IMPLEMENTAÇÃO DE UM SIMULADOR DAS LÓGICAS DE INTERTRAVAMENTO E PROTEÇÃO DE SUBESTAÇÕES DE ENERGIA ELÉTRICA

GERSON KAZUYOSHI KIDA

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

IMPLEMENTAÇÃO DE UM SIMULADOR DAS LÓGICAS DE INTERTRAVAMENTO E PROTEÇÃO DE SUBESTAÇÕES DE ENERGIA ELÉTRICA

GERSON KAZUYOSHI KIDA

ORIENTADOR: DR. KLEBER MELO E SILVA

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO: PPGENE.DM – 508/2012
BRASÍLIA/DF: SETEMBRO-2012

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

IMPLEMENTAÇÃO DE UM SIMULADOR DAS LÓGICAS DE INTERTRAVAMENTO E PROTEÇÃO DE SUBESTAÇÕES DE ENERGIA ELÉTRICA

GERSON KAZUYOSHI KIDA

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

APROVADA POR:

KLEBER MELO E SILVA, Doutor, ENE/UNB

(ORIENTADOR)

Flávio Elias Gomes de Deus, Doutor, ENE/UNB

(EXAMINADOR INTERNO)

Flávio Bezerra Costa, Doutor, FCT/UFRN

(EXAMINADOR EXTERNO)

BRASÍLIA (DF), 14 DE SETEMBRO DE 2012.

FICHA CATALOGRÁFICA

KIDA, GERSON KAZUYOSHI.

Implementação de um Simulador das Lógicas de Intertravamento e Proteção de Subestações de Energia Elétrica. [Distrito Federal], 2012.
xv, 119p., 210 x 297 mm (ENE/FT/UnB, Mestre, Dissertação de Mestrado – Universidade de Brasília. Faculdade de Tecnologia.

Dissertação de Mestrado – Universidade de Brasília, Faculdade de Tecnologia. Departamento de Engenharia Elétrica.

1. Simulador 2. Intertravamento

3. Orientação a Objeto 4. C#

5. Subestação de Energia Elétrica.

I. ENE/FT/UNB. II. Título (Série)

REFERÊNCIA BIBLIOGRÁFICA

KIDA, Gerson Kazuyoshi (2012). Implementação de um Simulador das Lógicas de Intertravamento e Proteção de Subestações de Energia. Dissertação de Mestrado em Engenharia Elétrica, Publicação ENE 508/2012 DM SETEMBRO/2012, Departamento de Engenharia Elétrica, Universidade de Brasília , Brasília, DF, 119p.

CESSÃO DE DIREITOS

AUTOR: Gerson Kazuyoshi Kida
TÍTULO: IMPLEMENTAÇÃO DE UM SIMULADOR DAS LÓGICAS DE INTERTRAVAMENTO E PROTEÇÃO DE SUBESTAÇÕES DE ENERGIA ELÉTRICA.
GRAU: Mestre ANO: 2012.

GERSON KAZUYOSHI KIDA
Av. Vereador Juliano Costa Marques, 615 – Cond. Parque Residencial Pantanal 3 – Edifício Torre dos Lagos - Apto. 402 – Bairro Aclimação
Cep 78.050-253 – Cuiabá – MT – Brasil

AGRADECIMENTOS

Ao meu orientador prof. Dr. Kleber Melo e Silva, pela sua grande colaboração para o desenvolvimento deste trabalho, permitindo a ampliação de meu conhecimento nesta área de engenharia elétrica.

Aos professores da UNB que participaram deste programa de capacitação, em especial o prof. Dr. Franklin da Costa Silva.

Ao Sr. Nonato administrador dos apartamentos de trânsito, que com sua gentileza e paciência, colaborou com a minha estadia em diversos momentos que estive em Brasília.

À direção e funcionários do IFMT que não mediram esforços para realização deste programa de Mestrado Interinstitucional.

Aos colegas de "sala" que participaram, compartilharam e incentivaram na busca do conhecimento para realização deste objetivo, em especial André e Marilson companheiros de viagem.

Aos professores e funcionários do Departamento de Informática do IFMT que colaboraram, direta e indiretamente, com minhas frequentes viagens a Brasília.

