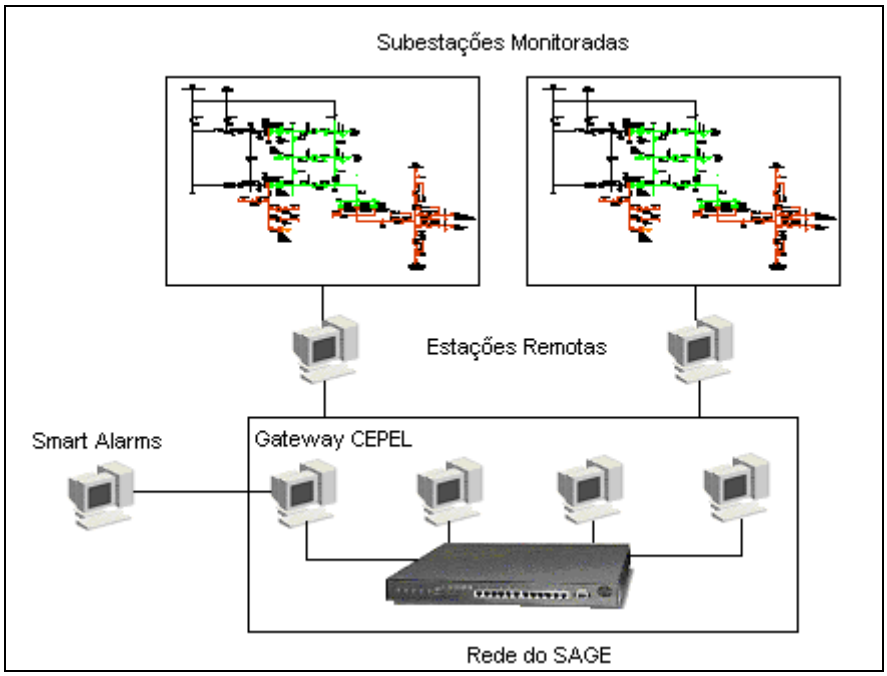

Figura 4.6: Estrutura de utilização do *Smart Alarms*

A principal vantagem na utilização do *gateway* de comunicação está na facilidade para testar o sistema em desenvolvimento. Como o *gateway* isola o *Smart Alarms* do restante do SAGE, ele evita que possíveis defeitos de software, comuns em todo processo de desenvolvimento, possam comprometer a estabilidade do SAGE. Adicionalmente, permite que sejam realizados testes remotos, evitando assim a necessidade constante de deslocamento de técnicos de Campina Grande para Recife, localização física do CROL.

4.3.2 Projeto Arquitetural

Nesta seção, será descrita a arquitetura interna do *Smart One,* a qual foi concebida para ser o mais flexível possível no tocante à inserção de novas funcionalidades no sistema. Para tal, foi utilizada uma abordagem baseada em componentes.

A Figura 4.7 apresenta a arquitetura básica do *Smart One.* Nela é possível observar o fluxo de informação dentro do sistema. O *gateway* CEPEL é o responsável por toda a comunicação entre o sistema de diagnóstico e o SAGE.

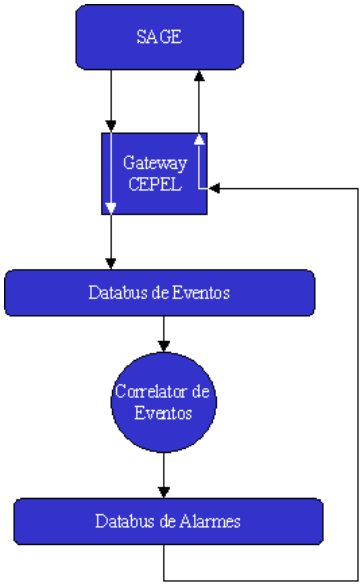

Figura 4.7: Arquitetura básica do *Smart One*

O *gateway* realiza dois tipos de operações:

- Insere os eventos recebidos pelo SAGE no databus de eventos do *Smart One;*
- Insere no SAGE os alarmes diagnosticados pelo *Smart One.*

O **correlator de eventos** é o núcleo do *Smart One* e é o responsável pelo diagnóstico propriamente dito. É no correlator de eventos que é implementada a técnica híbrida de correlação descrita na seção 4.2.

O **databus de eventos** e o **databus de alarmes** são os responsáveis por prover flexibilidade à arquitetura do sistema. Eles servem para isolar os **produtores de dados** dos **consumidores de dados**. Na arquitetura básica, o *gateway* não se comunica com o correlator de eventos para fornecer os eventos recebidos pelo SAGE, e sim, com o databus de eventos; da mesma forma, o correlator de eventos não se comunica com o *gateway* para inserir novos alarmes no SAGE, ele se comunica com o databus de alarmes que, por sua vez,fornece os novos alarmes para o *gateway* que os insere no SAGE.

A flexibilidade da arquitetura ficará mais evidente a seguir, quando forem exibidos os demais componentes da aplicação. Com a arquitetura básica, não é possível testar o sistema sem que ele esteja em comunicação com o SAGE; além disso, falhas em redes elétricas, felizmente, não são tão freqüentes para que se possa testar exaustivamente o sistema apenas com as informações recebidas pelo SAGE.

Para que seja possível testar o sistema sem a necessidade de comunicação com o SAGE e para que seja possível repetir falhas detectadas anteriormente na rede elétrica,a

arquitetura deve ser estendida com a adição dos componentes ilustrados em cinza na Figura 4.8.

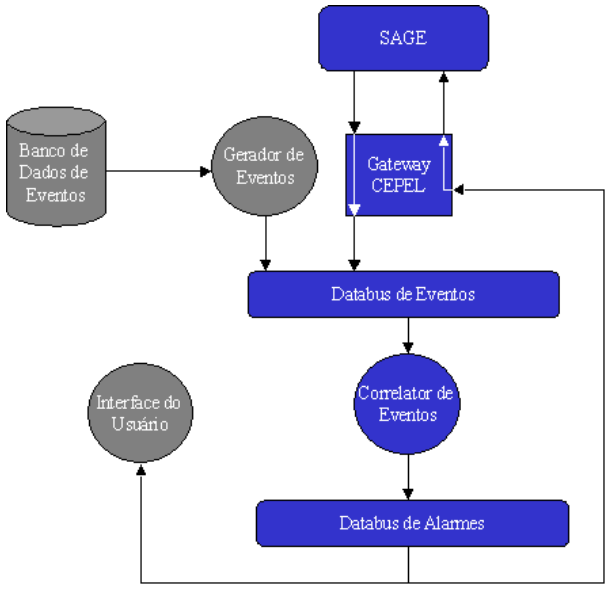

Figura 4.8: Extensão da arquitetura básica do *Smart One*

O **gerador de eventos** é o responsável por simular a comunicação com o SAGE através da reprodução de seqüências de eventos armazenadas em um **banco de dados**. Com o gerador de eventos, é possível repetir falhas anteriores detectadas na rede elétrica para realizar testes exaustivos no sistema de correlação de eventos. Além disso, é possível ajustar a velocidade da geração dos eventos para simular, por exemplo, um dia inteiro de monitoração da rede em poucos segundos.

Para exibir os alarmes diagnosticados, foi criado também um módulo de **interface do usuário**. A interface deve exibir a seqüência de eventos recebida pelo correlator e os alarmes diagnosticados.

Com a adição desses dois novos componentes, o *Smart One* passa a ser também uma poderosa ferramenta de treinamento ao permitir que sejam repetidas situações de estresse enfrentadas nos centros de controle nos momentos em que ocorreram falhas graves na rede elétrica.

Com a flexibilidade fornecida pelos **databus de eventos e de alarmes,** foi possível adicionar uma nova funcionalidade ao sistema sem que fosse necessário realizar qualquer alteração em seu núcleo. O correlator de eventos não precisa sofrer qualquer alteração para funcionar com os eventos recebidos do SAGE ou com os eventos oriundos do banco de dados uma vez que ele “não sabe” qual a origem dos eventos que está recebendo. Da mesma forma foi possível inserir um módulo de interface gráfica que tanto pode exibir os diagnósticos

realizados a partir de eventos oriundos do banco de dados quanto diagnósticos realizados a partir de eventos recebidos do SAGE.

Agora a arquitetura permite que falhas que ocorreram anteriormente na rede elétrica possam ser repetidas para realizar testes com o sistema de correlação de eventos, mas o que fazer com falhas raras que podem nunca ter ocorrido na prática ou que simplesmente não estão no banco de dados? Para que falhas que não estão presentes no banco de dados de eventos possam ser simuladas, foi adicionado mais um componente à arquitetura do sistema, um **editor de cenários**, que possibilita a criação de novas seqüências de eventos, correspondentes a falhas de ocorrência rara na rede elétrica.

Na Figura 4.9 é exibida a arquitetura final do *Smart One* onde é inserido o módulo do editor de cenários.

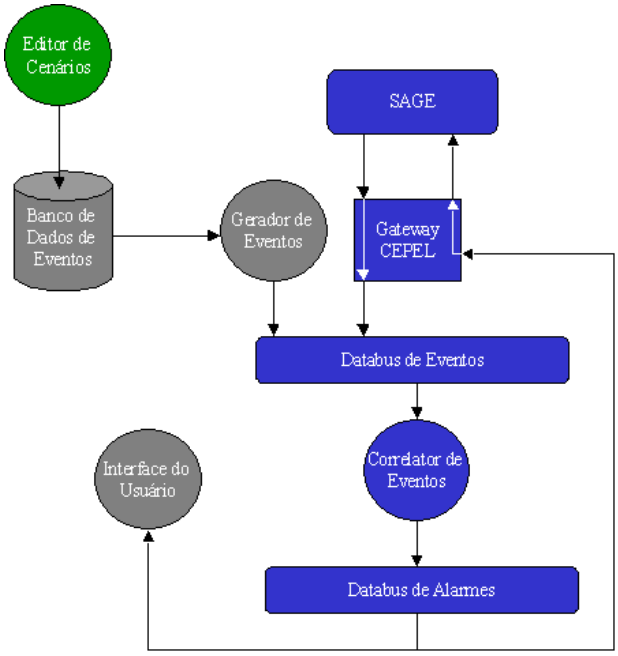

Figura 4.9: Arquitetura final do *Smart One*

4.4 Projeto detalhado

Nesta seção, é fornecido um maior detalhamento da arquitetura dos principais módulos dos *Smart One*: o módulo responsável pela geração dos eventos e o módulo responsável pela realização dos diagnósticos. Além disso, são fornecidos detalhes sobre o Databus e seu importante papel na arquitetura do sistema. Não é necessário detalhar com maior profundidade os demais módulos do sistema porque apenas esses três apresentam uma arquitetura mais complexa.

4.4.1 Gerador de Eventos

O módulo de geração de eventos tem a importante função de permitir que o *Smart One* possa ser testado sem a necessidade de se comunicar com o SAGE. Além disso, permite que seqüências de eventos previamente recebidas pelo SAGE e armazenadas em um banco de dados possam ser repetidas e que sejam criadas novas seqüências de eventos para falhas raras que nunca ocorreram na rede elétrica e não estariam, portanto, na base de dados caso ela contivesse apenas dados históricos. Outra vantagem é a possibilidade de simular uma grande quantidade de eventos em poucos segundos, acelerando o processo de testes e de treinamento.

A arquitetura do módulo de geração de eventos está ilustrada na Figura 4.10.

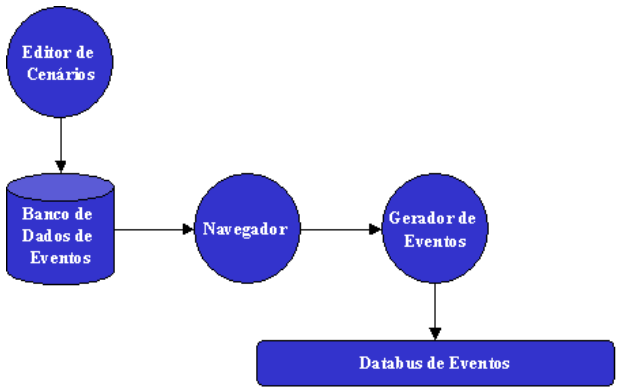

Figura 4.10: Arquitetura do módulo de geração de eventos

Todos os componentes do módulo de geração de eventos, com exceção do **Navegador**, já foram discutidos na seção anterior.

O navegador é uma ferramenta de busca e serve para facilitar a localização de seqüências de eventos no banco de dados. Com ele é possível localizar seqüências de eventos que contenham:

- Eventos de um determinado tipo;
- Eventos de um determinado tipo de equipamento;
- Eventos de um determinado equipamento;
- Eventos de um determinado dia.

Além disso, é possível combinar esses critérios de busca, utilizando o operador lógico **E**, para obter, por exemplo, eventos de um determinado tipo e de um determinado tipo de equipamento.

4.4.2 Correlator de Eventos

O módulo de correlação de eventos é o coração do *Smart One*. Ele é o responsável por diagnosticar os problemas da rede elétrica baseado nos eventos recebidos. A Figura 4.11 apresenta um esboço da arquitetura do módulo de correlação de eventos.

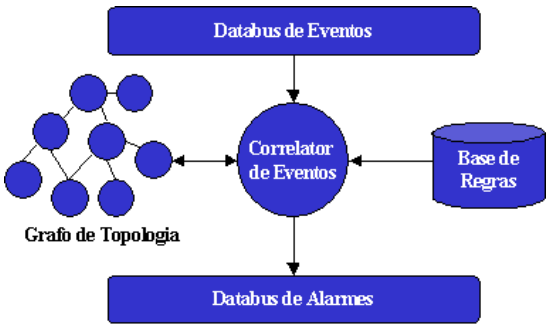

Figura 4.11: Módulo de correlação de eventos

No módulo de correlação de eventos, existem dois componentes que não foram discutidos anteriormente: o **grafo de topologia** e a **base de regras**. O **grafo de topologia** é o responsável por manter o estado da rede elétrica. O *Smart One* utiliza o grafo de topologia para estabelecer relações de conectividade entre os diversos equipamentos da rede. Além disso, ele armazena as mudanças de estado de cada um dos equipamentos da rede, representadas pela ocorrência de eventos. Por exemplo, na Figura 4.12, é exibido um grafo de topologia simples, que ilustra as conexões dos diversos equipamentos relacionados com a linha **05L8-Angelim II / Recife II**.

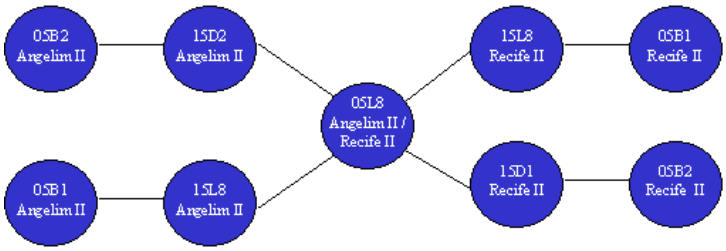

Figura 4.12: Grafo de topologia da linha 05L8-Angelim II / Recife II

Caso o sistema seja notificado da abertura dos disjuntores **15L8-Angelim II** e **15D2-Angelim II**, o grafo da Figura 4.12 será alterado para o grafo na Figura 4.13. Através de uma análise de conectividade no grafo da Figura 4.13, é possível detectar que houve um desligamento parcial da linha de transmissão 05L9 – Angelim II / Recife II no lado de Angelim II.

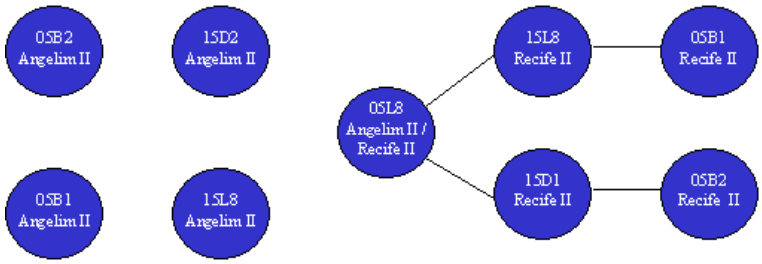

Figura 4.13: Novo grafo de topologia para linha 05L8-Angelim II / Recife II

As primitivas topológicas das regras genéricas são baseadas em relações de conectividade no grafo de topologia.

A **base de regras** é o repositório de regras genéricas para diagnosticar as possíveis falhas na rede elétrica. O correlator de eventos combina a regras contidas na base de regras com o conhecimento topológico armazenado no grafo de topologia para realizar o diagnóstico dos alarmes na rede elétrica.

Durante o projeto do *Smart One,* decidiu-se que a base de regras seria criada utilizando Java, dessa forma, cada regra é descrita por uma classe. A Figura 4.14 apresenta a classe que representa a regra para diagnosticar um **desligamento total de linha de transmissão sem atuação da proteção**.

```
public class DesligamentoTotalDeLTSemATPR extends Regra500KV{

  private static Regra instancia = null;

  private DesligamentoTotalDeLTSemATPR() {
  }

  public static Regra getInstancia() {
    if( instancia == null ) instancia = new DesligamentoTotalDeLTSemATPR();
    return instancia;
  }

  public boolean ativada( Equipamento equip ) {
    Linha linha = null;
    try { linha = (Linha) equip; }
    catch( ClassCastException cce ) { return false; }
    if( linha.getLinhaLadoDe() == null || linha.getLinhaLadoPara() == null )
       return false;
    return   linha.desarmeTotal() &&
             !linha.getEvento( GerenciadorDeEventos.ATPR_LD ) &&
             !linha.getEvento( GerenciadorDeEventos.ATPR_LP );
  }

  public String toString() {
    return "DESLIGAMENTO TOTAL DE LT SEM ATUACAO DA PROTECAO / DESARME MANUAL";
  }
}
```

Figura 4.14: Classe java representando a regra de desligamento total de linha sem atuação da proteção

Detalhes de implementação das regras serão discutidos no próximo capítulo.

4.4.3 Databus

O Databus tem a importante função de minimizar o acoplamento entre os diferentes módulos da aplicação. Com ele é possível quebrar a aplicação em partes independentes, como exibido na Figura 4.15, e que podem ser modificadas, testadas e substituídas independentemente umas das outras.

Figura 4.15: Separação dos módulos da aplicação

Um Databus pode funcionar de dois modos distintos: no primeiro modo, ele realiza uma espécie de *broadcast* ao enviar todas as informações que recebe dos produtores de dados para todos os consumidores de dados; no segundo modo, ele realiza uma propagação inteligente de dados; o consumidor, ao se registrar no Databus, fornece uma lista com seus **interesses;** quando o Databus recebe dados dos produtores, ele só os envia para os consumidores interessados nesses dados. Com isso, é possível diminuir e restringir o fluxo de informação dentro do sistema.

Na Figura 4.16, tem-se um exemplo prático de como o Databus consegue reduzir o acoplamento entre os componentes de uma aplicação. Suponha que a aplicação possua três produtores e três consumidores de dados e que todos os consumidores desejam receber dados de todos os produtores. Sem utilizar um Databus, é preciso fazer com que cada **Produtor** possua uma referência para cada um dos **Consumidores;** dessa forma, tem-se nove acoplamentos entre os componentes da aplicação. Além disso, caso um consumidor deixe de existir ou caso se queira adicionar um novo consumidor, todos os produtores serão afetados. Utilizando-se um Databus, é possível realizar a comunicação entre esses componentes com apenas seis acoplamentos: os três produtores e os três consumidores se registram no Databus. Não haverá qualquer alteração nos produtores; caso um consumidor deixe de existir ou se

queira adicionar um novo consumidor, simplesmente será removido ou adicionado um consumidor ao Databus.

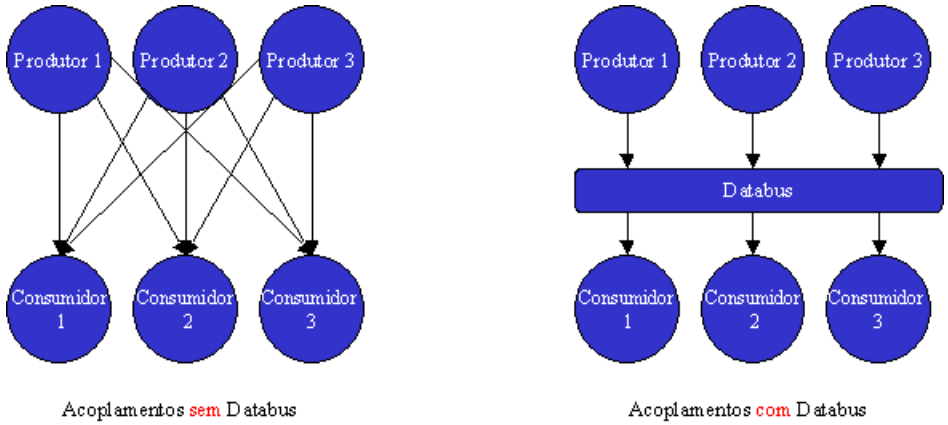

Figura 4.16: Acoplamento entre componentes sem e com Databus

Para casos como esse, pode-se ver que, sem utilizar Databus, serão necessários **P x C** acoplamentos, onde **P** é o número de produtores, e **C** é o número de consumidores. Já utilizando um Databus, serão necessários apenas **P + C** acoplamentos.