DEDICATÓRIA

Dedico aos meus pais Tosihiro Kida (in memoriam) e Sano Yurico Kida, que não mediram esforços para o crescimento de meu conhecimento.

À minha esposa Celia e meus filhos Gustavo e Marcos que representam tudo em minha vida e a razão de meus esforços, dedico a vocês este trabalho.

RESUMO

Neste trabalho é proposto um simulador das lógicas de intertravamento de chaves seccionadoras e proteção dos componentes elétricos de uma subestação de energia elétrica hipotética, desenvolvido na linguagem de programação C#, utilizando-se o paradigma da orientação a objetos. O simulador foi implementado visando servir como ferramenta de auxílio no treinamento de operadores de subestações, melhorando a agilidade na solução de distúrbios e retorno a condição normal. Os resultados obtidos nos testes e manobras comprovam a funcionalidade do sistema e a viabilidade de utilização para treinamento de operadores.

Palavras-chaves: Simulador; Intertravamento; Orientação a Objeto; C#; Subestação de Energia Elétrica.

ABSTRACT

In this work, it is proposed an interlock switchgear and power system protection logics simulator for a hypothetical power substation, which is implemented by using the C# language programming and the paradigm of object-oriented programming. The simulator aims to support the training of power substations operators, in order to improve its ability to deal with disturbances situations and the actions that must be done to restabilish the normal condition. The obtained results demonstrate the system functionality and feasibility of use for training operators of electric power substations.

Keywords: Simulator, Interlocking, Object Orientation, C #, Electricity Substation.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ANSI - American National Standards Institute

AT - Autotransformador

C# - Linguagem de programação

DJ - Disjuntor

ID - Identificador

IEEE - Institute of Electrical and Electronic Engineers

LT - Linha de Transmissão

NA - Normalmente aberto

NF - Normalmente fechado

SC - Chave seccionadora

SE - Subestação de Energia Elétrica

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO DO TEMA

A energia elétrica é um bem essencial à sociedade, tanto para a vida como para seu desenvolvimento. Ela está disponível de forma adequada aos consumidores, mas para que isso ocorra, é necessário um planejamento, projeto, instalação e principalmente a operação de uma vasta e complexa infraestrutura, que envolve a geração, transmissão, distribuição e centros de comandos.

Associado ao funcionamento de toda infraestrutura estão os constantes distúrbios causados por problemas internos nos equipamentos e também à causas externas, como, variações de carga, descargas atmosféricas, ventos ou tempestades.

O diagnóstico dos distúrbios em tempo real e a regularização do serviço são tarefas de muita responsabilidade e que podem ser de grande complexidade e decisiva para o retorno a condição normal. Numa situação de distúrbio as mensagens geradas pelos equipamentos causam enormes dificuldades para os operadores solucionar os problemas.

O desenvolvimento de ferramentas que auxiliem os operadores a desempenhar melhor suas funções tem contribuído para conseguir um melhor desempenho na operação de uma Subestação de Energia (SE).

É importante lembrar que o treinamento por meio eletrônico não substitui outros métodos empregados, sendo uma opção a mais para facilitar a aprendizagem dos operadores (LEITE, SILVA e FILHO, 2010).

Com a utilização da informática, a criação de um simulador de intertravamento de chaves seccionadoras de uma subestação, torna o treinamento de operadores para a operação de abertura e fechamentos de seccionadoras de forma mais fácil e efetivo e a um custo mais baixo.

1.2 REVISÃO BIBLIOGRÁFICA

Nesta seção são abordados os principais artigos que contribuíram com o desenvolvimento do trabalho.

O trabalho de Moutinho (2010) apresenta o desenvolvimento de um simulador de relés de relés de proteção, controle e supervisão aplicados às SEs da Eletronorte, utilizando informações fornecidas pelos equipamentos de potência, o aplicativo simula as informações em tempo real.

Padmanabhan e Kinder (2008) apresentam a troca de um sistema de intertravamento de Sistema de Controle de Subestação por uma versão mais recente, de maneira que a implementação do novo sistema não interfira na integridade do sistema existente.

Vasques (2011) apresenta solicitações elétricas impostas a autotransformadores de potência causados por manobras de chaves seccionadoras e disjuntores, avaliando seus efeitos e riscos envolvidos no processo. Contendo noções fundamentais e definições importantes de transformadores de potência.

O trabalho de Lima (2010) apresenta uma ampla fundamentação teórica e um estudo de desenvolvimento de software para medição de tempos de operação de disjuntores de alta tensão.

Leite *et al.* (2010) apresenta a utilização de realidade virtual em um sistema para treinamento de operadores a serem efetuadas na Usina Hidrelétrica de Tucuruí, salientando o conceito de aprendizagem na utilização de computadores.

Em Santos (2010) apresenta-se as diretrizes básicas para o anteprojeto de subestação bem como o estudo de viabilidade de expansão e descreve, também, características do sistema elétricos e seus equipamentos.

Paredes (2002) descreve uma integração de sistemas de supervisão, proteção e automação de subestações de energia elétrica, contemplando desde um histórico da evolução dos centros de controle aos arranjos típicos de uma subestação, e as técnicas de automação de sistemas elétricos.

Junior, Filho *et al.* (2006) apresentam a concepção, projeto e implementação de um sistema educacional de realidade virtual para o treinamento de procedimentos de manutenção de uma usina hidrelétrica.

Tam *et al.* (1999) descrevem a implantação e utilização de realidade virtual para o treinamento de operadores aplicado a *World Wide Web*, como uma alternativa viável e de baixo custo, podendo ser reutilizado em novas e futuras aplicações.

1.2.1 Linguagem de Programação C#

O C#, foi desenvolvida como linguagem padrão para desenvolvimento .Net, um *framework* criado para diminuir os problemas de compatibilidade entre as diversas tecnologias existentes para internet na época, virou um grande sucesso devido as suas vantagens e controvérsias (MACDONALD, 2010).

O *framework* .Net funciona em um ambiente estilo Java, onde o usuário tem o código fonte que após compilado é convertido em uma Linguagem Comum Intermediária (CIL ou IL), para em seguida ser enviada para um Ambiente de Execução Comum de Linguagem (*Common Language Runtime*).

A ferramenta oficial para desenvolvimento deste trabalho é o Visual Studio©, atualmente na versão 2010. Todavia, é possível criar algumas aplicações sem este ambiente de desenvolvimento, utilizando, por exemplo, um ambiente gratuito como o Mono-Project.

1.3 OBJETIVOS

O principal objetivo deste trabalho é desenvolver um programa de computador na linguagem C#, utilizando os conceitos do paradigma de orientação a objetos, que simule o intertravamento de chaves seccionadoras e a lógica das proteções de uma SE. Para isso, têm-se os seguintes objetivos específicos:

- Estudar os fundamentos da linguagem de programação C# e da orientação a objetos;
- Fazer uma revisão dos fundamentos de equipamentos de sistemas elétricos de potência e de suas proteções;
- Validar o simulador implementado mediante a avaliação do seu comportamento para diversas situações às quais não raro são vivenciadas por operadores de subestações de energia elétrica.

1.4 ORGANIZAÇÃO DO TEXTO

Este trabalho está organizado de acordo com a seguinte estrutura:

- No Capítulo 2, realiza-se uma breve descrição dos componentes que compõem uma subestação de energia elétrica de alta tensão;
- No Capítulo 3, descreve-se a implementação computacional em C#, demonstrando a modelagem dos principais componentes do sistema;

- No Capítulo 4, apresentam-se os testes de intertravamento e resultados obtidos em diversas manobras realizadas no sistema;
- No Capítulo 5, finalizando o trabalho, são apresentadas as conclusões propostas para trabalho futuro.

2 COMPONENTES DE UMA SUBESTAÇÃO

Segundo Paredes (2002), "As subestações de energia elétrica constituem um ponto do sistema elétrico de potência onde a energia é transformada, controlada e distribuída, deste modo, deve possuir ações e comandos coordenados a partir de programas e filosofias de operação, de conformidade com informações coletadas a partir dos sistemas de medição e proteção".