4.5 Considerações Finais

Neste capítulo, foram discutidos aspectos importantes do processo de desenvolvimento de qualquer aplicação: análise de requisitos e projeto arquitetural. Além disso, foi apresentada uma técnica híbrida para correlação de eventos que combina o raciocínio baseado em regras com o raciocínio baseado em modelos para obter uma base de regras mais flexível e mais facilmente atualizável devido a alterações na topologia da rede.

A análise de requisitos e o projeto da aplicação foram realizados de forma incremental, em três iterações. Assim, em cada iteração foram levantados os requisitos que o usuário gostaria de ver naquela versão da aplicação e foi elaborado um projeto arquitetural para atender a esses requisitos.

A arquitetura proposta mostrou-se bastante flexível devido à utilização do Databus. O baixo acoplamento provido pelo Databus permite que os vários componentes da aplicação possam ser substituídos sem implicar grandes modificações na estrutura da aplicação. Por exemplo, é bastante simples trocar a técnica de correlação de eventos por uma outra técnica, basta substituir o correlator de eventos. Outra opção seria utilizar duas técnicas de correlação de eventos diferentes ao mesmo tempo para comparar os diagnósticos; para isso, basta

adicionar um outro correlator de eventos. Essas duas modificações, desde que se tenha o novo correlator de eventos implementado, podem ser feitas em poucos minutos.

No próximo capítulo, serão discutidos os detalhes da implementação e a metodologia de testes do *Smart One.*

5 Uma Ferramenta de Tratamento de Eventos em Redes Elétricas: Implementação e Testes

Neste capítulo, são apresentados detalhes da implementação do *Smart One* e da metodologia de testes utilizada para validar o sistema. O *Smart One* é a primeira versão funcional do *Smart Alarms* e foi criado para validar a técnica híbrida de correlação de eventos, assegurando a sua aplicabilidade ao diagnóstico de falhas em redes elétricas. A funcionalidade do *Smart One* foi restrita ao diagnóstico das falhas em linhas de transmissão, o que representa aproximadamente metade do total de tipos de falhas na rede elétrica.

Além de testar a aplicabilidade da técnica híbrida de diagnóstico ao diagnóstico de falhas em redes elétricas, o *Smart One* também foi utilizado para validar o conhecimento extraído dos especialistas da CHESF sob a forma de regras de diagnóstico.

Este capítulo contém a organização geral do projeto de desenvolvimento (seção 5.1), seguida da enumeração das API[7]s desenvolvidas por terceiros e que foram utilizadas no projeto (seção 5.2). Mais adiante, são apresentadas as classes desenvolvidas (seção 5.3), o esquema lógico do banco de dados utilizado (seção 5.4) , a interface gráfica da aplicação (seção 5.5), a metodologia de testes adotada (seção 5.6) e, finalizando, algumas considerações finais sobre o assunto tratado no capítulo (seção 5.7).

5.1 Organização e Construção da Aplicação

O desenvolvimento desta aplicação se baseia nas etapas básicas de qualquer processo de desenvolvimento de software: análise do domínio do problema, projeto da solução, implementação e testes. As técnicas empregadas na engenharia de software atual apontam que um bom processo de desenvolvimento deve ser:

- **Iterativo -** com várias iterações no tempo;

[7] *Aplication Programming Interface*

- **Incremental -** ao final de cada iteração, devemos ter sempre novas versões incrementadas em funcionalidade e integradas em relação à versão anterior.

No capítulo 4, foi apresentado o projeto da solução, propondo uma arquitetura base para o cumprimento dos requisitos levantados durante a análise do domínio do problema. Os componentes dessa arquitetura foram organizados em pacotes de software de acordo com suas funcionalidades e implementados através das seguintes iterações:

Tabela 5.1:Iterações realizadas e pacotes desenvolvidos

Iteração	Módulo	Pacote	Responsabilidades
I	Iniciador	smartalarms	Iniciar a aplicação, instanciando e configurando seus principais componentes.
I	Correlator	smartalarms.correlator	Realizar o diagnóstico dos alarmes na rede elétrica.
I	Correlator	smartalarms.equipamentos	Modelar os equipamentos da rede elétrica e o grafo de conectividade utilizado no diagnóstico.
I	Infra-Estrutura	smartalarms.databus	Realizar a comunicação entre os diversos componentes do sistema e permitir que a arquitetura seja flexível.
I	Infra-Estrutura	smartalarms.util	Oferecer classes de utilidade geral para a aplicação.
I	Infra-Estrutura	smartalarms.log	Realizar a persistência das informações geradas pela aplicação.
I	Gerador de Eventos	smartalarms.sage	Simular o recebimento de eventos do SAGE para permitir testes da aplicação.
I	Gerador de Eventos	smartalarms.adaptador	Fornecer uma fonte transparente de dados para o gerador de eventos, permitindo que ele utilize diferentes repositórios de informação, como arquivos ou bancos de dados.
I	Interface Usuário	smartalarms.interfaceusuario	Fornecer uma interface padrão para a exibir as informações do sistema.
II	Interface Usuário	smartalarms.interfaceusuario	Fornecer uma interface gráfica que permita, entre outras coisas, selecionar subconjuntos de eventos para a simulação e ajustar os diversos parâmetros da aplicação.
III	Gerador de Eventos	smartalarms.bd	Prover acesso a bancos de dados
III	Gerador de Eventos	smartalarms.migracao	Converter os diferentes formatos de dados para um formato compatível com o banco de dados.
III	Interface Usuário Navegador	smartalarms.interfaceusuario	Fornecer uma forma eficiente para navegação nos dados contidos no banco de dados.
III	Interface Usuário Editor de Cenários	smartalarms.interfaceusuario	Permitir a criação de novas seqüências de eventos para simular falhas não contidas no banco de dados.

Além das classes implementadas, a aplicação utiliza outros arquivos e recursos organizados numa estrutura de diretórios, da seguinte forma:

- /SmartOne: diretório principal da aplicação onde são encontradas as classes, os arquivos e demais recursos utilizados e organizados em subdiretórios;

- /SmartOne/lib: bibliotecas de classes utilizadas;
- /SmartOne/src: código fonte das classes implementadas;
- /SmartOne/classes: *bytecode*[8] das classes implementadas;
- /SmartOne/cenarios: seqüências de eventos criadas, utilizando-se o editor de cenários e utilizadas para testar a aplicação;
- /SmartOne/config: arquivo de configuração da topologia (Apêndice A);
- /SmartOne/docs: javadocs especificando as APIs desenvolvidas;
- /SmartOne/logs: arquivos de log gerados pela aplicação;
- /SmartOne/images: imagens utilizadas na interface gráfica da aplicação.

5.2 APIs utilizadas

Nesta seção, são descritas as APIs fornecidas por outros desenvolvedores e que foram empregadas para a implementação do *Smart One*.

5.2.1 LOG4J

O LOG4J é parte do projeto Jakarta da Apache e possibilita o log de informações, permitindo um controle fino do que se deseja logar. Toda a configuração dos logs é feita em um arquivo de configuração que é lido durante a inicialização da aplicação. Com o LOG4J, é possível obter um contexto detalhado para as falhas de aplicação através da inserção de pontos de log nos locais críticos do processamento. Uma das funcionalidades dos LOG4J é a noção de herança de *loggers*. Com uma hierarquia de *loggers,* é possível controlar quais informações de log devem ser geradas com uma granularidade arbitrária. O LOG4J pode gravar dados em: consoles, arquivos, *OutputStreams,* objetos *java.io.Writer,* servidores remotos de log e muitos outros destinos.

5.2.2 Castor XML

O *Castor XML* é um *framework* Java desenvolvido pela Exolab que é utilizado para converter documentos XML em objetos Java e vice-versa. O *Castor* foi utilizado no projeto para realizar a leitura e o processamento do arquivo de configuração da topologia da rede

[8] O *bytecode* é o resultado da compilação de uma classe Java, sendo o código intermediário lido pelo interpretador Java.

elétrica. O arquivo de configuração utiliza XML e foi gerado automaticamente pelo Topogiggio, uma ferramenta que processa o arquivo de topologia em um formato proprietário da CHESF, extraindo os parâmetros topológicos das linhas de transmissão e gerando o arquivo de configuração do *Smart One* com todas as linhas de transmissão parametrizadas.

5.2.3 MonarchDate

O *MonarchDate* é um pacote com dois componentes gráficos para a manipulação de calendários. Suas principais funcionalidades são:

- seleção independente de anos e meses;
- realce de datas;
- seleção de datas em um intervalo definido;
- seleção múltipla de datas;
- internacionalização completa;
- associação de vários calendários em uma matriz.

O *MonarchDate* foi utilizado no navegador de eventos para possibilitar a seleção da data para a qual se deseja simular a seqüência de eventos.

5.2.4 API de Migração para o Banco de Dados

Esta API foi desenvolvida por Eloi Rocha Neto, aluno do Mestrado em Informática desta mesma instituição, e é responsável por processar os arquivos de eventos gerados pelo SAGE e inserir as seqüências de eventos no banco de dados, obedecendo ao modelo lógico do banco de dados discutido mais adiante.

5.3 Pacotes desenvolvidos

A Figura 5.1 dá uma visão geral da aplicação através de um diagrama que mostra as dependências entre seus pacotes. Nos tópicos seguintes, serão mostradas as classes desenvolvidas em cada pacote da aplicação, ressaltando alguns detalhes de implementação mais interessantes.

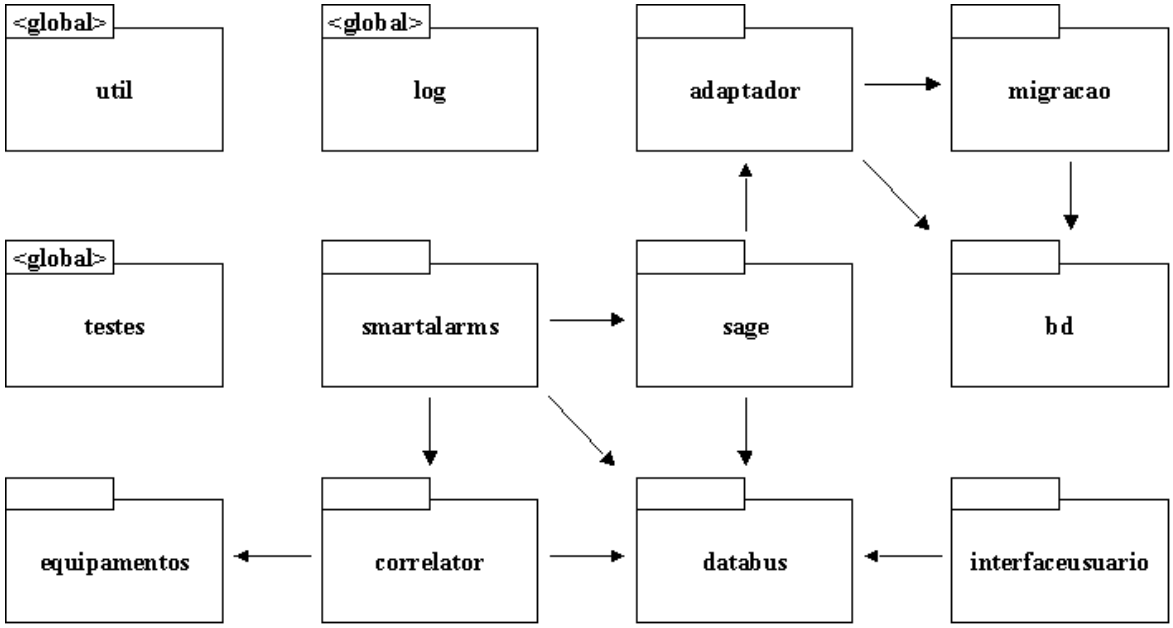

Figura 5.1: Relacionamento entre os pacotes desenvolvidos

5.3.1 Pacote smartalarms

Este pacote contém apenas uma classe executável chamada `smartone`, possuindo um método `main` responsável por instanciar os módulos da aplicação e associar os produtores de dados com o seu databus de saída e os consumidores de dados com o seu databus de entrada como especificado no capítulo 4.

São criadas duas instâncias da classe Databus: uma para eventos e outra para alarmes, como pode ser observado na Figura 5.2 que apresenta todos os componentes instanciados pelo *Smart One* e seus relacionamentos. O databus de eventos possui um produtor, o simulador do SAGE (Proxy), e dois consumidores: o correlator de eventos e a interface gráfica. O databus de alarmes possui um produtor, o correlator de dados e um consumidor: a interface gráfica.

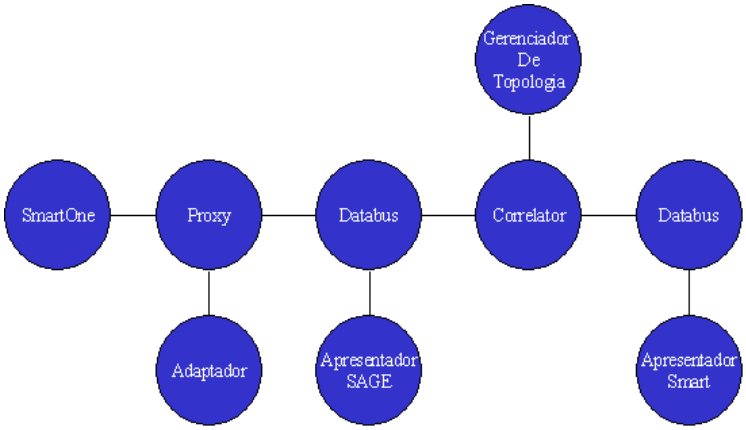

Figura 5.2: Diagrama de componentes instanciados do SmartOne

5.3.2 Pacote smartalarms.databus

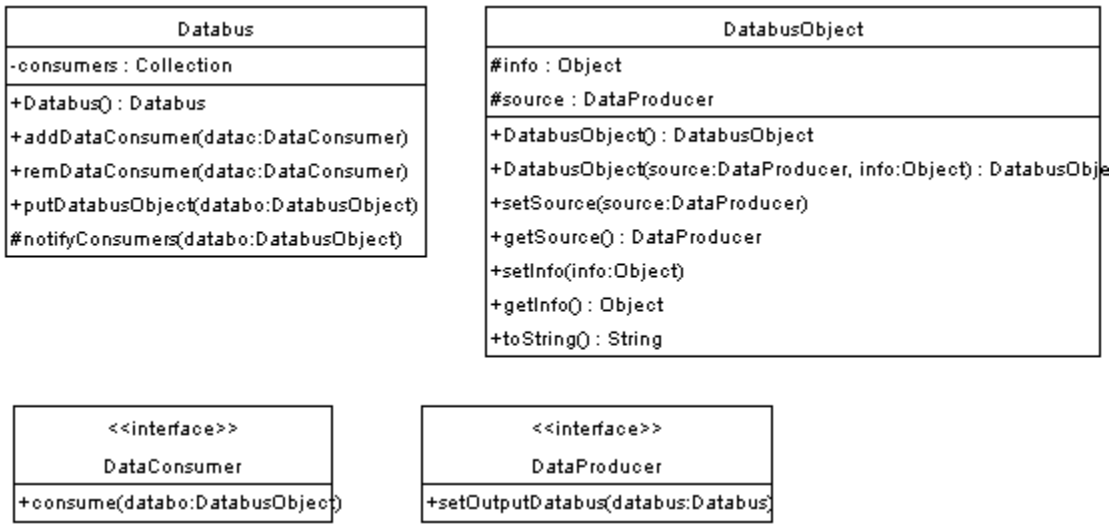

Figura 5.3: Diagrama de classes do pacote smartalarms.databus

Este pacote é o responsável pela distribuição de dados dentro do sistema e possui as seguintes classes e interfaces:

Databus: representa um barramento de dados que pode ter vários produtores e consumidores de dados; os produtores inserem dados no barramento, e esses dados são repassados para todos os consumidores;

DatabusObject: célula de dados que circula dentro do barramento. Possui um dado e um produtor associado. Dessa forma, permite que um consumidor, caso necessite, possa saber quem é o produtor do dado que ele está recebendo;

DataConsumer: interface que representa um consumidor de dados. Uma classe deve implementar esta interface caso queira receber dados de um barramento;

DataProducer: interface que representa um produtor de dados. Uma classe deve implementar esta interface caso queira inserir dados em um barramento.

5.3.3 Pacote smartalarms.correlator

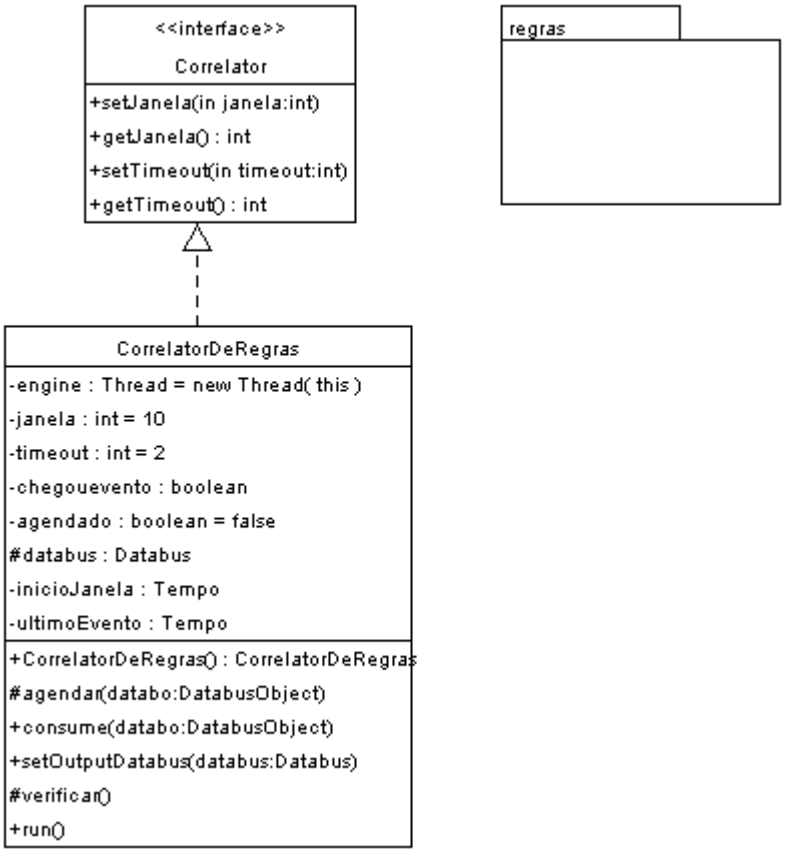

Figura 5.4: Diagrama de classes do pacote smartalarms.correlator

Este pacote é o responsável pelo diagnóstico das falhas na rede elétrica, contendo vários sub-pacotes para organizar as regras de diagnóstico para linhas de transmissão. Possui as seguintes classes e interfaces:

Correlator: interface que representa um correlator de eventos. Uma classe deve implementar esta interface caso queira ser um correlator de eventos.

CorrelatorDeRegras: classe que implementa a interface Correlator e realiza o diagnóstico utilizando regras genéricas, discutidas no capítulo 4.

No correlator de regras há dois parâmetros configuráveis: o *timeout* de diagnóstico e o *timeout* de inatividade. O *timeout* de diagnóstico é o período de tempo que o correlator deve aguardar, desde a chegada do primeiro evento, para realizar o diagnóstico. Esse *timeout* permite que o diagnóstico seja realizado após uma possível tempestade de eventos, eliminando diagnósticos desnecessários. O *timeout* de inatividade é o tempo que o correlator deve aguardar, depois do recebimento do último evento, para considerar que a espera por novos eventos deve ser interrompida, mesmo antes do fim do *timeout* de diagnóstico. Dessa

forma, com um *timeout* de diagnóstico de 10 segundos e com um *timeout* de inatividade de 5 segundos, o correlator de eventos irá aguardar 10 segundos depois da chegada do primeiro evento para realizar o diagnóstico, a menos que ele passe 5 segundos sem receber nenhum evento, o que implicaria uma antecipação do diagnóstico.

Uma descrição de cada uma das regras criadas pode ser encontrada no Apêndice B.