Nas SEs as operações a serem realizadas são definidas pelos equipamentos instalados e pelo arranjo de barramento, cada tipo de arranjo possui vantagens e desvantagens, sendo que neste trabalho será utilizada a configuração de barra dupla com cinco chaves, na qual a conexão entre as barras é feita através de um disjuntor denominado de disjuntor de amarre, conforme ilustrado na Figura 2.1.

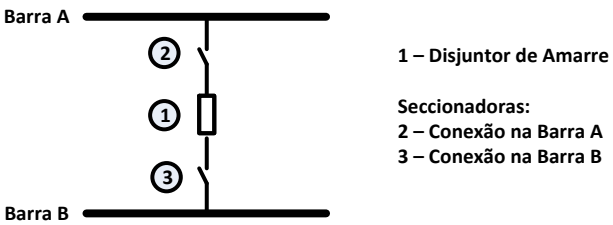

Figura 2. 1 - Barra dupla.

O sistema elétrico é trifásico, constituído de um barramento com 3 fases, que representa um nó elétrico trifásico, no qual são conectadas as linhas de transmissão (LTs), transformadores (ATs), etc. O conjunto de todos os elementos que permitem a conexão de um componente elétrico, a exemplo de LT, AT, etc., ao barramento é conhecido por *bay* (KINDERMANN, 2008).

A SE pode ser constituída de diversos barramentos, de modo a possibilitar manobras que evitem a interrupção no fornecimento de energia elétrica, em decorrência de manutenção preventiva, corretiva, inspeção ou de emergência (KINDERMANN, 2008).

Ao longo dos anos, várias configurações nos arranjos das barras foram desenvolvidas, que dependendo da configuração do arranjo, altera o grau de flexibilização da barra, obtendo-se maior continuidade no serviço oferecido.

A utilização do arranjo em barra dupla possui alto custo, devendo ser utilizado em instalações de grande porte e com existência de vários consumidores, entre outras considerações (PAREDES, 2002). Em particular, a configuração de barra dupla com cinco chaves é empregada tipicamente em SEs de 230 kV (KINDERMANN, 2008).

Conforme Mamede Filho e Mamede (2011), para sua proteção existem dois dispositivos básicos que podem ser empregados: os fusíveis e os relés.

Os fusíveis são dispositivos que atuam mediante a fusão de seu elemento metálico, dispositivo que não será alvo de estudo neste trabalho. Entretanto, os relés, segundo Mamede Filho e Mamede (2011), constituem uma grande variedade de dispositivos que oferecem proteção aos sistemas elétricos atuando nas diversas formas de anormalidades do sistema, como: sobrecarga, sobretensão, subtensão, curto-circuito, etc.

Atualmente, os relés digitais dominam o mercado, com a automação crescente dos sistemas elétricos, que são constituídos de circuitos eletrônicos com alta velocidade de processamento, que além da função de proteção, também realizam funções de controle, comunicação, acesso remoto, medidas elétricas, etc (FILHO e MAMEDE, 2011).

2.1 PROTEÇÃO DE SISTEMAS DE ENERGIA ELÉTRICA

O objetivo de um Sistema Elétrico de Potência é fornecer energia elétrica de modo adequado, confiável e sem interrupções. Na sua operação, acontecem, com certa frequência, falhas em seus componentes que causam interrupção no fornecimento de energia elétrica para os consumidores, sendo a falha mais comum o curto-circuito (FILHO e MAMEDE, 2011), que tem como resultado severos distúrbios de tensão afetando todo o sistema elétrico, muitas vezes, ocasionando danos irreparáveis ao sistema e ao consumidor.

Outros fatores que causam anormalidade são a sobrecarga, a subtensão e a sobretensão.

Segundo Mamede Filho e Mamede (2011), a principal função de um sistema de proteção é garantir o desligamento do sistema elétrico submetido a alguma anormalidade. De um modo geral, a proteção é projetada com base nos fusíveis e relés incorporados para acionamento do disjuntor, que é o equipamento que efetua a desconexão do circuito afetado pela falha.

Para a detecção de uma falha em um sistema elétrico é necessário analisar um dos seguintes critérios (FILHO e MAMEDE, 2011):

- Elevação da corrente;
- Elevação e redução da tensão;
- Inversão do sentido da corrente;
- Alteração da impedância do sistema;
- Comparação de módulo e ângulo de fase na entrada e na saída do sistema.

Segundo Paredes (2002), as concessionárias buscam automatizar ao máximo as subestações, de modo assistidas ou não assistidas, centralizando suas operações.

Os dispositivos de proteção são compostos por elementos organizados que verificam a condição do sistema e com as variáveis observadas no funcionamento do sistema, é possível efetuar as ações necessárias em caso de anormalidades (ESPINOZA, 2011).

O intertravamento de chaves seccionadoras torna-se necessário para impedir ao máximo a ocorrência de manobras indevidas, que podem culminar, em danos materiais e risco humano, além de ocasionar interrupções no fornecimento ao consumidor.

A realização de treinamento do pessoal responsável pela manutenção é um grande desafio, pois uma subestação não pode ser paralisada, tendo como consequência prejuízo financeiro, material ou de segurança das pessoas envolvidas, e possuem muita complexidade. Cujo treinamento não pode ser realizado somente por meio de documentos impressos, pois a experiência prática é imprescindível para permitir que novos funcionários conheçam o modo real utilizado na prática, inclusive conhecendo sua estrutura física (JUNIOR, FILHO, *et al.*, 2006).

Com a utilização de simuladores de sistema de proteção, pode-se aperfeiçoar o treinamento dos operadores, sem a necessidade de manipular uma subestação real, e com a mesma complexidade de um sistema digital de automação (MOUTINHO, 2010; TAM *et al.*, 1999).

2.2 EQUIPAMENTOS TÍPICOS INSTALADOS EM UMA SE

Neste tópico serão relacionados os equipamentos e relés que serão utilizados na SE hipotética para o desenvolvimento do sistema computacional.

2.2.1 Disjuntores

Disjuntores são dispositivos mecânicos de manobra utilizados nos circuitos elétricos capaz de estabelecer, conduzir e interromper a corrente elétrica sob condições normais e anormais, tais como um curto-circuito, com a finalidade de proteger equipamentos, linhas de transmissão e outros circuitos conectados além do ponto de sua instalação (LIMA, 2010), conforme Figura 2.2.

Figura 2. 2 - Disjuntor a pequeno volume de óleo, fonte: Lima (2010).

Em condições normais, os disjuntores permanecem com seus contatos fechados, permitindo que o fluxo de potência necessária aos circuitos conectados a este trafegue normalmente. A resistência dos contatos é na ordem de μΩ, portanto, o disjuntor é visto pelo circuito como uma via livre e não representa uma carga a mais ao sistema.

Em condições anormais, o disjuntor atua abrindo os seus contatos, de modo a eliminar a falta, protegendo os equipamentos e as linhas de transmissão localizadas dentro da sua área de atuação.

Segundo Lima (2010), interromper correntes de curto-circuito é a tarefa mais crítica do disjuntor, pois durante a abertura dos contatos são gerados arcos elétricos que dissipam grandes quantidades de energia, por efeito joule, e temperaturas com valores da ordem de 50000 ºC.

2.2.2 Chaves Seccionadoras

As chaves seccionadoras são dispositivos que efetuam a conexão e desconexão de várias partes de um circuito elétrico para efetuar manobras de operação e manobras de manutenção. Podem abrir circuitos com tensão abaixo da nominal e sem condução de corrente. Antes de efetuar uma operação de abertura é necessariamente abrir primeiramente os disjuntores, pois as seccionadoras não possuem capacidade de abrir um circuito em carga (SANTOS, 2010), conforme Figura 2.3.

Figura 2. 3 - Chave Seccionadora, fonte: Catálogo Weg.

2.2.3 Auto-Transformadores

O auto-tranformados é um dos equipamentos mais complexos e de mais alto custo em uma subestação, também denominado transformador de potência. São equipamentos capazes de elevar ou reduzir a tensão e são utilizados desde as usinas de produção, onde a tensão gerada é elevada para que ocorra a diminuição da perda na transmissão, até os pontos de consumo, normalmente uma subestação, onde ocorre a redução da tensão a níveis adequados para distribuição (VASQUES, 2011), conforme Figura 2.4.