5.3.4 Pacote smartalarms.equipamentos

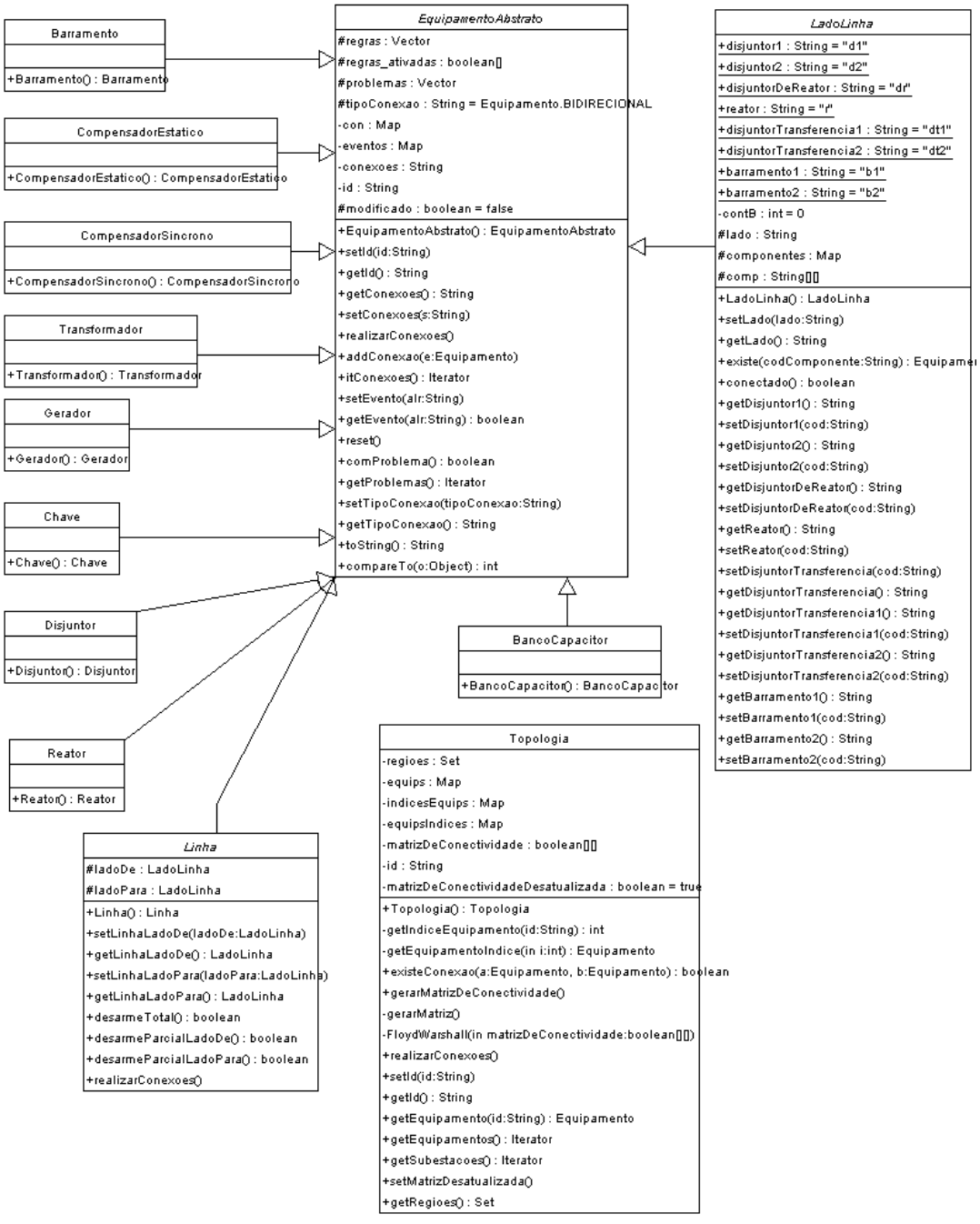

Figura 5.5: Diagrama parcial de classes do pacote smartalarms.equipamentos

Este pacote foi criado para representar as entidades especificas do sistema elétrico. Além disso, ele é o responsável pelas informações topológicas da rede elétrica, utilizadas pelo correlator de eventos durante o diagnóstico das falhas e detem as seguintes classes e interfaces:

GerenciadorDeTopologia: Esta classe é utilizada para realizar a configuração do grafo de topologia, responsável pela informação de conectividade do sistema, através de informações oriundas de um arquivo de definição de topologia. O arquivo de topologia é escrito em XML, e sua decodificação é feita com a biblioteca *Castor*. A classe GerenciadorDeTopologia obedece ao padrão de projeto *Singleton* para garantir que apenas uma instância da classe seja criada, evitando que o arquivo de configuração seja processado mais de uma vez.

Topologia: Representa o grafo de topologia propriamente dito. É responsável por prover informações de conectividade sobre todos os equipamentos da rede. O grafo de topologia é representado por uma matriz de conectividade que inicialmente possui valores *true* na linha *i*, coluna *j*, caso o equipamento de índice *i* esteja conectado diretamente ao equipamento de índice *j* e *false* caso contrário. Apenas essa informação não é suficiente para o diagnóstico, pois é necessário saber se um equipamento está conectado a outro de forma indireta. Esse é um problema clássico da teoria dos grafos, conhecido por *Transitive Closure,* e que pode ser resolvido empregando-se um algoritmo com complexidade $O(n^3)$ criado por Floyd e Warshall (FLOYD, 1962).

A classe topologia é a responsável por implementar as primitivas topológicas. Essas primitivas resumem-se a pesquisas no grafo de topologia. Por exemplo, a primitiva Conectado(Linha,Barra) tem complexidade O(1) e é realizada com uma simples checagem do grafo de topologia. Já a primitiva Desligamento total(Linha) tem complexidade O(n), onde n é o número de conexões da Linha, e verifica todas as conexões da linha para saber se a ela tem conexão com alguma barra.

Regiao: Essa classe foi criada para permitir uma melhor organização do arquivo de configuração; desta forma, o arquivo é dividido em regiões como CROL (Centro Regional de Operação Leste), CROS (Centro Regional de Operação Sul), CRON (Centro Regional de Operação Norte), etc.

Subestacao: Classe criada para permitir que equipamentos possam ser divididos em subestações, organizando melhor o arquivo de configuração.

Equipamento: Interface que representa um equipamento da rede elétrica; uma classe deve implementar esta interface caso queira ser tratada como um equipamento.

EquipamentoAbstrato: Classe abstrata que implementa a interface equipamento e provê métodos úteis para equipamentos concretos, evitando a reescrita de métodos comuns.

Foram criadas classes para representar cada um dos equipamentos discutidos no capítulo 2, a saber:

- **BancoCapacitor**
- **Barramento**
- **Chave**
- **CompensadorEstatico**
- **CompensadorSincrono**
- **Disjuntor**
- **Gerador**
- **Reator**
- **Transformador**
- **Linha**

O caso das linhas de transmissão é diferente do dos demais equipamentos. Uma linha de transmissão possui dois lados: um lado na subestação de origem e outro na subestação de destino. Cada um desses lados foi representado por uma classe chamada LadoLinha. Dessa forma, a classe Linha possui dois objetos do tipo LadoLinha para representar cada um dos seus lados.

Ficou constatado também que linhas de tensões diferentes precisariam ser tratadas de forma diferente. Assim, foram criadas classes para linhas de transmissão de 13, 69, 138, 230 e 500 KV, com suas respectivas classes para representar cada lado da linha. Com isso, a classe Linha13KV possui duas instâncias da classe LadoLinha13KV, a classe Linha69KV, duas instâncias da classe LadoLinha69KV, e assim por diante.

5.3.5 Pacote smartalarms.sage

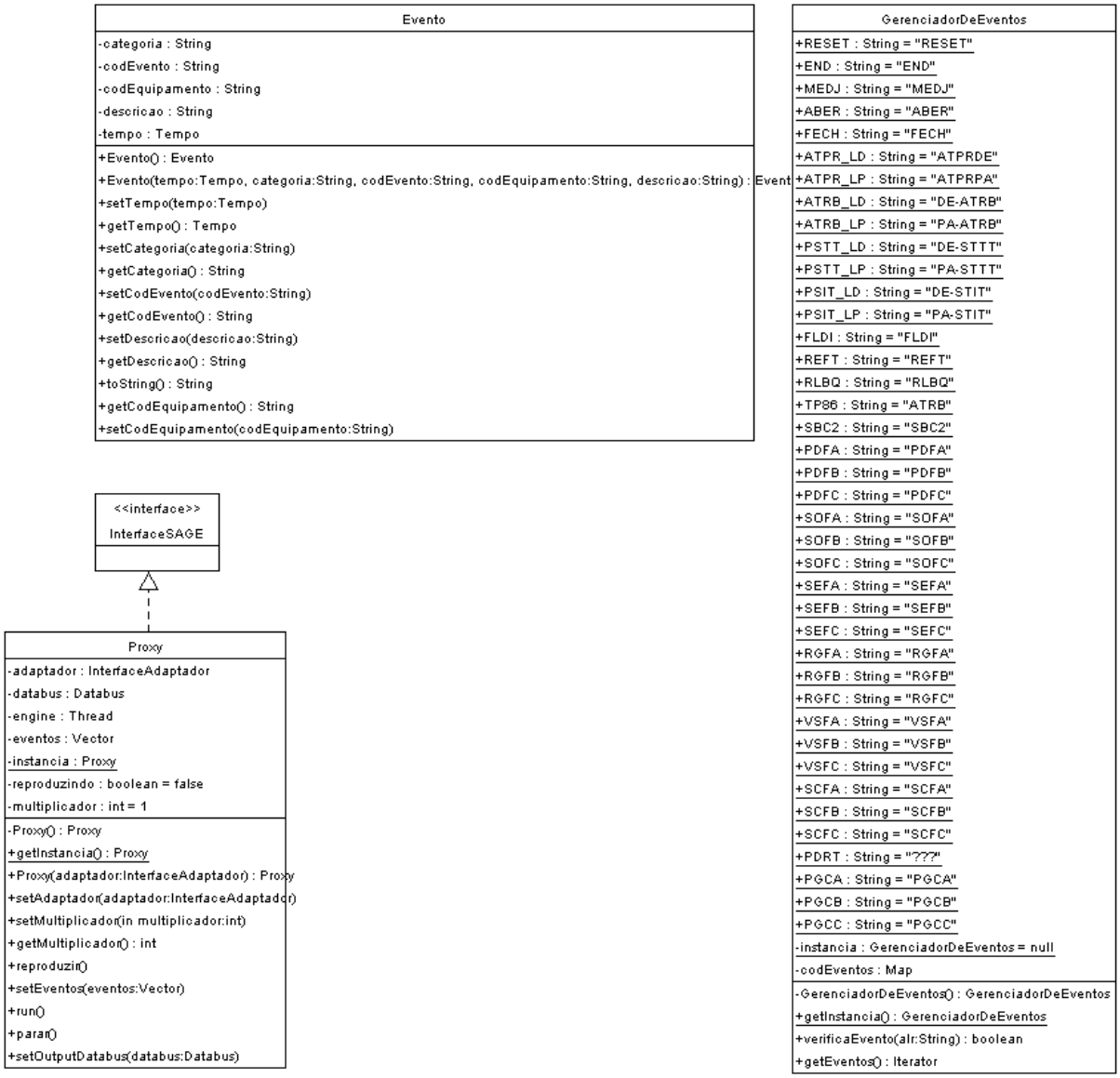

Figura 5.6: Diagrama de classes do pacote smartalarms.sage

Este pacote foi criado para permitir que o sistema fosse testado sem a necessidade de estar em comunicação com o SAGE. Possui as seguintes classes e interfaces:

InterfaceSAGE: Interface que indica que uma classe é um gerador de eventos (como o SAGE o é, do ponto de vista do *Smart Alarms*). Uma classe deve implementar esta interface caso deseje atuar como um simulador do SAGE.

Proxy: É um simulador do SAGE, que implementa a interface InterfaceSAGE e gera eventos a partir de uma fonte de dados, determinada pelo adaptador utilizado. Adaptadores serão comentados mais adiante. O Proxy permite que a velocidade da geração de eventos possa ser configurada de acordo com a necessidade do usuário.

O controle de aceleração do simulador do SAGE é baseado no *timestamp* dos eventos, dessa forma, a simulação acelerada mantém as mesmas janelas de diagnóstico que a simulação em tempo real. Com uma aceleração zero, o simulador gera um evento, calcula a diferença de tempo para o próximo evento e espera essa diferença de tempo antes de gerar o próximo evento. Caso a aceleração seja diferente de zero, o simulador divide o intervalo de tempo que deveria esperar para gerar o próximo evento pela aceleração especificada.

Seja qual for a aceleração utilizada, o simulador mantém sempre as mesmas janelas de diagnóstico, ou seja, o correlator de eventos sempre irá tratar o mesmo subconjunto de eventos a cada vez que o diagnóstico for realizado. Para garantir isso, o *timeout* de diagnóstico e o de inatividade também têm que ser baseados nos *timestamps* dos eventos. Dessa forma, quando se inicia uma janela de diagnóstico, o correlator armazena o *timestamp* do primeiro evento; caso a diferença de tempo entre o *timestamp* de um evento recebido e o *timestamp* do primeiro evento da janela seja maior que o *timeout* de diagnóstico ou a diferença entre o *timestamp* de um evento recebido e o *timestamp* do último evento recebido seja maior que o *timeout* de inatividade, o diagnóstico é realizado.

Gerenciador de Eventos: Esta classe é utilizada para identificar os eventos como válidos ou não antes que eles sejam processados pelo correlator de eventos.

Evento: Representa cada um dos eventos recebidos pela aplicação.

5.3.6 Pacote smartalarms.interfaceusuario

Este pacote fornece as classes necessárias para a implementação da interface do usuário. Existem duas interfaces disponíveis: uma em modo texto, empregada na primeira iteração do projeto, e outra em modo gráfico, utilizada nas demais iterações. A interface gráfica foi desenvolvida com componentes Swing, que é uma biblioteca avançada fornecida pela linguagem Java para a confecção de interfaces gráficas. Este pacote possui as seguintes classes e interfaces:

Apresentador: Interface que representa um apresentador de dados. O qual é uma classe capaz de receber dados de um databus e exibi-los para o usuário de alguma forma.

ApresentadorPadrao: Esta classe representa um apresentador para interface em modo texto.

ApresentadorGrafico: Classe que representa um apresentador para interface gráfica e que estende a classe JInternalFrame da biblioteca Swing para configurá-lo de acordo com as necessidades da aplicação.

ApresentadorGraficoProxy: Estende a classe ApresentadorGrafico para, além de apresentar os eventos gerados pelo Proxy, fornecer comandos para configurar os parâmetros da geração de eventos, como a velocidade da simulação e para permitir que sejam selecionadas subconjuntos das seqüências de eventos disponíveis.

ApresentadorGraficoSmart: Estende a classe ApresentadorGrafico para fornecer comandos para configurar os parâmetros do correlator de eventos, como *timeout* de diagnóstico e apresentar as falhas diagnosticadas.

GerenciadorDeInterface: Controla o funcionamento da interface gráfica, como alinhamento das janelas e textos dos menus.

EditorDeCenarios: Editor que permite a criação de seqüências novas de eventos, para a simulação de falhas não encontradas no banco de dados. Esse componente foi criado pelo aluno Eloi Rocha Neto, aluno do curso de Mestrado em Informática desta mesma instituição.

Navegador: Fornece a interface gráfica para consultas ao banco de dados de eventos. É o navegador que possibilita a localização de informações especificas dentro do banco de dados de eventos.

SmartFrame: Janela principal da aplicação, com três apresentadores gráficos, um navegador de eventos e um editor de cenários.

5.3.7 Pacote smartalarms.adaptador

Este pacote é o responsável por prover uma fonte de dados transparente para o simulador do SAGE. Com adaptadores, é possível isolar o meio físico do meio lógico; assim o simulador do SAGE pode obter seqüências de eventos tanto de um banco de dados quanto de um arquivo com um cenário criado com o editor de cenários.

Este pacote é composto das seguintes classes e interfaces:

InterfaceAdaptador: Esta interface indica que uma classe é uma fonte de dados.

AdaptadorBD: Um adaptador para acesso a um banco de dados.

AdaptadorCenarios: Um adaptador para acesso a arquivos criados com o editor de cenários.

AdaptadorTopologia: Um adaptador para acesso ao arquivo de definição da topologia da rede elétrica.

5.3.8 Pacote smartalarms.bd

Este pacote encapsula o código necessário para realizar a conexão e as consultas ao banco de dados, constituido das seguintes classes e interfaces:

GerenciadorDeConexoesBD: Gerencia as conexões ao banco de dados. É a classe responsável por carregar o driver JDBC específico para o banco de dados utilizado.

ConexaoBDIF: Interface que indica que uma classe fornece acesso a um banco de dados.

ConexaoBDMicrosoft: Classe que possibilita a conexão e consultas ao banco de dados Microsoft SQL Server 2000, utilizado para armazenar as seqüências de eventos.

5.3.9 Pacote smartalarms.log

Este pacote gerencia a utilização do LOG4J para realizar o log das informações geradas pelo sistema durante as diferentes fases do processamento dos eventos e é constituído apenas da classe `LogManager` que gerencia o processo de log das informações.

5.3.10 Pacote smartalarms.util

Este pacote foi criado para agrupar as classes que oferecem funções úteis a todo o restante da aplicação, com apenas a classe `Tempo,` que representa um determinado instante no tempo e é utilizada como *timestamp* para os eventos e alarmes.

5.3.11 Pacote smartalarms.testes

Neste pacote estão as classes responsáveis pelos testes realizados no sistema, e uma classe para testar cada uma das regras criadas. A metodologia de testes será discutida em detalhes na seção 5.7.

5.4 Esquema Lógico do Banco de dados

Nesta seção, serão apresentadas as tabelas criadas no banco de dados para armazenar as diferentes informações necessárias para a simulação da monitoração da rede elétrica. Foi utilizado o gerenciador de banco de dados Microsoft SQL Server 2000, devido a sua disponibilidade no ambiente de desenvolvimento. Para realizar o acesso ao banco de dados foi aplicado o driver JDBC, fornecido pela própria Microsoft para acesso ao SQL Server 2000. A seguir é detalhada cada uma das tabelas criadas.

5.4.1 Tabela de Equipamentos

Esta tabela associa um código a cada tipo de equipamento encontrado na rede elétrica. Constitui-se de duas colunas, uma com um número inteiro para armazenar o ID do equipamento e outro com uma seqüência de caracteres para armazenar o tipo do equipamento. Seu conteúdo pode ser visto na Tabela 5.2.

Tabela 5.2: Tabela de Equipamentos

id	tipo_equip
1	LINHA
2	DISJUNTOR
3	TRANSFORMADOR
4	BARRAMENTO
5	REATOR
6	CHAVE
7	COMPENSADOR
8	RELIGADOR
9	SECCIONADORA
10	OUTRO

5.4.2 Tabela de Eventos

Esta tabela armazena os eventos coletados pelas estações remotas de monitoramento e recebidos pelo SAGE. Cada linha desta tabela representa um evento. Possui 8 colunas, a saber:

- **tempo:** É um campo do tipo *time* que armazena o exato instante em que o evento foi coletado pelas estações remotas de monitoração.
- **categoria:** Seqüência de caracteres que representa a categoria do evento, A qual do indica o seu tipo.
- **subestacao:** Seqüência de caracteres que representa a subestação do equipamento no qual o evento ocorreu.

- **cod_equip:** Seqüência de caracteres que representa o código do equipamento no qual o evento ocorreu.
- **cod_evento:** Seqüência de caracteres que identifica de forma inequívoca o evento que ocorreu.
- **tipo_equip:** Índice para a tabela de equipamentos que indica o tipo do equipamento no qual o evento ocorreu.
- **descricao:** Seqüência de caracteres que fornece uma breve descrição sobre o evento.
- **data:** Índice para a tabela de tempo que indica a data em que o evento ocorreu.

5.5 Interface da aplicação

A interface gráfica da aplicação precisa atender a três requisitos principais:

- Permitir que sejam selecionadas seqüências de eventos para simulação e subconjuntos dessas seqüências;
- Permitir que a saída exibida antes do processamento dos eventos seja comparada com a saída após o processamento;
- Possibilitar a configuração dos diversos parâmetros da aplicação de forma simples e intuitiva.

Para atender a estes requisitos, foi criada a interface gráfica exibida na Figura 5.7.

A interface principal da aplicação é composta de três componentes:

- A janela do gerador de eventos: na parte superior da janela principal, permite que sejam selecionados subconjuntos das seqüências de eventos e configurar a aceleração da simulação. Possui também os controles para iniciar ou interromper a simulação e para recuperar os eventos do dia anterior ou do dia seguinte aos que estão sendo exibidos.
- A janela do emulador do SAGE: na parte central da janela principal, exibe a seqüência de eventos que deve ser analisada pelo operador caso não seja realizado processamento algum nos eventos. É uma replicação do que é exibido pelo SAGE durante a monitoração normal da rede elétrica.

- A janela do *Smart One*: na parte inferior da janela principal, exibe os diagnósticos realizados pelo mecanismo de inferência. Permite configurar os dois parâmetros do mecanismo de inferência: *timeout* de inatividade e *timeout* de diagnóstico.

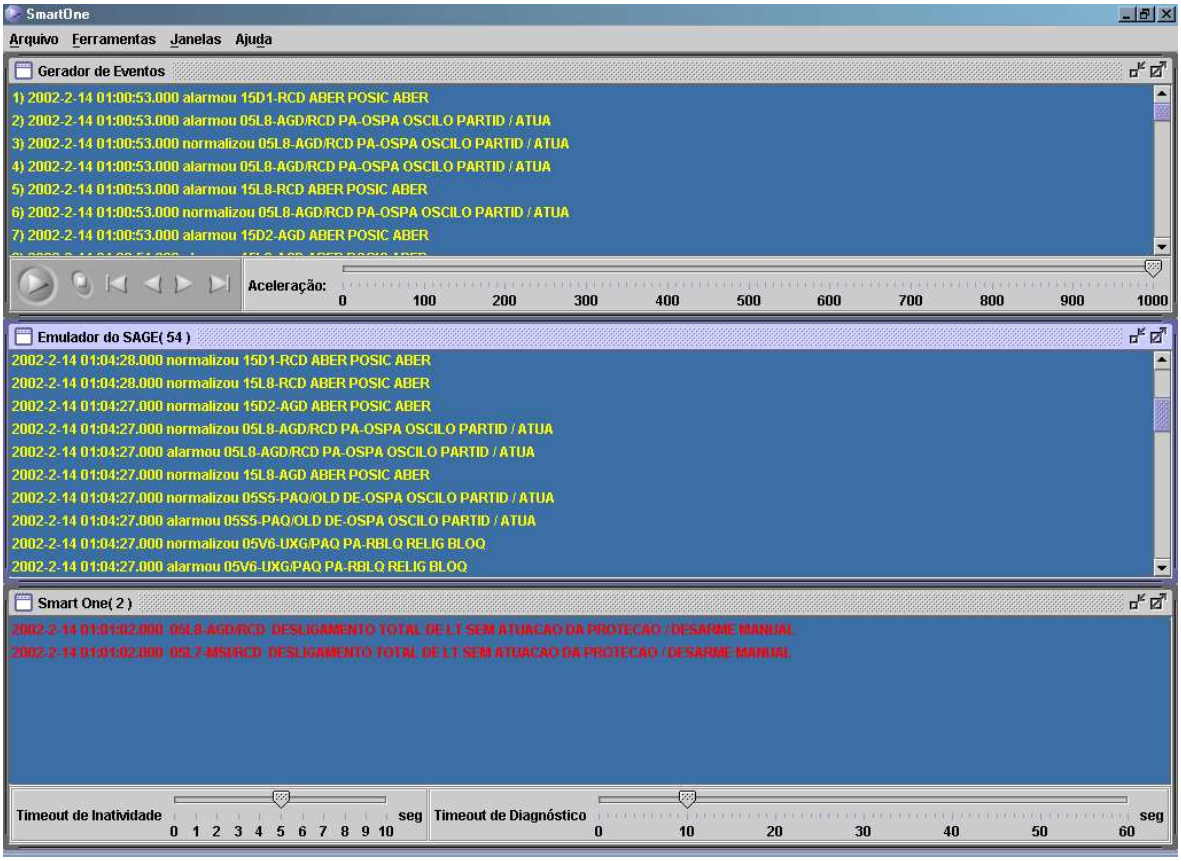

Figura 5.7: Interface principal da aplicação

Além da janela principal, a aplicação possui outras duas janelas: uma para o navegador de eventos e outra para o editor de cenários.

5.5.1 Navegador de Eventos

O navegador de eventos, exibido na Figura 5.8, é o responsável por realizar as pesquisas no banco de dados onde estão armazenadas as seqüências de eventos. Na janela do navegador, é possível selecionar os diferentes parâmetros de busca, cujo resultado é exibido utilizando-se um calendário fornecido pelo *MonarchDate.* A seleção da seqüência de eventos desejada é realizada com cliques do mouse nos dias exibidos no calendário.

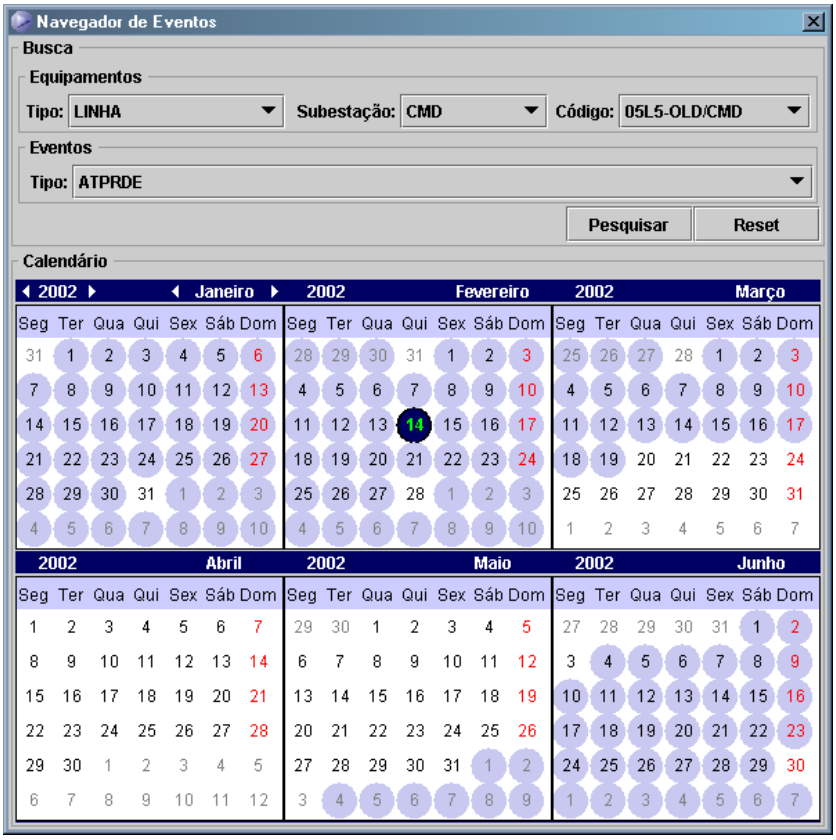

Figura 5.8: Navegador de Eventos

5.5.2 Editor de Cenários

O editor de cenários, exibido na Figura 5.9, permite que sejam criadas novas seqüências de eventos para simular falhas não encontradas no banco de dados de eventos. Este componente foi criado integralmente por Eloi Rocha Neto, aluno do Mestrado em Informática desta mesma instituição.

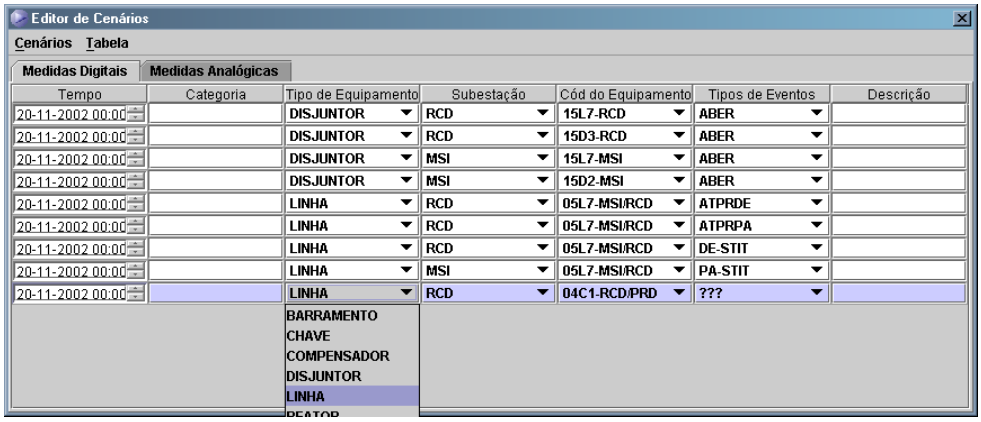

Figura 5.9: Editor de cenários

5.6 Metodologia de testes

Testar software é uma atividade que deve ser realizada a custo baixo, para que seja executada com freqüência e, portanto, deve ser automatizada. Para a implementação de testes automáticos, empregou-se o *framework* JUnit (GAMMA, 1999), que auxilia na construção e execução de testes de unidade para código escrito em Java.O JUnit define como estruturar os testes e provê as ferramentas para executá-los. Além das classes definidas no projeto, foram implementadas outras classes para testar as classes construídas. Depois de efetuados os testes, passa-se a ter uma maior confiança no código implementado e, quando é necessário realizar alguma modificação em uma classe que já funciona, os testes aumentam a confiança de que a classe ainda está funcionando da forma desejada.

Usando JUnit, os testes implementados são subclasses de `TestCase`. Como em Java as classes estão contidas em pacotes, é preciso decidir onde colocar as classes de testes. Além dos pacotes desenvolvidos, foi criado o pacote `smartalarms.testes`, que contém todos os testes das classes construídas para aplicação. Por exemplo, para testar a classe `smartalarms.databus.Databus`, foi criada uma classe chamada `smartalarms.testes.databus.TestDatabus`, que estende as funcionalidades de `TestCase`, adicionando os testes que serão realizados.

Os métodos das classes de testes são métodos desprovidos de parâmetros, que chamam os métodos das classes testadas e comparam o resultado obtido com o resultado esperado no teste. Utilizando JUnit, é possível ainda executar automaticamente todos os testes construídos através da linha de comando ou com ajuda de ferramentas visuais, facilitando a localização e correção dos erros identificados. Uma limitação dessa abordagem é que ela não permite que os métodos privados sejam testados diretamente, uma vez que eles não podem ser invocados pela classe de teste. Foram criados 55 testes diferentes para as classes da aplicação.

Apesar de os testes mencionados acima ajudarem a localizar possíveis erros no software, eles não garantem o mais importante: que o resultado do processamento esteja de acordo com o esperado, no caso específico do *Smart One,* se o diagnóstico está correto. Para isso, são necessários testes adicionais, chamados testes de aceitação, que testam o software como um todo, e não, cada classe individualmente. Para o *Smart One* foram desenvolvidos 51 testes de aceitação, um para cada regra de diagnóstico, os quais foram criados em conjunto com os especialistas da CHESF e são formados por uma seqüência de eventos e por uma seqüência de diagnósticos. Para cada seqüência de eventos foram feitos diagnósticos manuais

pelos especialistas da CHESF e são esses diagnósticos que devem ser emitidos pelo *Smart One* durante a simulação de cada seqüência de eventos.

Na Figura 5.10, pode ser observado um dos testes de aceitação criados e o diagnóstico emitido pelos especialistas da CHESF.

```
Seqüência de Eventos:
Data e Hora do Evento          Equipamento       Código do Evento
2002-10-20 00:00:01.000,       15L7-RCD,         ABER
2002-10-20 00:00:02.000,       15D3-RCD,         ABER
2002-10-20 00:00:03.000,       15L7-MSI,         ABER
2002-10-20 00:00:04.000,       15D2-MSI,         ABER
2002-10-20 00:00:05.000,       05L7-MSI/RCD,     ATPRDE
2002-10-20 00:00:06.000,       05L7-MSI/RCD,     ATPRPA
2002-10-20 00:00:07.000,       05L7-MSI/RCD,     PA-STTT
2002-10-20 00:00:08.000,       05L7-MSI/RCD,     PA-ATRB

Diagnóstico:
Desarme total da linha 05L7-MSI/RCD por sobretensão no lado para.
```

Figura 5.10: Um teste de aceitação do *Smart One*

Para automatizar os testes de aceitação, foi utilizado também o JUnit. O processo de testes consiste em simular todas as seqüências de eventos criadas e comparar os diagnósticos emitidos pelo *Smart One* com os diagnósticos feitos pelos especialistas humanos, embutidos nos testes como resultado esperado.

5.7 Considerações finais

Neste capítulo foi apresentado todo o processo de implementação do *Smart One*, que é um protótipo do *Smart Alarms* e cujo desenvolvimento foi realizado em três iterações, no decorrer de oito meses. Foi utilizado o *Borland JBuilder* 7.0 para criar as 239 classes Java do sistema, incluindo as classes feitas para testar a aplicação, totalizando 16681 linhas de código. O protótipo desenvolvido apresentou-se bastante estável e seus diagnósticos são compatíveis com os diagnósticos dos especialistas humanos.

Durante a fase de testes do sistema, utilizou-se um computador pessoal com processador AMD Athlon XP 1900+ com 512 MB de memória RAM. O sistema mostrou-se bastante estável e com um desempenho satisfatório. Como campo de testes, foi empregada a rede elétrica do CROL, com cerca de 10.000 dispositivos, e o sistema foi capaz de processar a matriz de conectividade, com tamanho 10.000 x 10.000 e principal gargalo do diagnóstico, em cerca de dois segundos. Esse tempo foi obtido com uma pequena alteração no algoritmo de Floyd e Warshall (FLOYD, 1962) que, inicialmente, levava 10 minutos para processar a

matriz. A alteração no algoritmo consistiu em quebrar a condição de comparação em duas partes, na Figura 5.11 pode ser visto o algoritmo original. Observe a comparação realizada nas linhas 6 e 7, na Figura 5.12 essa comparação foi quebrada nas linhas 5 e 7.Com essa simples alteração, o tempo de execução do algoritmo, para uma matriz de 10.000 linhas e 10.000 colunas, caiu de dez minutos para dois segundos.

```
1  private void FloydWarshall( boolean [][] matrizDeConectividade ) {
2
3      for( int intermediario = 0; intermediario < matrizDeConectividade.length; intermediario++ )
4          for( int origem = 0; origem < matrizDeConectividade.length; origem++ )
5              for( int destino = 0; destino < matrizDeConectividade.length; destino++ )
6                  if( matrizDeConectividade[ origem ][ intermediario ] &&
7                      matrizDeConectividade[ intermediario ][ destino ] )
8                      matrizDeConectividade[ origem ][ destino ] = true;
9
10 }
```

Figura 5.11: Algoritmo de Floyd-Warshall original

```
1  private void FloydWarshall( boolean [][] matrizDeConectividade ) {
2
3      for( int intermediario = 0; intermediario < matrizDeConectividade.length; intermediario++ )
4          for( int origem = 0; origem < matrizDeConectividade.length; origem++ )
5              if( matrizDeConectividade[ origem ][ intermediario ] )
6                  for( int destino = 0; destino < matrizDeConectividade.length; destino++ )
7                      if( matrizDeConectividade[ intermediario ][ destino ] )
8                          matrizDeConectividade[ origem ][ destino ] = true;
9
10 }
```

Figura 5.12: Algoritmo de Floyd-Warshall modificado

6 Conclusões e Sugestões para Trabalhos Futuros

Este trabalho abordou todo o processo de especificação, projeto e implementação de uma ferramenta de diagnóstico automático de falhas em sistemas de transmissão de energia elétrica. Além disso, apresentou uma técnica híbrida de correlação de eventos que elimina a maior das deficiências do raciocínio baseado em regras: a sua limitação quando aplicado a redes com constantes modificações de topologia.

A técnica híbrida desenvolvida combina o raciocínio baseado em regras, com o raciocínio baseado em modelos para praticamente eliminar a principal deficiência do raciocínio baseado em regras: a dificuldade de manter a base de regras atualizada. Além disso, a nova técnica possibilita a criação de regras genéricas, permitindo o emprego de uma única regra para diagnosticar um problema em todos os equipamentos de um mesmo tipo.

A ferramenta desenvolvida, chamada *Smart One*, é a primeira versão do *Smart Alarms,* uma ferramenta que está sendo desenvolvida pela Universidade Federal de Campina Grande em parceria com a Companhia Hidro Elétrica do São Francisco (CHESF) e que será integrada ao sistema de gerenciamento de energia da CHESF para automatizar o diagnóstico de falhas em seus centros de controle.

Os objetivos principais do desenvolvimento do *Smart One* foram permitir a verificação da aplicabilidade de técnicas de correlação de eventos ao diagnóstico de falhas em redes elétricas e permitir validar o conhecimento extraído dos especialistas da CHESF sob a forma de regras de diagnóstico. Com o *Smart One,* foi possível testar a aplicabilidade da técnica híbrida de correlação de eventos ao diagnóstico de falhas em redes elétricas e validar o conhecimento contido nas regras para o diagnóstico de problemas em linhas de transmissão de 500kV, 230kV, 138kV, 69kV e 13.8kV.

Todo o conhecimento contido nas regras de diagnóstico foi obtido durante reuniões com especialistas em supervisão e controle de sistemas de geração e transmissão de energia elétrica da CHESF. Neste trabalho, como em qualquer outro que envolva o desenvolvimento

de um sistema especialista, o grande gargalo foi a aquisição do conhecimento. Vários fatores contribuíram para que essa tarefa fosse bastante árdua, entre eles, a pouca disponibilidade de tempo dos especialistas para as reuniões uma vez que eles continuaram desempenhando as suas atividades normais na CHESF, a grande quantidade de conhecimento necessário para o diagnóstico e a pouca experiência da equipe de análise e desenvolvimento com sistemas de potência. Mesmo assim, devido ao entusiasmo de toda a equipe envolvida no processo de aquisição de conhecimento, foi possível criar e validar praticamente todas as regras para o diagnóstico de falhas em linhas de transmissão.

Os resultados finais deste trabalho de Mestrado são uma técnica híbrida de correlação de eventos bastante eficiente para o diagnóstico de falhas em redes elétricas, uma base de regras para o diagnóstico de falhas em linhas de transmissão e uma ferramenta de diagnóstico de falhas capaz de diagnosticar uma falha nas linhas de transmissão do CROL (Centro Regional de Operação Leste, utilizado como piloto da aplicação) em menos de dois segundos e que pode ser utilizada para realizar treinamentos nos centros de controle.

Devido a presença de ruídos nos fluxos de eventos recebidos pelo sistema ainda não é possível aferir com precisão a taxa de acerto dos diagnósticos mas estão sendo realizados estudos para identificar a freqüência de geração dos ruídos, como parte do trabalho de mestrado de Eloi Rocha Neto, e com isso será possível obter uma taxa precisa de acertos do sistema.

No restante deste capítulo, são discutidos os resultados da satisfação dos requisitos (seção 6.1) e apresentadas propostas para trabalhos futuros (seção 6.2).

6.1 Satisfação dos Requisitos

Nesta seção, são reapresentados os requisitos levantados para a aplicação e discutidos aspectos de implementação relativos a sua satisfação. Todos os requisitos levantados para a aplicação foram satisfeitos.

6.1.1 Requisitos Funcionais

A seguir, são discutidos aspectos da implementação de cada um dos requisitos funcionais levantados na seção 4.1.1.

- **Eventos SAGE**

R. O sistema deve receber com entrada seqüências de eventos semelhantes às seqüências de eventos do SAGE.[9]

S. O *Smart One* recebe como entrada para o processamento **as mesmas** seqüências de eventos do SAGE.

R. Deve ser possível escolher e alternar entre as diversas seqüências de eventos disponíveis na base de dados histórica, possibilitando a simulação da monitoração da rede elétrica realizada em diferentes datas.

S. Esse requisito foi atendido pelo Navegador de Eventos (Figura 5.8) que permite a seleção de diferentes datas na base histórica de dados para a simulação da monitoração.

- **Velocidade de Simulação**

R. O sistema deve simular, em tempo real, a geração de eventos do SAGE e deve permitir que a velocidade de simulação seja alterada.

S. A interface principal do *Smart One* possui um *Slider* para ajustar a velocidade da simulação (Figura 6.1); a aceleração pode variar de 0, para tempo real, até 1000, para simulação sem intervalos entre os eventos. Seja qual for a aceleração utilizada, os resultados dos diagnósticos são sempre os mesmos.

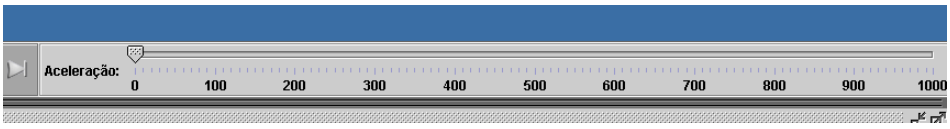

Figura 6.1: Slider para o ajuste da velocidade de simulação

- **Diagnóstico**

R. O sistema deve reconhecer todas as regras de linhas de transmissão da CHESF e realizar o diagnóstico dos alarmes;

S. O sistema reconhece todas as regras de linhas de transmissão da CHESF e possui testes de aceitação para apontar defeitos na aplicação caso alguma dessas regras deixe de ser reconhecida;

[9] R significa requisito e S significa satisfação do requisito

R. Deve ser fácil inserir novas regras de correlação no programa, as quais podem estar embutidas no código, mas deve haver uma maneira de isto ser feito sem que seja preciso realizar grandes modificações no programa;

S. As regras estão embutidas no código, sob a forma de classes Java. A criação de novas regras é realizada através da extensão de classes e implementação de métodos e pode ser realizada em poucos minutos;

R. As informações topológicas devem estar fora do código do programa;

S. O Apêndice A apresenta o arquivo de configuração que contém todas as informações topológicas utilizadas pelo sistema.

- **Momento da Análise**

R. Deve ser possível escolher em que momento a análise dos eventos deve ser realizada;

S. A interface principal do *Smart One* possui um *Slider* para ajustar os parâmetros de tempo do diagnóstico (Figura 6.2). Pode-se especificar a duração da janela de diagnóstico e a duração do *timeout* de inatividade.

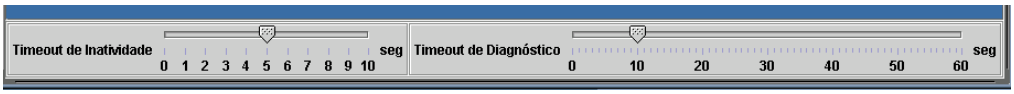

Figura 6.2: Slider para selecionar o timeout de inatividade e o timeout de diagnóstico

- **Log**

R. Deve ser possível definir pontos de *log* de informações em várias etapas do processo de tratamento de eventos;

S. O pacote log4j, do projeto Jakarta da Apache, foi utilizado para realizar o esquema de logs da aplicação.

- **Subconjuntos de Eventos**

R. Deve ser possível selecionar graficamente subconjuntos de uma seqüência de eventos para simulação;

S. A janela do gerador de eventos tem total suporte à seleção de subconjuntos dos eventos (Figura 6.3).

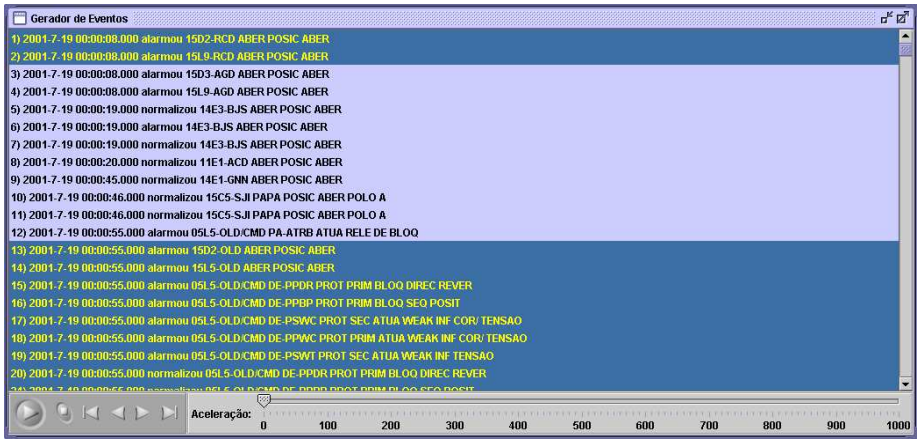

Figura 6.3: Janela do gerador de eventos

- **Interface padrão**

  R. O sistema deve apresentar uma saída não gráfica para o usuário;

  S. Foi criada uma interface inicial em modo MS-DOS para exibir as informações do sistema.

- **Interface gráfica**

  R. O sistema deve possuir uma interface gráfica onde estarão disponíveis todas as opções de configuração e que exiba o diagnóstico dos problemas.

  S. Foi criada uma interface gráfica utilizando componentes da biblioteca gráfica SWING da Sun para exibir os diagnósticos das falhas. A interface principal da aplicação pode ser observada na Figura 5.7.

- **Conversor de Topologia**

  R. O SAGE armazena as informações topológicas em arquivos com um formato proprietário. Deve existir uma forma de converter as informações topológicas contidas nesses arquivos para um formato compatível com o sistema;

  S. Esse componente foi desenvolvido pelo professor Jacques Philippe Sauvé e é capaz de extrair os parâmetros topológicos usados nas regras a partir dos arquivos de topologia do SAGE.

- **Navegador de Eventos**

R. O protótipo possui um banco de dados temporal, onde são armazenadas as seqüências de eventos obtidas com a monitoração da rede elétrica;

S. O navegador de eventos pode ser observado na Figura 5.8. Além de permitir selecionar diferentes datas na base histórica de dados, ele permite realizar uma série de buscas.

- **Editor de Cenários**

R. Deve existir um editor de cenários que permita modificar e criar seqüências de eventos para serem reproduzidas;

S. Esse componente foi desenvolvido pelo aluno Eloi Rocha Neto, do Mestrado em Informática desta mesma instituição, e possibilita a criação de novas seqüências de eventos para simulação. Sua interface pode ser observada na Figura 5.9.

6.1.2 Requisitos Não Funcionais

Na seção 4.1.2 foram apresentados os três requisitos não funcionais da aplicação:

R. O diagnóstico de um problema deve ser realizado em, no máximo, 100 ms, para que a duração do processamento possa ser imperceptível ao operador.

S. O tempo gasto para o diagnóstico de um único problema não pôde ser medido com precisão. Na máquina utilizada para os testes, o sistema leva, no máximo dois segundos para realizar todos os diagnósticos de uma janela de correlação. Na mesma janela, podem ser diagnosticados vários problemas distintos, dependendo da seqüência de eventos recebidos. O maior gasto de tempo está relacionado com o processamento da matriz de conectividade. O diagnóstico propriamente dito é realizado quase que instantaneamente. Pode-se considerar que o requisito não foi satisfeito por inteiro, ainda se precisa verificar se o tempo de dois segundos será satisfatório para o operador;

R. As telas do sistema devem emular o mais fielmente possível as telas do SAGE, incluindo *layout*, cor e *timestamp* de eventos, para permitir um melhor conforto visual para os operadores do sistema, acostumados com as telas do SAGE.

S. O *design* das telas do sistema, exibido na Figura 5.10, foi aprovado pelos operadores dos centros de controle e se assemelha bastante, em cor e distribuição de informações, com as telas do SAGE;

R. O sistema deve exibir as informações que seriam exibidas pelo SAGE e as informações obtidas com o diagnóstico, permitindo uma observação fácil dos resultados do processamento;

S. A interface principal do sistema possui três sub-janelas, uma para exibir o gerador de eventos, uma para exibir o emulador do SAGE e outra para exibir os diagnósticos do *Smart One*. Portanto é bastante simples comparar os diagnósticos realizados com as informações exibidas pelo SAGE.

6.2 Trabalhos Futuros

Considerando que este trabalho de mestrado é uma parte de um projeto de mais longo prazo, é natural que haja extensões da aplicação desenvolvida para que ela atinja o objetivo final do projeto, que é a implantação do sistema nos centros de controle da CHESF e a sua integração com o seu sistema de gerenciamento de energia.

As propostas de trabalhos futuros foram divididas em duas partes: as três primeiras dizem respeito a trabalhos que precisam ser realizados para que o objetivo final do projeto *Smart Alarms* seja atingido. As demais propostas concernem a trabalhos relacionados com o projeto, mas que não são fundamentais para que ele tenha êxito.

a) Criar e validar as demais regras de apoio ao diagnóstico

Até o momento, foram criadas e validadas apenas as regras para o diagnóstico de falhas em linhas de transmissão. Os sistemas elétricos possuem muitos outros equipamentos que podem falhar, tais como: transformadores, reatores, barramentos e compensadores. É preciso complementar a base de regras com a criação e validação das regras para estes outros equipamentos. Este trabalho já está sendo elaborado.

b) Processar os eventos para remover ruídos

Esta é uma questão bastante importante para o sucesso do projeto *Smart Alarms*. A técnica híbrida de diagnóstico processa um fluxo de eventos que ela considera perfeito, mas já foi detectado que existem ruídos neste fluxo de eventos. Considera-se como ruído o aparecimento de eventos espúrios que, na verdade, não ocorreram na rede elétrica, e a perda

de eventos que realmente aconteceram. Eventos atrasados podem ser considerados uma combinação de evento perdido e espúrio. É preciso fazer um estudo para avaliar a gravidade desses ruídos nos eventos e desenvolver técnicas para tratar dos ruídos antes ou durante a realização do diagnóstico. Uma alternativa é a de criar um filtro para limpar o fluxo de eventos. A Figura 6.4 ilustra a extensão da arquitetura do *Smart One* com a inserção do filtro de eventos. Para isso, é preciso criar um novo barramento para receber os eventos filtrados. Uma outra possibilidade para lidar com os ruídos na seqüência de eventos é a adoção de uma nova técnica que considere incertezas durante o diagnóstico, como a técnica de *codebooks* (KLIGER, 1995).

c) Integrar o SmartOne com o SAGE

O último passo antes de o sistema entrar em produção na CHESF é sua integração com o SAGE para que ele possa receber as seqüências de eventos do SAGE, realizar os diagnósticos e inseri-los de volta no sistema SAGE para apresentação ao operador. Um grande passo para a integração com o SAGE foi dado com a criação do *gateway* de comunicação do CEPEL. Com esse gateway, é possível estabelecer uma comunicação entre e nosso sistema e o SAGE sem interferir no escalonamento de tempo real deste último.

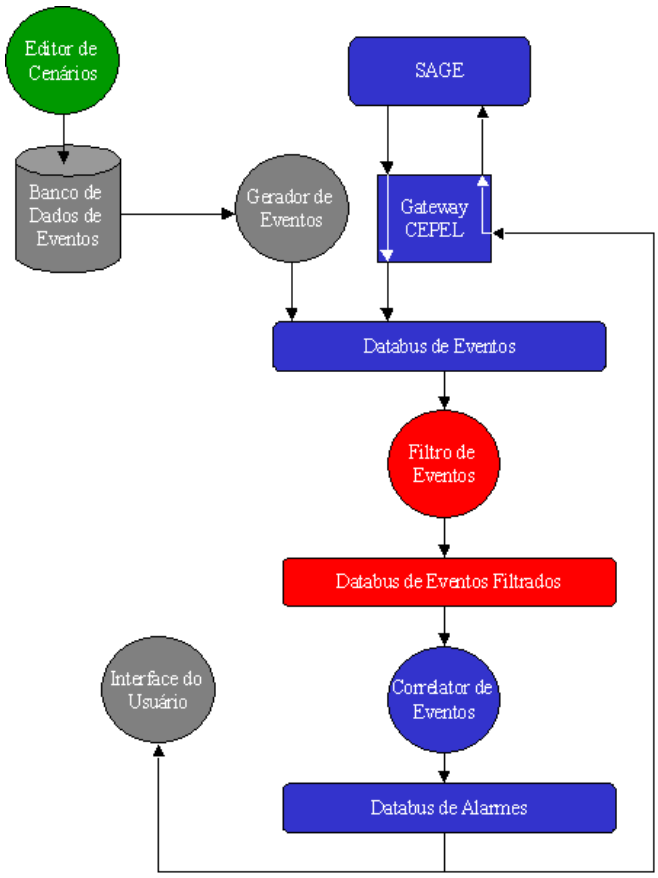

Figura 6.4: Arquitetura do Smart One estendida com o filtro de eventos e um novo databus

d) Módulo de explicação de diagnósticos

Um módulo adicional que pode ser inserido no *Smart One* é um módulo de explicação de diagnóstico, o qual seria o encarregado de explicar quais eventos levaram o motor de inferência a realizar um determinado diagnóstico. Esse módulo seria útil para dar uma maior confiança aos operadores dos centros de controle nos diagnósticos realizados pelo sistema uma vez que eles poderiam observar os motivos que levaram o sistema a chegar a uma dada conclusão.

e) Módulo de sugestão de soluções

Uma vez que os diagnósticos são realizados automaticamente, o próximo passo seria automatizar a solução dos problemas. Para isso, pode-se inserir no *Smart One* um módulo de sugestão de soluções, encarregado de sugerir, de acordo com o problema diagnosticado, a melhor solução (ou uma solução adequada) para normalizar a situação do sistema. Esse módulo envolveria uma grande quantidade de conhecimento dos especialistas da CHESF e um enorme trabalho para extrair e validar esse conhecimento. Desta forma, talvez merecesse ser tratado com mais destaque, possivelmente em um novo projeto de cooperação entre a CHESF e a UFCG.

f) Geração de relatórios

Um módulo que não foi previsto no projeto do *Smart One,* mas que talvez fosse bastante útil é um módulo de geração de relatórios. Esse módulo seria encarregado de gerar relatórios com estatísticas sobre as falhas diagnosticadas nos equipamentos. Com ele seria possível, por exemplo, descobrir qual a linha de transmissão da CHESF que mais falhou no último mês ou quais foram os disjuntores que mais mudaram de estado na última semana. Com essas informações estatísticas, seria possível otimizar o esquema de manutenção para tentar prevenir a ocorrência de falhas na rede elétrica.

g) Algoritmo de processamento da matriz de conectividade

O grande gargalo para que o tempo de diagnóstico seja de dois segundos é o processamento da matriz de conectividade. Apesar de o tempo de dois segundos ser aparentemente aceitável para a realização do diagnóstico, é possível reduzi-lo com a modificação do algoritmo de processamento da matriz. O algoritmo atual processa toda a matriz toda vez que um único disjuntor se abre. Pode-se estudar uma modificação no algoritmo para que ele processe apenas a parte da matriz que representa os equipamentos

afetados pela abertura de um disjuntor. Com essa alteração, deve ser possível diminuir o tempo de diagnóstico para bem menos que um segundo.

h) Estudar a aplicabilidade da técnica a redes de computadores e de telecomunicações

A última das sugestões para trabalhos futuros é a realização de um estudo da aplicabilidade da técnica híbrida desenvolvida aqui ao diagnóstico de falhas em redes de computadores e redes de telecomunicações.

Apêndice A

Este apêndice apresenta uma parte do arquivo de configuração do *Smart One* utilizado para armazenar as informações topológicas do sistema e que é gerado automáticamente pela ferramenta Topogiggio, criada pelo professor Jacques Philippe Sauvé.

Este apêndice ilustra o tipo de informação topológica utilizado pelo *Smart One,* é a partir das informações contidas nesse arquivo de configuração que o sistema cria o seu modelo da rede elétrica, um grafo bidirecional, e inicializa o estado de todos os equipamentos da rede elétrica.

A versão apresentada a seguir possui apenas as informações topológicas da subestação Açu II, uma das várias subestações do sistema leste da CHESF.

```
<SistemaEletrico id="CHESF">
<Regiao id="CROL">
<Subestacao id="ACD" nome="ACU II">
<BancoCapacitor id="01H1-ACD" conexoes="51H1-ACD"/>
<BancoCapacitor id="01H2-ACD" conexoes="51H2-ACD"/>
<BancoCapacitor id="01H3-ACD" conexoes="51H3-ACD"/>
<BancoCapacitor id="01H4-ACD" conexoes="51H4-ACD"/>
<Barramento id="01B1-ACD" conexoes="11T4-ACD,11T5-ACD,31Y1-4-ACD"/>
<Barramento id="01B2-ACD" conexoes="31E1-4-ACD,31H1-4-ACD,31H2-4-ACD,31H3-4-ACD,31H4-4-
ACD,31Y1-4-ACD"/>
<Barramento id="02BP-ACD" conexoes="32D1-1-ACD,12F1-ACD,12F2-ACD,12F3-ACD,12F4-ACD,12F5-
ACD,12T4-ACD,12T5-ACD"/>
<Barramento id="03BP-ACD" conexoes="13C2-ACD,33C2-6-ACD,13T2-ACD,33T2-6-ACD"/>
<Barramento id="04B1-ACD" conexoes="34D1-1-ACD,34L2-1-ACD,34T2-1-ACD,34T4-1-ACD,34T5-1-ACD"/>
<Barramento id="04B2-ACD" conexoes="34D1-2-ACD,34L2-2-ACD,34T2-2-ACD,34T4-2-ACD,34T5-2-ACD"/>
<Chave id="31E1-4-ACD" conexoes="01B2-ACD,31H1-4-ACD,31H2-4-ACD,31H3-4-ACD,31H4-4-ACD,31Y1-4-
ACD,11E1-ACD" estadoNormal="F"/>
<Chave id="31H1-4-ACD" conexoes="01B2-ACD,31E1-4-ACD,31H2-4-ACD,31H3-4-ACD,31H4-4-ACD,31Y1-4-
ACD,51H1-ACD" estadoNormal="F"/>
<Chave id="31H2-4-ACD" conexoes="01B2-ACD,31E1-4-ACD,31H1-4-ACD,31H3-4-ACD,31H4-4-ACD,31Y1-4-
ACD,51H2-ACD" estadoNormal="F"/>
<Chave id="31H3-4-ACD" conexoes="01B2-ACD,31E1-4-ACD,31H1-4-ACD,31H2-4-ACD,31H4-4-ACD,31Y1-4-
ACD,51H3-ACD" estadoNormal="F"/>
<Chave id="31H4-4-ACD" conexoes="01B2-ACD,31E1-4-ACD,31H1-4-ACD,31H2-4-ACD,31H3-4-ACD,31Y1-4-
ACD,51H4-ACD" estadoNormal="F"/>
<Chave id="31Y1-4-ACD" conexoes="01B1-ACD,11T4-ACD,11T5-ACD,01B2-ACD,31E1-4-ACD,31H1-4-
ACD,31H2-4-ACD,31H3-4-ACD,31H4-4-ACD" estadoNormal="F"/>
<Chave id="32D1-1-ACD" conexoes="02BP-ACD,12F1-ACD,12F2-ACD,12F3-ACD,12F4-ACD,12F5-ACD,12T4-
ACD,12T5-ACD,12D1-ACD" estadoNormal="F"/>
<Chave id="32D1-2-ACD" conexoes="32F1-6-ACD,32F2-6-ACD,32F3-6-ACD,32F4-6-ACD,32F5-6-ACD,32T4-
6-ACD,32T5-6-ACD,12D1-ACD" estadoNormal="F"/>
<Chave id="32F1-6-ACD" conexoes="32D1-2-ACD,32F2-6-ACD,32F3-6-ACD,32F4-6-ACD,32F5-6-ACD,32T4-
6-ACD,32T5-6-ACD,02F1-ACU/ACD,12F1-ACD" estadoNormal="A"/>
<Chave id="32F2-6-ACD" conexoes="32D1-2-ACD,32F1-6-ACD,32F3-6-ACD,32F4-6-ACD,32F5-6-ACD,32T4-
6-ACD,32T5-6-ACD,02F2-MCA/ACD,12F2-ACD" estadoNormal="A"/>
```

```
<Chave id="32F3-6-ACD" conexoes="32D1-2-ACD,32F1-6-ACD,32F2-6-ACD,32F4-6-ACD,32F5-6-ACD,32T4-
6-ACD,32T5-6-ACD,02F3-PNC/ACD,12F3-ACD" estadoNormal="A"/>
<Chave id="32F4-6-ACD" conexoes="32D1-2-ACD,32F1-6-ACD,32F2-6-ACD,32F3-6-ACD,32F5-6-ACD,32T4-
6-ACD,32T5-6-ACD,02F4-EST/ACD,12F4-ACD" estadoNormal="A"/>
<Chave id="32F5-6-ACD" conexoes="32D1-2-ACD,32F1-6-ACD,32F2-6-ACD,32F3-6-ACD,32F4-6-ACD,32T4-
6-ACD,32T5-6-ACD,02F5-MSU/ACD,12F5-ACD" estadoNormal="A"/>
<Chave id="32T4-6-ACD" conexoes="32D1-2-ACD,32F1-6-ACD,32F2-6-ACD,32F3-6-ACD,32F4-6-ACD,32F5-
6-ACD,32T5-6-ACD,04T4-S-ACD,12T4-ACD" estadoNormal="A"/>
<Chave id="32T5-6-ACD" conexoes="32D1-2-ACD,32F1-6-ACD,32F2-6-ACD,32F3-6-ACD,32F4-6-ACD,32F5-
6-ACD,32T4-6-ACD,04T5-S-ACD,12T5-ACD" estadoNormal="A"/>
<Chave id="33C2-6-ACD" conexoes="03BP-ACD,13C2-ACD,13T2-ACD,33T2-6-ACD,03C2-SMD/ACD,13C2-ACD"
estadoNormal="A"/>
<Chave id="33T2-6-ACD" conexoes="03BP-ACD,13C2-ACD,33C2-6-ACD,13T2-ACD,04T2-ACD,13T2-ACD"
estadoNormal="A"/>
<Chave id="34D1-1-ACD" conexoes="04B1-ACD,34L2-1-ACD,34T2-1-ACD,34T4-1-ACD,34T5-1-ACD,14D1-
ACD" estadoNormal="F"/>
<Chave id="34D1-2-ACD" conexoes="04B2-ACD,34L2-2-ACD,34T2-2-ACD,34T4-2-ACD,34T5-2-ACD,14D1-
ACD" estadoNormal="F"/>
<Chave id="34L2-1-ACD" conexoes="04B1-ACD,34D1-1-ACD,34T2-1-ACD,34T4-1-ACD,34T5-1-ACD,14L2-
ACD,34L2-2-ACD,34L2-6-ACD" estadoNormal="F"/>
<Chave id="34L2-2-ACD" conexoes="04B2-ACD,34D1-2-ACD,34T2-2-ACD,34T4-2-ACD,34T5-2-ACD,14L2-
ACD,34L2-1-ACD,34L2-6-ACD" estadoNormal="A"/>
<Chave id="34L2-6-ACD" conexoes="14L2-ACD,34L2-1-ACD,34L2-2-ACD,04L2-MSD/ACD,14L2-ACD"
estadoNormal="A"/>
<Chave id="34T2-1-ACD" conexoes="04B1-ACD,34D1-1-ACD,34L2-1-ACD,34T4-1-ACD,34T5-1-ACD,14T2-
ACD,34T2-2-ACD,34T2-6-ACD" estadoNormal="F"/>
<Chave id="34T2-2-ACD" conexoes="04B2-ACD,34D1-2-ACD,34L2-2-ACD,34T4-2-ACD,34T5-2-ACD,14T2-
ACD,34T2-1-ACD,34T2-6-ACD" estadoNormal="A"/>
<Chave id="34T2-6-ACD" conexoes="14T2-ACD,34T2-1-ACD,34T2-2-ACD,04T2-ACD,14T2-ACD"
estadoNormal="A"/>
<Chave id="34T4-1-ACD" conexoes="04B1-ACD,34D1-1-ACD,34L2-1-ACD,34T2-1-ACD,34T5-1-ACD,14T4-
ACD,34T4-2-ACD,34T4-6-ACD" estadoNormal="F"/>
<Chave id="34T4-2-ACD" conexoes="04B2-ACD,34D1-2-ACD,34L2-2-ACD,34T2-2-ACD,34T5-2-ACD,14T4-
ACD,34T4-1-ACD,34T4-6-ACD" estadoNormal="A"/>
<Chave id="34T4-6-ACD" conexoes="14T4-ACD,34T4-1-ACD,34T4-2-ACD,04T4-P-ACD,14T4-ACD"
estadoNormal="A"/>
<Chave id="34T5-1-ACD" conexoes="04B1-ACD,34D1-1-ACD,34L2-1-ACD,34T2-1-ACD,34T4-1-ACD,14T5-
ACD,34T5-2-ACD,34T5-6-ACD" estadoNormal="F"/>
<Chave id="34T5-2-ACD" conexoes="04B2-ACD,34D1-2-ACD,34L2-2-ACD,34T2-2-ACD,34T4-2-ACD,14T5-
ACD,34T5-1-ACD,34T5-6-ACD" estadoNormal="A"/>
<Chave id="34T5-6-ACD" conexoes="14T5-ACD,34T5-1-ACD,34T5-2-ACD,04T5-P-ACD,14T5-ACD"
estadoNormal="A"/>
<Chave id="51H1-ACD" conexoes="31H1-4-ACD,01H1-ACD" estadoNormal="F"/>
<Chave id="51H2-ACD" conexoes="31H2-4-ACD,01H2-ACD" estadoNormal="F"/>
<Chave id="51H3-ACD" conexoes="31H3-4-ACD,01H3-ACD" estadoNormal="F"/>
<Chave id="51H4-ACD" conexoes="31H4-4-ACD,01H4-ACD" estadoNormal="F"/>
<Disjuntor id="11E1-ACD" conexoes="31E1-4-ACD,01E1-ACD" estadoNormal="A"/>
<Disjuntor id="11T4-ACD" conexoes="01B1-ACD,11T5-ACD,31Y1-4-ACD,04T4-T-ACD" estadoNormal="F"/>
<Disjuntor id="11T5-ACD" conexoes="01B1-ACD,11T4-ACD,31Y1-4-ACD,04T5-T-ACD" estadoNormal="A"/>
<Disjuntor id="12D1-ACD" conexoes="32D1-1-ACD,32D1-2-ACD" estadoNormal="A"/>
<Disjuntor id="12F1-ACD" conexoes="02BP-ACD,32D1-1-ACD,12F2-ACD,12F3-ACD,12F4-ACD,12F5-
ACD,12T4-ACD,12T5-ACD,02F1-ACU/ACD,32F1-6-ACD" estadoNormal="F"/>
<Disjuntor id="12F2-ACD" conexoes="02BP-ACD,32D1-1-ACD,12F1-ACD,12F3-ACD,12F4-ACD,12F5-
ACD,12T4-ACD,12T5-ACD,02F2-MCA/ACD,32F2-6-ACD" estadoNormal="F"/>
<Disjuntor id="12F3-ACD" conexoes="02BP-ACD,32D1-1-ACD,12F1-ACD,12F2-ACD,12F4-ACD,12F5-
ACD,12T4-ACD,12T5-ACD,02F3-PNC/ACD,32F3-6-ACD" estadoNormal="F"/>
<Disjuntor id="12F4-ACD" conexoes="02BP-ACD,32D1-1-ACD,12F1-ACD,12F2-ACD,12F3-ACD,12F5-
ACD,12T4-ACD,12T5-ACD,02F4-EST/ACD,32F4-6-ACD" estadoNormal="F"/>
<Disjuntor id="12F5-ACD" conexoes="02BP-ACD,32D1-1-ACD,12F1-ACD,12F2-ACD,12F3-ACD,12F4-
ACD,12T4-ACD,12T5-ACD,02F5-MSU/ACD,32F5-6-ACD" estadoNormal="F"/>
<Disjuntor id="12T4-ACD" conexoes="02BP-ACD,32D1-1-ACD,12F1-ACD,12F2-ACD,12F3-ACD,12F4-
ACD,12F5-ACD,12T5-ACD,04T4-S-ACD,32T4-6-ACD" estadoNormal="F"/>
<Disjuntor id="12T5-ACD" conexoes="02BP-ACD,32D1-1-ACD,12F1-ACD,12F2-ACD,12F3-ACD,12F4-
ACD,12F5-ACD,12T4-ACD,04T5-S-ACD,32T5-6-ACD" estadoNormal="F"/>
<Disjuntor id="13C2-ACD" conexoes="03BP-ACD,33C2-6-ACD,13T2-ACD,33T2-6-ACD,03C2-SMD/ACD,33C2-
6-ACD" estadoNormal="F"/>
<Disjuntor id="13T2-ACD" conexoes="03BP-ACD,13C2-ACD,33C2-6-ACD,33T2-6-ACD,04T2-ACD,33T2-6-
ACD" estadoNormal="F"/>
<Disjuntor id="14D1-ACD" conexoes="34D1-1-ACD,34D1-2-ACD" estadoNormal="A"/>
<Disjuntor id="14L2-ACD" conexoes="34L2-1-ACD,34L2-2-ACD,34L2-6-ACD,04L2-MSD/ACD,34L2-6-ACD"
estadoNormal="F"/>
<Disjuntor id="14T2-ACD" conexoes="34T2-1-ACD,34T2-2-ACD,34T2-6-ACD,04T2-ACD,34T2-6-ACD"
estadoNormal="F"/>
<Disjuntor id="14T4-ACD" conexoes="34T4-1-ACD,34T4-2-ACD,34T4-6-ACD,04T4-P-ACD,34T4-6-ACD"
estadoNormal="F"/>
<Disjuntor id="14T5-ACD" conexoes="34T5-1-ACD,34T5-2-ACD,34T5-6-ACD,04T5-P-ACD,34T5-6-ACD"
estadoNormal="F"/>
```

```
<Linha69KV id="02F1-ACU/ACD" conexoes="12F1-ACD,32F1-6-ACD" lado="DE">
<DisjuntorLinha>12F1-ACD</DisjuntorLinha>
<Barramento1>02BP-ACD</Barramento1>
<DisjuntorTransferencia1>12D1-ACD</DisjuntorTransferencia1>
</Linha69KV>
<Linha69KV id="02F2-MCA/ACD" conexoes="12F2-ACD,32F2-6-ACD" lado="DE">
<DisjuntorLinha>12F2-ACD</DisjuntorLinha>
<Barramento1>02BP-ACD</Barramento1>
<DisjuntorTransferencia1>12D1-ACD</DisjuntorTransferencia1>
</Linha69KV>
<Linha69KV id="02F3-PNC/ACD" conexoes="12F3-ACD,32F3-6-ACD" lado="DE">
<DisjuntorLinha>12F3-ACD</DisjuntorLinha>
<Barramento1>02BP-ACD</Barramento1>
<DisjuntorTransferencia1>12D1-ACD</DisjuntorTransferencia1>
</Linha69KV>
<Linha69KV id="02F4-EST/ACD" conexoes="12F4-ACD,32F4-6-ACD" lado="DE">
<DisjuntorLinha>12F4-ACD</DisjuntorLinha>
<Barramento1>02BP-ACD</Barramento1>
<DisjuntorTransferencia1>12D1-ACD</DisjuntorTransferencia1>
</Linha69KV>
<Linha69KV id="02F5-MSU/ACD" conexoes="12F5-ACD,32F5-6-ACD" lado="DE">
<DisjuntorLinha>12F5-ACD</DisjuntorLinha>
<Barramento1>02BP-ACD</Barramento1>
<DisjuntorTransferencia1>12D1-ACD</DisjuntorTransferencia1>
</Linha69KV>
<Linha138KV id="03C2-SMD/ACD" conexoes="13C2-ACD,33C2-6-ACD" lado="PARA">
<DisjuntorLinha>13C2-ACD</DisjuntorLinha>
<Barramento1>03BP-ACD</Barramento1>
</Linha138KV>
<Linha230KV id="04L2-MSD/ACD" conexoes="14L2-ACD,34L2-6-ACD" lado="PARA">
<DisjuntorLinha>14L2-ACD</DisjuntorLinha>
<Barramento1>04B1-ACD</Barramento1>
<Barramento2>04B2-ACD</Barramento2>
<DisjuntorTransferencia1>14D1-ACD</DisjuntorTransferencia1>
</Linha230KV>
<Reator id="01E1-ACD" conexoes="11E1-ACD"/>
<Transformador id="04T2-ACD" conexoes="14T2-ACD,34T2-6-ACD,13T2-ACD,33T2-6-ACD"/>
<Transformador id="04T4-P-ACD" conexoes="14T4-ACD,34T4-6-ACD,04T4-S-ACD,04T4-T-ACD"/>
<Transformador id="04T4-S-ACD" conexoes="12T4-ACD,32T4-6-ACD,04T4-P-ACD,04T4-T-ACD"/>
<Transformador id="04T4-T-ACD" conexoes="11T4-ACD,04T4-P-ACD,04T4-S-ACD"/>
<Transformador id="04T5-P-ACD" conexoes="14T5-ACD,34T5-6-ACD,04T5-S-ACD,04T5-T-ACD"/>
<Transformador id="04T5-S-ACD" conexoes="12T5-ACD,32T5-6-ACD,04T5-P-ACD,04T5-T-ACD"/>
<Transformador id="04T5-T-ACD" conexoes="11T5-ACD,04T5-P-ACD,04T5-S-ACD"/>
</Subestacao>
</Regiao>
</SistemaEletrico>
```

Apêndice B

Neste apêndice são apresentadas as regras de diagnóstico criadas e validadas utilizando o *Smart On*e. Foram criadas regras para diagnóstico de problemas em linhas de 13.8, 69, 138, 230 e 500kV, as principais tensões utilizadas na CHESF.

Este apêndice serve como referência para a implementação e modificação das regras do *Smart One* e como base para a confeção dos próximos relatórios com o levantamento das regras de diagnóstico para os demais equipamentos da CHESF.

O documento com as regras foi gerado pelo professor Jorge César Abrantes de Figueiredo, uma dos integrantes do projeto *Smart Alarms* e atual responsável pela aquisição do conhecimento junto aos especialistas da CHESF.

Regras de Linhas de Transmissão do Sistema Leste (13.8KV)

Topologia

Linhas de 13.8KV:
SE's:
Arranjos:
- 2 barras + disjuntor de transferência
- Barra principal
- Barra principal + Chave bypass

Dispositivos:
- Disjuntor de linha: Dj1
- Chaves do disjuntor de linha: Dj1-1, Dj1-2, Dj1-4, Dj1-5 e Dj1-6
- Disjuntor de transferência: DjT1
- Chaves do disjuntor de transferência: DjT1-1 e DjT1-2
- Chave bypass: Dj1-6

Macro EXISTE(parâmetro)
Retorna TRUE ou FALSE, indicando se o parâmetro existe.

Macro !CONECTADO(linha, subest)
<< Arranjo 1: Duas barras + disjuntor de transferência>>
((CV(subest, Dj1-1) & CV(subest, Dj1-2)) |
((DJ(subest, Dj1)|CV(subest, Dj1-4)|CV(subest, Dj1-5))& CV(subest,Dj1-6)) &
((DJ(subest, Djt1)| CV(subest, Djt1-1) | CV(subest, Djt1-2)|
CV(subest, B1-2)) & ! CV(subest, Dj1-6) & DJ(subest, Dj1)))

```
<< Arranjo 2: Barra principal>>
(DJ(subest, Dj1) | CV(subest, Dj1-4) | CV(subest, Dj1-5))

<< Arranjo 3: Barra principal + Chave bypass>>
(((DJ(subest, Dj1)|CV(subest, Dj1-4)|CV(subest, Dj1-5))&CV(subest, Dj1-6)))
```

Macro DESARMETOTAL(linha, de, para)

```
!CONECTADO(linha, de) & !CONECTADO(linha, para) &
MW(de, ↓0) & MW(para, ↓0) &
MVAR(de, ↓0) & MVAR(para, ↓0)
```

Macro DESLIGAMENTOTOTAL(linha, de, para)

```
DESARMETOTAL(linha, de, para) &
!ATPR(de) &
!ATPR(para)
```

Macro DESARMEPARCIALDE(linha, de, para)

```
!CONECTADO(linha, de) & CONECTADO(linha, para) &
MW(de, ↓0) & MW(para, ↓0) &
MVAR(de, ↓0) & MVAR(para, !0)
```

Macro DESLIGAMENTOPARCIALDE(linha, de, para)

```
DESARMEPARCIALDE(linha, de, para) &
!ATPR(de)
```

Macro DESARMEPARCIALPARA(linha, de, para)

```
CONECTADO(linha, de) & !CONECTADO(linha, para) &
MW(de, ↓0) & MW(para, ↓0) &
MVAR(de, !0) & MVAR(para, ↓0)
```

Macro DESLIGAMENTOPARCIALPARA(linha, de, para)

```
DESARMEPARCIALPARA(linha, de, para) &
!ATPR(para)
```

R1. Desligamento LT Concessionária
Condição de existência: A regra existe se lado para for CI

```
!CONECTADO(linha, de) &
!ATPR(de) &
MW(de, ↓0) &
MVAR(de, ↓0)
```

R2. Desarme LT Concessionária com atuação da proteção
Condição de existência: A regra existe se lado para for CI

```
!CONECTADO(linha, de) &
MW(de, ↓0) &
MVAR(de, ↓0) &
ATPR(de)
```

Regras de Linhas de Transmissão do Sistema Leste (69KV)

Topologia

Linhas de 69KV:
SE's: STD, CRD, ACD, SMD, CGD, CGU, BVT, TAC, RLD, PEN, GNN, BGI, AGL, RIB, MRR, PRD, MRD, NTD, RCD, MSI,

Arranjos:

- 2 barras + disjuntor de transferência (90%)

- Barra principal + chave bypass
- Barra principal + chave de barra
- CGU (disjuntor transferido ou não)

Dispositivos:

- Disjuntor de linha: Dj1
- Chaves do disjuntor de linha: Dj1-1, Dj1-2, Dj1-4, Dj1-5 e Dj1-6
- Disjuntor de transferência: DjT1
- Chaves do disjuntor de transferência: DjT1-1 e DjT1-2
- Disjuntor de transferência (segundo): DjT2
- Chaves do disjuntor de transferência (segundo): DjT2-1 e DjT2-2
- Chave bypass: Dj1-6
- Barra: B1
- Chaves da Barra: B1-1 e B1-2

Macro EXISTE(parâmetro)
Retorna TRUE ou FALSE, indicando se o parâmetro existe.

Macro !CONECTADO(linha, subest)

```
<< Arranjo 1: Duas barras + disjuntor de transferência>>
((CV(subest, Dj1-1) & CV(subest, Dj1-2)) |
((DJ(subest, Dj1)|CV(subest, Dj1-4)|CV(subest, Dj1-5))&CV(subest, Dj1-6)) &
((DJ(subest, Djt1)| CV(subest, Djt1-1) | CV(subest, Djt1-2)|
CV(subest, B1-2)) & ! CV(subest, Dj1-6) & DJ(subest, Dj1))

<< Arranjo 2: Duas barras + dois disjuntores de transferência>>
((CV(subest, Dj1-1) & CV(subest, Dj1-2)) |
((DJ(subest,Dj1)|CV(lsubest,Dj1-4)|CV(subest, Dj1-5))&CV(subest,Dj1-6))&
((DJ(subest, Djt1)| CV(subest, Djt1-1) | CV(subest, Djt1-2) |
CV(subest, B1-2)) & ! CV(subest, Dj1-6) & DJ(subest, Dj1)) &
((DJ(subest, Djt2)| CV(subest, Djt2-1) | CV(subest, Djt2-2)|
CV(subest, B1-1)) & ! CV(subest, Dj1-6) & DJ(subest, Dj1))

<< Arranjo 3: Barra principal + bypass>>
(((DJ(subest, Dj1)|CV(lsubest, Dj1-4)|CV(subest, Dj1-5))&CV(subest,Dj1-6))

<< Arranjo 4: Barra principal + chave de barra>>
(((DJ(subest, Dj1) | CV(lsubest, Dj1-4) | CV(subest, Dj1-5)) &
CV(subest, B1-1) &
CV(subest, B1-2)) )

<< Arranjo 5: Barra principal + disjuntor de transferência>>
(((DJ(subest, Dj1)|CV(subest, Dj1-4)|CV(subest, Dj1-5))&CV(subest,Dj1-6)) &
(((DJ(subest, Djt1)| CV(subest, Djt1-1) |
CV(subest, Djt1-2)) & ! CV(subest, Dj1-6) & DJ(subest, Dj1)) |
!DJ(subset, Djt1))
```

Macro DESARMETOTAL(linha, de, para)

```
!CONECTADO(linha, de) & !CONECTADO(linha, para) &
MW(de, ↓0) & MW(para, ↓0) &
MVAR(de, ↓0) & MVAR(para, ↓0)
```

Macro DESLIGAMENTOTOTAL(linha, de, para)

```
DESARMETOTAL(linha, de, para) &
!ATPR(de) &
!ATPR(para)
```

Macro DESARMEPARCIALDE(linha, de, para)

```
!CONECTADO(linha, de) & CONECTADO(linha, para) &
MW(de, ↓0) & MW(para, ↓0) &
MVAR(de, ↓0) & MVAR(para, !0)
```

Macro DESLIGAMENTOPARCIALDE(linha, de, para)

```
DESARMEPARCIALDE(linha, de, para) &
!ATPR(de)
```

Macro DESARMEPARCIALPARA(linha, de, para)

```
CONECTADO(linha,de) & !CONECTADO(linha, para) &
MW(de, ↓0) & MW(para, ↓0) &
MVAR(de, !0) & MVAR(para, ↓0)
```

Macro DESLIGAMENTOPARCIALPARA(linha, de, para)

```
DESARMEPARCIALPARA(linha, de, para) &
!ATPR(para)
```

R1. Desligamento LT Concessionária
Condição de existência: A regra existe se lado para for CI

```
!CONECTADO(linha, de) &
!ATPR(de) &
MW(de, ↓0) &
MVAR(de, ↓0)
```

R21. Desarme LT Concessionária com atuação da proteção
Condição de existência: A regra existe se lado para for CI

```
!CONECTADO(linha, de) &
MW(de, ↓0) &
MVAR(de, ↓0) &
ATPR(de)
```

Regras de Linhas de Transmissão do Sistema Leste (138KV)

Topologia

Linhas de 138KV:

- SE's: CGD, STD, ACD, CRD, SMD
- Arranjos:
  - 2 barras + disjuntor de transferência (CGD)
  - Barra principal + chave de linha (SMD)
  - Barra Simples (CRD)
  - Barra Simples + Chave bypass (STD e ACD)
- Dispositivos:
  - Primeiro disjuntor: Dj1
  - Chaves do primeiro disjuntor: Dj1-1, Dj1-2, Dj1-4, Dj1-5 e Dj1-6
  - Disjuntor de transferência: DjT1
  - Chaves do disjuntor de transferência: DjT1-1 e DjT1-2
  - Chave de linha: Linha-8

Macro EXISTE(parâmetro)

Retorna TRUE ou FALSE, indicando se o parâmetro existe.

Macro !CONECTADO(linha, parâmetro)

```
<< Arranjo 1: Duas barras + disjuntor de transferência>>
((CV(subest, Dj1-1) & CV(subest, Dj1-2)) |
((DJ(subest, Dj1)|CV(subest, Dj1-4) |CV(subest, Dj1-5))&CV(subest, Dj1-6))&
((DJ(subest, Djt1)| CV(subest, Djt1-1) | CV(subest, Djt1-2)|
CV(subest, B1-2)) & ! CV(subest, Dj1-6) & DJ(subest, Dj1))
```

```
<< Arranjo 2: Barra Principal + Chave de linha>>
(((DJ(subest,Dj1)|CV(subest,Dj1-4)|CV(subest, Dj1-5))&CV(subest, linha-8))
```

```
<< Arranjo 3: Barra principal>>
(DJ(subest, Dj1) | CV(subest, Dj1-4) | CV(subest, Dj1-5))

<< Arranjo 4: Barra principal + Chave bypass>>
(((DJ(subest, Dj1)|CV(subest, Dj1-4)|CV(subest, Dj1-5))&CV(subest, Dj1-6)))
```

Macro DESARMETOTAL(linha, de, para)

```
!CONECTADO(linha, de)   & !CONECTADO(linha, para)     &
KV(de,↓0)               & KV(para, ↓0)                &
MW(de, ↓0)              & MW(para, ↓0)                &
MVAR(de, ↓0)            & MVAR(para, ↓0)
```

Macro DESLIGAMENTOTOTAL(linha, de, para)

```
DESARMETOTAL(linha, de, para) &
!ATPR(de) &
!ATPR(para)
```

Macro DESARMEPARCIALDE(linha, de, para)

```
!CONECTADO(linha, de)   & CONECTADO(linha, para)      &
KV(de, !0)              & KV(para, !0)                &
MW(de, ↓0)              & MW(para, ↓0)                &
MVAR(de, ↓0)                  & MVAR(para, !0)
```

Macro DESLIGAMENTOPARCIALDE(linha, de, para)

```
DESARMEPARCIALDE(linha, de, para) &
!ATPR(de)
```

Macro DESARMEPARCIALPARA(linha, de, para)

```
CONECTADO(linha, de)    & !CONECTADO(linha,para)      &
KV(de, !0)              & KV(para, !0)                &
MW(de, ↓0)              & MW(para, ↓0)                &
MVAR(de, !0)                  & MVAR(para, ↓0)
```

Macro DESLIGAMENTOPARCIALPARA(linha, de, para)

```
DESARMEPARCIALPARA(linha, de, para) &
!ATPR(para)
```

R1. Desligamento Total de LT - Manual

Condição de existência: **Existe(para)**
DESLIGAMENTOTOTAL(linha, de, para)

R2. Desligamento parcial de LT lado DE - Manual

Condição de existência: A regra sempre existe.
DESLIGAMENTOPARCIALDE(linha, de, para)

R3. Desligamento parcial de LT lado PARA - Manual

Condição de existência: Existe(para)
DESLIGAMENTOPARCIALPARA(linha, de, para)

R4. Desarme parcial de LT lado DE com atuação com bloqueio

Condição de existência: Existe(Relé de Bloqueio).

```
DESARMEPARCIALDE(linha, de, para)   &
ATPR(de)          & !(ATPR(para))   &
RBLOQUEIO(de)     & !(RBLOQUEIO(para))
```

R5. Desarme parcial de LT lado PARA com atuação com bloqueio

Condição de existência: Existe(Relé de Bloqueio)

```
DESARMEPARCIALPARA(linha, de, para)       &
ATPR(para)  & !(ATPR(de))           &
RBLOQUEIO(para) & !(RBLOQUEIO(de))
```

Regras de Linhas de Transmissão do Sistema Leste (230KV)

Topologia
Linhas de 230KV:
SE's: RCD, MSI, ACD, TAC, RLD, PEN, GNN, BGI, AGL, RIB, MRR, PRD, MRD, NTD, CGD
Arranjos:
- 2 barras + disjuntor de transferência
- Barra principal + disjuntor de transferência
- RCD e AGL têm 2 disjuntores de transferência

Dispositivos:
- Disjuntor de linha: Dj1
- Chaves do disjuntor de linha: Dj1-1, Dj1-2, Dj1-4, Dj1-5 e Dj1-6
- Disjuntor de transferência: DjT1
- Chaves do disjuntor de transferência: DjT1-1 e DjT1-2
- Disjuntor de transferência (segundo): DjT2
- Chaves do disjuntor de transferência (segundo): DjT2-1 e DjT2-2
- Barra 1 (principal): B1
- Barra 2 (auxiliar): B2
- Chaves da Barra 1: B1-1 e B1-2
- Chaves da Barra 2: B2-1 e B2-2

Macro EXISTE(parâmetro)
Retorna TRUE ou FALSE, indicando se o parâmetro existe.

Macro !CONECTADO(linha, parâmetro)
Parâmetro pode ser subestação ou barra.

```
<< Arranjo 1: Duas barras + disjuntor de transferência>>
((CV(subest, Dj1-1) & CV(subest, Dj1-2)) |
((DJ(subest, Dj1)|CV(subest, Dj1-4)|CV(subest, Dj1-5))&CV(subest, Dj1-6)) &
((DJ(subest, Djt1)| CV(subest, Djt1-1) | CV(subest, Djt1-2)|
CV(subest, B1-2)) & ! CV(subest, Dj1-6) & DJ(subest, Dj1))
```

```
<< Arranjo 2: Duas barras + dois disjuntores de transferência>>
((CV(subest, Dj1-1) & CV(subest, Dj1-2)) |
((DJ(subest, Dj1)|CV(lsubest, Dj1-4)|CV(subest, Dj1-5))&CV(subest, Dj1-6))&
((DJ(subest, Djt1)| CV(subest, Djt1-1) | CV(subest, Djt1-2) |
CV(subest, B1-2)) & ! CV(subest, Dj1-6) & DJ(subest, Dj1)) &
((DJ(subest, Djt2)| CV(subest, Djt2-1) | CV(subest, Djt2-2)|
CV(subest, B1-1)) & ! CV(subest, Dj1-6) & DJ(subest, Dj1))
```

```
<< Arranjo 3: Barra principal + disjuntor de transferência>>
(((DJ(subest, Dj1)|CV(subest, Dj1-4)|CV(subest, Dj1-5))&CV(subest, Dj1-6))&
((DJ(subest, Djt1)| CV(subest, Djt1-1) |
CV(subest, Djt1-2)) & ! CV(subest, Dj1-6) & DJ(subest, Dj1))
```

Macro DESARMETOTAL(linha, de, para)

```
!CONECTADO(linha, de)   & !CONECTADO(linha, para)     &
KV(de,↓0)                     & KV(para, ↓0)                &
MW(de, ↓0)              & MW(para, ↓0)                &
MVAR(de, ↓0)            & MVAR(para, ↓0)
```

Macro DESLIGAMENTOTOTAL(linha, de, para)

```
DESARMETOTAL(linha, de, para)       &
!ATPR(de)                                  &
!ATPR(para)
```

Macro DESARMEPARCIALDE(linha, de, para)

```
!CONECTADO(linha, de)   & CONECTADO(linha, para)      &
KV(de, !0)              & KV(para, !0)                &
MW(de, ↓0)              & MW(para, ↓0)                &
MVAR(de, ↓0)                  & MVAR(para, !0)
```

Macro DESLIGAMENTOPARCIALDE(linha, de, para)

```
DESARMEPARCIALDE(linha, de, para) &
!ATPR(de)
```

Macro DESARMEPARCIALPARA(linha, de, para)

```
CONECTADO(linha, de)    & !CONECTADO(linha,para)      &
KV(de, !0)              & KV(para, !0)                &
MW(de, ↓0)              & MW(para, ↓0)                &
MVAR(de, !0)                  & MVAR(para, ↓0)
```

Macro DESLIGAMENTOPARCIALPARA(linha, de, para)

```
DESARMEPARCIALPARA(linha, de, para) &
!ATPR(para)
```

R1. Desligamento Total de LT - Manual

Condição de existência: Existe(para)

```
DESLIGAMENTOTOTAL(linha, de, para)
```

R2. Desarme total de linha com atuação da proteção sem religamento

Condição de existência: **Existe(para)**

```
DESARMETOTAL(linha, de, para)                         &
ATPR(de)                & ATPR(para)                   &
!RBLOQUEIO(de)          & !RBLOQUEIO(para)            &
![SOBTENTEMP(de)]       & ![SOBTENTEMP(para)]         &
![SOBTENINST(de)]       & ![SOBTENINST(para)]         &
!FALHADJ(de, dj1) & !FALHADJ(para, dj1)                        &
!((RELIGEFETDJ(de, dj1) & DJ(de, dj1) & !RELIGBLOQDJ(de, dj1))     &
!((RELIGEFETDJ(para, dj1) & DJ(para, dj1) & !RELIGBLOQDJ(para, dj1))
```

R3. Desarme total de linha com atuação da proteção sem religamento lado DE com religamento sem sucesso lado PARA

Condição de existência: Existe(para)

```
DESARMETOTAL(linha, de, para)                 &
ATPR(de)                & ATPR(para)           &
!RBLOQUEIO(de)          & !RBLOQUEIO(para)    &
![SOBTENTEMP(de)]       & ![SOBTENTEMP(para)]                  &
![SOBTENINST(de)]       & ![SOBTENINST(para)]                  &
!FALHADJ(de, dj1) & !FALHADJ(para, dj1)                  &
!((RELIGEFETDJ(de, dj1) & DJ(de, dj1) & !RELIGBLOQDJ(de, dj1))     &
((RELIGEFETDJ(para, dj1) & !DJ(para, dj1) & !RELIGBLOQDJ(para, dj1))
```

R4. Desarme total de linha com atuação da proteção sem religamento lado PARA com religamento sem sucesso lado DE

Condição de existência: **Existe(para)**

```
DESARMETOTAL(linha, de, para)                 &
ATPR(de)                & ATPR(para)           &
!RBLOQUEIO(de)          & !RBLOQUEIO(para)    &
![SOBTENTEMP(de)]       & ![SOBTENTEMP(para)]                  &
![SOBTENINST(de)]       & ![SOBTENINST(para)]                  &
!FALHADJ(de, dj1) & !FALHADJ(para, dj1)                  &
((RELIGEFETDJ(de, dj1) & DJ(de, dj1) & !RELIGBLOQDJ(de, dj1))     &
    !((RELIGEFETDJ(para, dj1) & !DJ(para, dj1) & !RELIGBLOQDJ(para, dj1))
```

R5. Desarme total de linha com atuação da proteção com bloqueio por sobretensão

Condição de existência: Existe(para)

DESARMETOTAL(linha, de, para) &
(ATPR(de) | ATPR(para)) &
(RBLOQUEIO(de) | RBLOQUEIO(para)) &
(SOBTENTEMP(de) | SOBTENINST(de)) &
(SOBTENTEMP(para) | SOBTENINST(para))

R6. Desarme total de linha com atuação da proteção com bloqueio por sobretensão lado DE

Condição de existência: Existe(para)

DESARMETOTAL(linha, de, para) &
(ATPR(de) | ATPR(para)) &
(RBLOQUEIO(de) | RBLOQUEIO(para)) &
(SOBTENTEMP(de) | SOBTENINST(de)) &
!SOBTENTEMP(para) &
!SOBTENINST(para)

R7. Desarme total de linha com atuação da proteção com bloqueio por sobretensão lado PARA

Condição de existência: Existe(para)

DESARMETOTAL(linha, de, para) &
(ATPR(de) | ATPR(para)) &
(RBLOQUEIO(de) | RBLOQUEIO(para)) &
(SOBTENTEMP(para) | SOBTENINST(para)) &
!SOBTENTEMP(de) &
!SOBTENINST(de)

R8. Desarme total de linha com atuação da proteção com bloqueio por falha do disjuntor lado DE

Condição de existência: Existe(para)

!CONECTADO(linha, para) &
(!CONECTADO(linha, (de, B1)) | FALHADJ(de, Dj1)) &
(!CONECTADO(linha, (de, B2)) | FALHADJ(de, Djt1)) &
(ATPR(de) | ATPR(para)) &
(RBLOQUEIO(de) | RBLOQUEIO(para)) &
(FALHADJ(de, dj_1) | FALHADJ(de, djt1) | [FALHADJ(de, djt2)])

R9. Desarme total de linha com atuação da proteção com bloqueio por falha do disjuntor lado PARA

Condição de existência: Existe(para)

!CONECTADO(linha, de) &
(!CONECTADO(linha, (para, B1)) | FALHADJ(para, Dj1)) &
(!CONECTADO(linha, (para, B2)) | FALHADJ(para, Djt1)) &
(ATPR(de) | ATPR(para)) &
(RBLOQUEIO(de) | RBLOQUEIO(para)) &
(FALHADJ(para, dj_1) | FALHADJ(para, djt1) | [FALHADJ(para, djt2)])

R10. Desarme total de linha com atuação da proteção com bloqueio por falha do disjuntor de reator (lado DE)

Condição de existência: Existe(para) & Existe(de, dj_r) & Existe(de, reator)

DESARMETOTAL(linha, de, para) &
FALHADJ(de, dj_r) &
(ATPR(de) | ATPR(para)) &

(RBLOQUEIO(de) | RBLOQUEIO(para)) &
TRIP86(de, reator) &
(ATUAFASE(de, A, reator) |
ATUAFASE(de, B, reator) |
ATUAFASE(de, C, reator) |
ATUAFASEPARALELO(de, reator) |
SOBTMPOLEO(de, A, reator) |
SOBTMPOLEO(de, B, reator) |
SOBTMPOLEO(de, C, reator) |
SOBTMPENROL(de, A, reator) |
SOBTMPENROL(de, B, reator) |
SOBTMPENROL(de, C, reator) |
GAS(de, A, reator) |
GAS(de, B, reator) |
GAS(de, C, reator) |
VALVSEG(de, A, reator) |
VALVSEG(de, B, reator) |
VALVSEG(de, C, reator) |
SOBCOR(de, A, reator) |
SOBCOR(de, B, reator) |
SOBCOR(de, C, reator) |
PROTINTRENROL(de, reator) |
PROTINTRVALV(de, reator) |
PROTINTROLEO(de, reator))

R11. Desarme total de linha com atuação da proteção com bloqueio por falha do disjuntor de reator (lado PARA)

Condição de existência: **Existe(para) & Existe(para, dj_r) & Existe(para, reator)**

DESARMETOTAL(linha, de, para) &
FALHADJ(para, dj_r) &
(ATPR(de) | ATPR(para)) &
(RBLOQUEIO(de) | RBLOQUEIO(para)) &
TRIP86(para, reator) &
(ATUAFASE(para, A, reator) |
ATUAFASE(para, B, reator) |
ATUAFASE(para, C, reator) |
ATUAFASEPARALELO(para, reator) |
SOBTMPOLEO(para, A, reator) |
SOBTMPOLEO(para, B, reator) |
SOBTMPOLEO(para, C, reator) |
SOBTMPENROL(para, A, reator) |
SOBTMPENROL(para, B, reator) |
SOBTMPENROL(para, C, reator) |
GAS(para, A, reator) |
GAS(para, B, reator) |
GAS(para, C, reator) |
VALVSEG(para, A, reator) |
VALVSEG(para, B, reator) |
VALVSEG(para, C, reator) |
SOBCOR(para, A, reator) |
SOBCOR(para, B, reator) |
SOBCOR(para, C, reator) |
PROTINTRENROL(para, reator) |
PROTINTRVALV(para, reator) |

PROTINTROLEO(para, reator))

R12. Desligamento parcial de LT lado DE - Manual
Condição de existência: A regra sempre existe.
DESLIGAMENTOPARCIALDE(linha, de, para)

R13. Desligamento parcial de LT lado PARA - Manual
Condição de existência: Existe(para)
DESLIGAMENTOPARCIALPARA(linha, de, para)

R14. Desarme parcial de LT lado DE por atuação indevida da proteção sem bloqueio
Condição de existência: A regra sempre existe.
DESARMEPARCIALDE(linha, de, para) &
ATPR(de) & !(ATPR(para) &
!RBLOQUEIO(de) &
!(SOBTENTEMP(de) | SOBTENINST(de)) &
!(FALHADJ(de, dj_1) | FALHADJ(de, dt1) | [FALHADJ(de, dt2)])

R15. Desarme parcial de LT lado PARA por atuação indevida da proteção sem bloqueio
Condição de existência: Existe(para)
DESARMEPARCIALPARA(linha, de, para) &
ATPR(para)) & !(ATPR(de) &
!(RBLOQUEIO(para)) &
!(SOBTENTEMP(para) | SOBTENINST(para)) &
!(FALHADJ(para, dj_1) | FALHADJ(para, djt1) | [FALHADJ(para, dt2)])

R16. Desarme parcial de LT lado DE com atuação da proteção com bloqueio
Condição de existência: A regra sempre existe.
DESARMEPARCIALDE(linha, de, para) &
ATPR(de) & !(ATPR(para)) &
RBLOQUEIO(de) & !(RBLOQUEIO(para))

R17. Desarme parcial de LT lado PARA com atuação da proteção com bloqueio
Condição de existência: Existe(para)
DESARMEPARCIALPARA(linha, de, para) &
ATPR(para) & !(ATPR(de)) &
RBLOQUEIO(para) & !(RBLOQUEIO(de))

R18. Desligamento Consumidor Industrial
Condição de existência: A regra existe se lado para for CI
!CONECTADO(linha, de) &
!ATPR(de) &
MW(de, ↓0) &
MVAR(de, ↓0)

R19. Desarme Consumidor Industrial com Atuação da Proteção com Bloqueio
Condição de existência: A regra existe se lado para for CI
!CONECTADO(linha, de) &
MW(de, ↓0) &
MVAR(de, ↓0) &
ATPR(de) &
RBLOQUEIO(de)

R20. Desarme Consumidor Industrial com Atuação da Proteção sem Bloqueio
Condição de existência: A regra existe se lado para for CI
!CONECTADO(linha, de) &
MW(de, ↓0) &
MVAR(de, ↓0) &
ATPR(de) &

```
!RBLOQUEIO(de)
```

R21. Desarme Consumidor Industrial com atuação da proteção com bloqueio por falha do disjuntor lado

```
Condição de existência:  A regra existe se lado para for CI
(!CONECTADO(linha, (de, B1)) | FALHADJ(de, Dj1))     &
(!CONECTADO(linha, (de, B2)) | FALHADJ(de, Djt1))    &
ATPR(de)                                             &
RBLOQUEIO(de)                                        &
(FALHADJ(de, dj1) | FALHADJ(de, djt1) | [FALHADJ(de, djt2)])
```

Regras de Linhas de Transmissão do Sistema Leste (500KV)

Topologia

```
Linhas de 500KV:
SE's: AGD, RCD, MSI
      Arranjo: 2 barras, disjuntor e meio
      Dispositivos:
         o  Disjuntor de linha: Dj1
         o  Chaves do disjuntor de linha: Dj1-4 e Dj1-5
         o  Disjuntor central: Dj2
         o  Chaves do disjuntor central: Dj2-1 e Dj2-2
         o  Disjuntor de reator: Djr
         o  Chave do disjuntor de reator: Djr-8
         o  Reator
         o  Barra 1: B1
         o  Barra2: B2
```

Macro EXISTE(parâmetro)

```
Retorna TRUE ou FALSE, indicando se o parâmetro existe.
```

Macro !CONECTADO(linha, parâmetro)

```
Parâmetro pode ser subestação ou barra.
<< Arranjo 1 >>
((DJ(subest, dj1) | CV(subest, dj1-4) | CV(subest, dj1-5)) &
(DJ(subest, dj2) | CV(subest, dj2-1) | CV(subest, dj2-2))
```

Macro DESARMETOTAL(linha, de, para)

```
!CONECTADO(linha, de)   & !CONECTADO(linha, para)     &
KV(de,↓0)                     & KV(para, ↓0)                &
MW(de, ↓0)              & MW(para, ↓0)                &
MVAR(de, ↓0)            & MVAR(para, ↓0)
```

Macro DESLIGAMENTOTOTAL(linha, de, para)

```
DESARMETOTAL(linha, de, para)       &
!ATPR(de)                                  &
!ATPR(para)
```

Macro DESARMEPARCIALDE(linha, de, para)

```
!CONECTADO(linha, de)   & CONECTADO(linha, para)      &
KV(de, !0)              & KV(para, !0)                &
MW(de, ↓0)              & MW(para, ↓0)                &
MVAR(de, ↓0)                  & MVAR(para, !0)
```

Macro DESLIGAMENTOPARCIALDE(linha, de, para)

```
DESARMEPARCIALDE(linha, de, para) &
!ATPR(de)
```

Macro DESARMEPARCIALPARA(linha, de, para)

```
CONECTADO(linha, de)    & !CONECTADO(linha,para)      &
```

KV(de, !0) & KV(para, !0) &
MW(de, ↓0) & MW(para, ↓0) &
MVAR(de, !0) & MVAR(para, ↓0)

Macro DESLIGAMENTOPARCIALPARA(linha, de, para)
DESARMEPARCIALPARA(linha, de, para) &
!ATPR(para)

R1. Desligamento Total de LT - Manual
Condição de existência: Existe(para)
DESLIGAMENTOTOTAL(linha, de, para)

R2. Desarme total de linha com atuação da proteção sem religamento
Condição de existência: Existe(para)
DESARMETOTAL(linha, de, para) &
ATPR(de) & ATPR(para) &
!RBLOQUEIO(de) & !RBLOQUEIO(para) &
!SOBTENTEMP(de) & !SOBTENTEMP(para) &
!SOBTENINST(de) & !SOBTENINST(para) &
!FALHADJ(de, dj_1) & !FALHADJ(de, dj_2) &
!FALHADJ(para, dj_1) & !FALHADJ(para, dj_2) &
!((RELIGEFETDJ(de, dj_1) & DJ(de, dj_1) & !RELIGBLOQDJ(de, dj_1)) |
(RELIGEFETDJ(de, dj_2) & DJ(de, dj_2) & !RELIGBLOQDJ(de, dj_2))) &
!((RELIGEFETDJ(para, dj_1) & DJ(para, dj_1) & !RELIGBLOQDJ(para, dj_1)) |
(RELIGEFETDJ(para, dj_2) & DJ(para, dj_2) & !RELIGBLOQDJ(para, dj_2)))

R3. Desarme total de linha com atuação da proteção sem religamento lado DE com religamento sem sucesso lado PARA
Condição de existência: Existe(para)
DESARMETOTAL(linha, de, para) &
ATPR(de) & ATPR(para) &
!RBLOQUEIO(de) & !RBLOQUEIO(para) &
!SOBTENTEMP(de) & !SOBTENTEMP(para) &
!SOBTENINST(de) & !SOBTENINST(para) &
!FALHADJ(de, dj_1) & !FALHADJ(de, dj_2) &
!FALHADJ(para, dj_1) & !FALHADJ(para, dj_2) &
!((RELIGEFETDJ(de, dj_1) & DJ(de, dj_1) & !RELIGBLOQDJ(de, dj_1)) |
(RELIGEFETDJ(de, dj_2) & DJ(de, dj_2) & !RELIGBLOQDJ(de, dj_2))) &
((RELIGEFETDJ(para, dj_1) & !DJ(para, dj_1) & !RELIGBLOQDJ(para, dj_1)) |
(RELIGEFETDJ(para, dj_2) & !DJ(para, dj_2) & !RELIGBLOQDJ(para, dj_2)))

R4. Desarme total de linha com atuação da proteção sem religamento lado PARA e com religamento sem sucesso lado DE
Condição de existência: Existe(para)
DESARMETOTAL(linha, de, para) &
ATPR(de) & ATPR(para) &
!RBLOQUEIO(de) & !RBLOQUEIO(para) &
!SOBTENTEMP(de) & !SOBTENTEMP(para) &
!SOBTENINST(de) & !SOBTENINST(para) &
!FALHADJ(de, dj_1) & !FALHADJ(de, dj_2) &
!FALHADJ(para, dj_1) & !FALHADJ(para, dj_2) &
((RELIGEFETDJ(de, dj_1) & !DJ(de, dj_1) & !RELIGBLOQDJ(de, dj_1)) |
(RELIGEFETDJ(de, dj_2) & !DJ(de, dj_2) & !RELIGBLOQDJ(de, dj_2))) &
!((RELIGEFETDJ(para, dj_1) & DJ(para, dj_1) & !RELIGBLOQDJ(para, dj_1)) |
(RELIGEFETDJ(para, dj_2) & DJ(para, dj_2) & !RELIGBLOQDJ(para, dj_2)))

R5. Desarme total de linha com atuação da proteção com bloqueio por sobretensão

Condição de existência: Existe(para)

DESARMETOTAL(linha, de, para) &
(ATPR(de) | ATPR(para)) &
(RBLOQUEIO(de) | RBLOQUEIO(para)) &
(SOBTENTEMP(de) | SOBTENINST(de)) &
(SOBTENTEMP(para) | SOBTENINST(para))

R6. Desarme total de linha com atuação da proteção com bloqueio por sobretensão lado DE

Condição de existência: Existe(para)

DESARMETOTAL(linha, de, para) &
(ATPR(de) | ATPR(para)) &
(RBLOQUEIO(de) | RBLOQUEIO(para)) &
(SOBTENTEMP(de) | SOBTENINST(de)) &
!SOBTENTEMP(para) &
!SOBTENINST(para)

R7. Desarme total de linha com atuação da proteção com bloqueio por sobretensão lado PARA

Condição de existência: Existe(para)

DESARMETOTAL(linha, de, para) &
(ATPR(de) | ATPR(para)) &
(RBLOQUEIO(de) | RBLOQUEIO(para)) &
(SOBTENTEMP(para) | SOBTENINST(para)) &
!SOBTENTEMP(de) &
!SOBTENINST(de)

R8. Desarme total de linha com atuação da proteção com bloqueio por falha do disjuntor lado DE

Condição de existência: Existe(para)

!CONECTADO(linha, para) &
(!CONECTADO(linha, (de, B1)) | FALHADJ(de, dj1)) &
(!CONECTADO(linha, (de, B2)) | FALHADJ(de, dj2)) &
(ATPR(de) | ATPR(para)) &
(RBLOQUEIO(de) | RBLOQUEIO(para)) &
(FALHADJ(de, dj_1) | FALHADJ(de, dj_2))

R9. Desarme total de linha com atuação da proteção com bloqueio por falha do disjuntor lado PARA

Condição de existência: Existe(para)

!CONECTADO(linha, de) &
(!CONECTADO(linha, (para, B1)) | FALHADJ(para, dj1)) &
(!CONECTADO(linha, (para, B2)) | FALHADJ(para, dj2))&
(ATPR(de) | ATPR(para)) &
(RBLOQUEIO(de) | RBLOQUEIO(para)) &
(FALHADJ(para, dj_1) | FALHADJ(para, dj_2))

R10. Desarme total de linha com atuação da proteção com bloqueio por falha do disjuntor de reator (lado DE)

Condição de existência: Existe(para) & Existe(de, dj_r) & Existe(de, reator)

DESARMETOTAL(linha, de, para) &
FALHADJ(de, dj_r) &
(RBLOQUEIO(de) | RBLOQUEIO(para)) &
TRIP86(de, reator) &
(ATUAFASE(de, A, reator) |

ATUAFASE(de, B, reator) |
ATUAFASE(de, C, reator) |
ATUAFASEPARALELO(de, reator) |
SOBTMPOLEO(de, A, reator) |
SOBTMPOLEO(de, B, reator) |
SOBTMPOLEO(de, C, reator) |
SOBTMPENROL(de, A, reator) |
SOBTMPENROL(de, B, reator) |
SOBTMPENROL(de, C, reator) |
GAS(de, A, reator) |
GAS(de, B, reator) |
GAS(de, C, reator) |
VALVSEG(de, A, reator) |
VALVSEG(de, B, reator) |
VALVSEG(de, C, reator) |
SOBCOR(de, A, reator) |
SOBCOR(de, B, reator) |
SOBCOR(de, C, reator) |
PROTINTRENROL(de, reator) |
PROTINTRVALV(de, reator) |
PROTINTROLEO(de, reator))

R11. Desarme total de linha com atuação da proteção com bloqueio por falha do disjuntor de reator (lado PARA)

Condição de existência: Existe(para) & Existe(para, dj_r) & Existe(para, reator)

DESARMETOTAL(linha, de, para) &
FALHADJ(para, dj_r) &
(RBLOQUEIO(de) | RBLOQUEIO(para)) &
TRIP86(para, reator) &
(ATUAFASE(para, A, reator) |
ATUAFASE(para, B, reator) |
ATUAFASE(para, C, reator) |
ATUAFASEPARALELO(para, reator) |
SOBTMPOLEO(para, A, reator) |
SOBTMPOLEO(para, B, reator) |
SOBTMPOLEO(para, C, reator) |
SOBTMPENROL(para, A, reator) |
SOBTMPENROL(para, B, reator) |
SOBTMPENROL(para, C, reator) |
GAS(para, A, reator) |
GAS(para, B, reator) |
GAS(para, C, reator) |
VALVSEG(para, A, reator) |
VALVSEG(para, B, reator) |
VALVSEG(para, C, reator) |
SOBCOR(para, A, reator) |
SOBCOR(para, B, reator) |
SOBCOR(para, C, reator) |
PROTINTRENROL(para, reator) |
PROTINTRVALV(para, reator) |
PROTINTROLEO(para, reator))

R12. Desarme total de linha com atuação da proteção com bloqueio por proteção do reator (lado PARA)

Condição de existência: Existe(para) & Existe((para, reator))
DESARMETOTAL(linha, de, para) &
!FALHADJ(para, dj_r) &
(RBLOQUEIO(de) | RBLOQUEIO(para)) &
TRIP86(para, reator) &
(ATUAFASE(para, A, reator) |
ATUAFASE(para, B, reator) |
ATUAFASE(para, C, reator) |
ATUAFASEPARALELO(para, reator) |
SOBTMPOLEO(para, A, reator) |
SOBTMPOLEO(para, B, reator) |
SOBTMPOLEO(para, C, reator) |
SOBTMPENROL(para, A, reator) |
SOBTMPENROL(para, B, reator) |
SOBTMPENROL(para, C, reator) |
GAS(para, A, reator) |
GAS(para, B, reator) |
GAS(para, C, reator) |
VALVSEG(para, A, reator) |
VALVSEG(para, B, reator) |
VALVSEG(para, C, reator) |
SOBCOR(para, A, reator) |
SOBCOR(para, B, reator) |
SOBCOR(para, C, reator) |
PROTINTRENROL(para, reator) |
PROTINTRVALV(para, reator) |
PROTINTROLEO(para, reator))

R13. Desarme total de linha com atuação da proteção com bloqueio por proteção do reator (lado DE)

Condição de existência: Existe(para) & Existe((de, reator))
DESARMETOTAL(linha, de, para) &
!FALHADJ(de, dj_r) &
(RBLOQUEIO(de) | RBLOQUEIO(para)) &
TRIP86(de, reator) &
(ATUAFASE(de, A, reator) |
ATUAFASE(de, B, reator) |
ATUAFASE(de, C, reator) |
ATUAFASEPARALELO(para, reator) |
SOBTMPOLEO(de, A, reator) |
SOBTMPOLEO(de, B, reator) |
SOBTMPOLEO(de, C, reator) |
SOBTMPENROL(de, A, reator) |
SOBTMPENROL(de, B, reator) |
SOBTMPENROL(de, C, reator) |
GAS(de, A, reator) |
GAS(de, B, reator) |
GAS(de, C, reator) |
VALVSEG(de, A, reator) |
VALVSEG(de, B, reator) |
VALVSEG(de, C, reator) |
SOBCOR(de, A, reator) |
SOBCOR(de, B, reator) |
SOBCOR(de, C, reator) |
PROTINTRENROL(de, reator) |

PROTINTRVALV(de, reator) |
PROTINTROLEO(de, reator))

R14. Desligamento parcial de LT lado DE - Manual
Condição de existência: A regra sempre existe.
DESLIGAMENTOPARCIALDE(linha, de, para)

R15. Desligamento parcial de LT lado PARA - Manual
Condição de existência: Existe(para)
DESLIGAMENTOPARCIALPARA(linha, de, para)

R16. Desarme parcial de LT lado DE por atuação indevida da proteção sem bloqueio
Condição de existência: A regra sempre existe.
DESARMEPARCIALDE(linha, de, para) &
ATPR(de) & !(ATPR(para) &
!(RBLOQUEIO(de)) &
!(SOBTENTEMP(de) | SOBTENINST(de)) &
!(FALHADJ(de, dj_1) | FALHADJ(de, dj_2))

R17. Desarme parcial de LT lado PARA por atuação indevida da proteção sem bloqueio
Condição de existência: Existe(para)
DESARMEPARCIALPARA(linha, de, para) &
ATPR(para)) & !(ATPR(de) &
!(RBLOQUEIO(para)) &
!(SOBTENTEMP(para) | SOBTENINST(para)) &
!(FALHADJ(para, dj_1) | FALHADJ(para, dj_2))

R18. Desarme parcial de LT lado DE com atuação indevida da proteção com bloqueio
Condição de existência: A regra sempre existe.
DESARMEPARCIALDE(linha, de, para) &
ATPR(de) &
(!ATPR(para) | SOBTENTEMP(de) | SOBTENINST(de))&
!(FALHADJ(de, dj_1) | FALHADJ(de, dj_2)) &
RBLOQUEIO(de) & !(RBLOQUEIO(para))

R19. Desarme parcial de LT lado PARA com atuação indevida da proteção com bloqueio
Condição de existência: Existe(para)
DESARMEPARCIALPARA(linha, de, para) &
ATPR(para) &
(!ATPR(de) | SOBTENTEMP(para) | SOBTENINST(para)) &
!(FALHADJ(para, dj_1) | FALHADJ(para, dj_2)) &
RBLOQUEIO(para) & !(RBLOQUEIO(de))

R20. Desarme parcial de LT lado DE por sobrecarga do disjuntor central
Condição de existência: **Existe((de, dj2))**
DESARMEPARCIALDE(linha, de, para) &
SOBCARGA(de, dj2) &
!FALHADJ(de, dj2)

R21. Desarme parcial de LT lado PARA por sobrecarga do disjuntor central
Condição de existência: **Existe((para, dj2))**
DESARMEPARCIALPARA(linha, de, para) &
SOBCARGA(para, dj2) &
!FALHADJ(para, dj2)

Referências Bibliográficas

(AZEVEDO, 2001) AZEVEDO, G. P.; **Aplicações de Técnicas de Inteligência Artificial à Operação em Tempo Real de Sistemas de Potência.** II Workshop de Técnicas de I.A. Aplicadas a Sistemas de Potência e Industriais, 2001.

(BIBAS, 1996) BIBAS, S., CORDIER M.O., DAGUE P., DOUSSON C., LÉVI F. E ROZÉ L. **Alarm Driven supervision for telecommunication networks:** I-Off-line scenario generation**.** Anales des telecommunications, vol 9/10, pp. 430-500, 1996.

(BIELER, 1994) BIELER, K.; GLAVITSCH, H. **Evaluation of different AI-methods for fault diagnosis in power systems**. In:. International Conference on Intelligent System Application to Power Systems*,* 1994, Nanterre Cedex, France, 1994. v. 1, p. 209-216.

(BRUNNER, 1993) BRUNNER, T.; NEJDL, W.; SCHWARZJIRG, H.; STURM, M. **Online expert system for power system diagnosis and restoration. Intelligent Systems Engineering.** v. 2, n. 1, p. 15-24, 1993.

(CEPEL, 2002) CEPEL. **Guia de Instalação SAGE 2002**. Maio. 2002.

(CORDIER, 2000) CORDIER, M.; DOUSSON, C. **Alarm Driven Monitoring Based on Chronicles**. Proc. of SAFEPROCESS 2000, pp 286-291. Budapeste, Hungria.

(CORDIER, 2001) CORDIER, M.; KRIVINE, J.; LABORIE, P.; THIEBAUX, S. **Alarm processing and Reconfiguration in Power Distribution Systems.** 2001.

(CRONK, 1998) CRONK, R.; CALLAHAN, P.; BERNSTEIN, L. **Rule-based expert systems for network management and operations**: an introduction. IEEE Network, v. 2, n. 5, p. 7-21, 1988.

(DABBAGHCHI, 1997) DABBAGHCHI, I.; CHRISTIE, R.; ROSENWALD, G.; LIU, C. **AI Application Areas in Power Systems.** IEEE Expert, v. 12, n. 1, p. 58-66, 1997.

(DIETZ, 1988) DIETZ, W.E.; KIECH, E.L.; ALI, M. **Pattern-based Fault Diagnosis using Neural Networks.** 1988.

(DOUSSON, 1993) DOUSSON, C.; GABORIT, P.; GHALLAB. **Situation recognition: Representation and algorithms**. Proc. of the 13th IJCAI pp. 166-172. Chambéry, France, 1993.

(DOUSSON, 1996) DOUSSON C., **Alarm driven supervision for telecommunication networks**: II-On-line chronicle recognition. Annales des Télécommunications, vol 9/10, pp. 501-508, 1996.

(FLOYD, 1962) FLOYD, R. W. **Algorithm 97 (Shortest Path)**. Communications of the ACM, Volume 5, Number 6, pp. 345, 1962.

(FONTAINE, 1997) FONTAINE, D.; RAMAUX, N. **An approach by graph for the recognition of temporal scenarios**. IEEE transaction on System, Man and Cybernetics, 1997.

(FREY, 1997) FREY, J.; LEWIS, L. **Multi-level reasoning for managing distributed enterprises and their networks**. In: Integrated Network Management, 5, 1997, p. 5-16

(GAMMA, 1999) GAMMA, E.; BECK, K. **Junit**: A cook's tour. Java Report, 4(5):27--38, May 1999.

(GARDNER, 1996) GARDNER, R. R.; HARLE, D. A. **Methods and Systems for Alarm Correlation**, Proceedings of Globecom 96, Vol.1, pp.136-140, London, Nov. 18-22, 1996, pp 136-140.

(GHALLAB, 1996) GHALLAB, M. **On Chronicles**: Representation, On-line Recognition and Learning. Proc of 5th International Conference on Principles of Knowledge Representation and Resoaning (KR-96). Massachusetts, USA, 1996.

(HARLOW, 1998) HARLOW, J. H. Transformers. In: GRIGSBY, L. L. **The Eletric Power Engineering Handbook**. Auburn, Alabama: CRC Press/IEEE Press, 1998. 3, p3.1-3.268.

(HASAN, 1999) HASAN, M.; SUGLA, B.; VISWANATHAN R. **A Conceptual Framework for Network Management Event Correlation and Filtering Systems**. Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management (IM), 1999.

(HP, 1995) HEWLLET-PACKARD. **HP OpenView event correlation for the telecommunications environment**: Technology brief, September 1995.

(ISO/IEC, 1984) ISO/IEC. Information Processing Systems - **Open Systems Interconnection**: Basic Reference Model. International Organization for Standardization and International Electrotechnical Committee, International Standard 7498, 1984.

(JAKOBSON, 1993) JAKOBSON, G.; WEISSMAN, M. **Alarm Correlation**. IEEE Network, v. 7, n. 6, p. 52-59, nov. 1993.

(JAKOBSON, 1995) JAKOBSON, G.; WEISSMAN, M. **Real-time telecomunication network management: extending event correlation with temporal constraints**. In: IFIP/IEEE Internation Symposium on Integrated Network Management, 4, 1995, p. 290-301.

(JEFFRIES, 2001) JEFFRIES, R. E.; ANDERSON, A.; HENDRICKSON, C. **Extreme Programming Installed**. Addison-Wesley, 2001.

(KARADY, 1998) KARADY, G. G. Transmission System. In: GRIGSBY, L. L. **The Eletric Power Engineering Handbook**. Auburn, Alabama: CRC Press/IEEE Press, 1998. 4, p4.1-4.169

(KATKER, 1997) KATKER, S.; PATEROK, M. **Fault isolation and event correlation for integrated fault management**. In: Integrated Network Management, 5, 1997, p. 583-596.

(KATZELA, 1995) KATZELA, I.; SCHWARTZ, M. **Schemes for fault identification in communication networks**. IEEE Transactions on Networking, v. 3, n. 6, p. 753-764, dec. 1995.

(KLIGER, 1995) KLIGER, S.; YEMINI, S.; YEMINI, Y.; OHSIE, D.; STOLFO, S. **A coding approach to event correlation**. In IFIP/IEEE Internation Symposium on Integrated Network Management, 4, 1995, p. 266-277.

(LABORIE, 1997) LABORIE, P e J-P-KRIVINE. GEMO: **A model-based approach for an alarm processing function in power distribution system networks**. International conference on Intelligent System Application to Power Systems (ISAP'97), p. 135-141, Seul, Coréia do Sul.

(LEVY, 1994) LÉVY, F. (1994). **Recognising scenarios**: a study. Fifth international workshop of diagnosis (DX'94). p. 174-178. New Paltz.

(LEWIS, 1993) LEWIS, L. **A Case-Based Reasoning Approach to the Resolution of Faults in Communications Networks**. In: Integrated Network Management, 3, p. 671-682, 1993.

(LEWIS, 1993-2) LEWIS, L.; DREO, G. **Extendig trouble ticket systems to fault diagnostics**. IEEE Network, v. 7, n. 6, p. 44-51, nov. 1993.

(LEWIS, 1999) LEWIS, L. **Event Correlation in SPECTRUM and Other Commercial Products.** Technical Report ctron-Imp-99-05, Cabletron: 1999.

(MAGDA, 1999) MAGDA, Projeto. **MAGDA/PF/LIV/001 Requirements Versão 1**. 1999.

(MANSFIELD, 1993) MANSFIELD, G.; JAYANTHI, K.; HIGUCHI, K.; NEMOTO, Y.; NOGUCHI, S. **The MIKB model for intelligent network management**. In: IEEE Internation Conference on Communications,1993, p. 1210-1214.

(MCDONALD,1998) MCDONALD, J. D. Substations. In: GRIGSBY, L. L. **The Eletric Power Engineering Handbook**. Auburn, Alabama: CRC Press/IEEE Press, 1998. 5, 5.1-5.134.

(MEIRA, 1997) MEIRA, D. **A Model for Alarm Correlation in Telecommunications Networks**. Ph.D Thesis. Computer Science. Institue of Exact Sciences (ICEx) of the UFMG. Belo Horizonte, Brazil, 1997.

(OATES, 1997) OATES, T.; JENSEN, D.; COHEN, P.R. **Automatically Acquiring Rules for Event Correlation from Event Logs.** Computer Science Technical Report 97-14, Experimental Knowledge Systems Laboratory, Computer Science Department, University of Massachusetts, 1997.

(RAHMAN, 1998) RAHMAN, S. Eletric Power Generation: Non-Conventional Methods. In: GRIGSBY, L. L. **The Eletric Power Engineering Handbook**. Auburn, Alabama: CRC Press/IEEE Press, 1998. p. 1, 1.1-1.14.

(RAMAKUMAR, 1998) RAMAKUMAR, R. Eletric Power Generation: Conventional Methods. In: GRIGSBY, L. L. **The Eletric Power Engineering Handbook**. Auburn, Alabama: CRC Press/IEEE Press, 1998. p. 2, 2.1-2.27.

(SASISEKHARAN, 1994) SASISEKHARAN, R.; SESHADRI, V.; WEISS, S. **Using machine learning to monitor network performance**. IEEE Conference on Artificial Intelligence Applications, p. 92-98, 1994.

(SEAGATE, 1996) SEAGATE. **Nervecenter Pro**: The complete solution for managing network and system behavior, September 1996. http://www.sems.com/Products/West/nervecenter/Nervecenter.html

(SILVA, 1999) SILVA, A.; FILHO, A.; PEREIRA, L.; LIMA, L.; LAMBERT, N.; AMORIM, M.; AZEVEDO, G. SAGE **Architecture for Power System Competitive Environments**. 1999.

(VALE, 1997) VALE, Z.; MOURA, A.; FERNANDES, M.; MARQUES, A; ROSADO, C.; RAMOS, C. **SPARSE**: An Intelligent Alarm Processor and Operator Assistant. IEEE Expert 12(3): p. 86-93,

1997.

(WOLLENBERG, 1998) WOLLENBERG, B. F. Power System Operation and Control. In: GRIGSBY, L. L. **The Eletric Power Engineering Handbook**. Auburn, Alabama: CRC Press/IEEE Press, 1998. 12, p. 12.1-12.53.

(YEMINI) YEMINI, Y.; YEMINI, S.; KLIGER, S. **Apparatus and Method for Event Correlation and Problem Reporting.** United States Patent 5,528,516.

(YEMINI, 1996) YEMINI, S.; KLIGER, S.; MOZES, E.; YEMINI, Y.; OHSIE, D. **High Speed and Robust Event Correlation**. IEEE Communications Magazine, p. 82-90, Mai. 1996.

DUARTE, Alexandre Nóbrega.

D812T

Tratamento de Eventos em Redes Elétricas: Uma Ferramenta

Campina Grande: 2003. 140 f. Dissertação (Mestrado em Informática) – Universidade Federal de Campina Grande – 2003.

1. Redes de Computadores

2. Correlação de Eventos

3. Processamento de Eventos

4. Diagnóstico de Causa Raiz

5. Raciocínio Baseado em Regras

6. Raciocínio Baseado em Modelos

CDU – 621.391