Figura 2. 4 - Autotransformado, fonte: Vasques (2011).

2.2.4 Relés de Proteção

Os relés são dispositivos que oferecem proteção aos sistemas elétricos em caso de anormalidade no sistema elétrico. Cada tipo de relé de proteção possui características técnicas que definem sua função no sistema elétrico, configurado dentro dos limites de sua atuação no esquema de proteção e controle (FILHO e MAMEDE, 2011), conforme Figura 2.5.

Figura 2. 5 - Relé de Proteção, tipo SEL-311C , da Schweitzer, fonte: http://www.selinc.com.br/produtos/SEL-311C.aspx em 01/08/2012.

A seguir, apresenta-se uma breve descrição dos relés que serão utilizados no simulador implementado nesse trabalho, a numeração representa a designação da função exercida pelos elementos, aparelhos e dispositivos utilizados nos circuitos elétricos de acordo com a padronização atual C37-2 da IEEE/ANSI (KINDERMANN, 2008):

- Relé de distância (21): É um relé que opera no aumento ou diminuição da impedância, admintância ou a reatância (KINDERMANN, 2008);

- Relé de sincronismo (25): Têm a função de comparar a frequência entre duas fontes de geração, que se deseja trabalhar em paralelo (FILHO e MAMEDE, 2011), é um dipositivo obrigatório neste caso. O relé de sincronismo compara os módulos das diferenças máximas entre das tensões, das frequências de fases e das defasagens angulares das fontes;

- Relés de sobrecorrente (50/51): Segundo Mamede Filho e Mamede (2011), todos os segmentos dos sistemas elétricos são protegidos por relés de sobrecorrente, que é um dispositivo que responde a corrente elétrica quando o módulo da corrente ultrapassa um valor pré-determinado;

- Relé de sobretensão (59): São utilizados na proteção de sistemas elétricos que atuam em quando o nível de tensão ultrapassa os valores máximos permitidos pelos equipamentos (FILHO e MAMEDE, 2011);

- Relé de fechamento ou de abertura temporizada (62): Segundo Kindermann (2008), é um relé que opera em conjunto com o dispositivo, de forma temporizada, uma sequência automática, que dá início a operação de fechamento, paralização ou abertura de relés de proteção;

- Relé diferencial de corrente (87): Segundo Mamede Filho e Mamede (2011), efetua a comparação entre as correntes que circulam no sistema ou nos terminais de um equipamento, em caso de diferença do módulo da corrente, é enviado um sinal para atuação do disjuntor;

- Relé auxiliar de bloqueio (86): São dispositivos utilizados na proteção para assegurar a integridade dos equipamentos de força e a segurança dos operadores (FILHO e MAMEDE, 2011).

2.3 ESTADOS DE OPERAÇAO DO SISTEMA DE POTÊNCIA

Os estados de operação definem os modos de operação que podem estar ocorrendo em uma SE, pois nem sempre é possível oferecer total disponibilidade em todos os equipamentos, sendo representado da seguinte forma:

- Estado de operação NORMAL: Trabalhar neste estado é o objetivo dos operadores, ocorre quando as condições de igualde e desigualdade estão satisfeitas, os indicadores de tensão, frequência da demanda estão atendidas e todos os componentes estão trabalhando dentro de limites especificados (PAREDES, 2002);

- Estado de operação EMERGÊNCIA: Quando ocorre alguma condição de indicadores fora dos limites especificados, por exemplo, a sobrecarga de algum componente, ou quando a qualidade de energia de fornecimento não pode ser mantida ocorre a operação em estado de operação de emergência (PAREDES, 2002). Caso não sejam realizadas medidas de controle todo o sistema poderá se deteriorar, Portanto, o controle é necessário para minimizar as inconveniências para os consumidores;

- Estado de operação RESTAURATIVO: Este estado representa todo o processo realizado para alcançar a restauração do sistema ao estado de operação normal.

2.4 O PROCESSO DE OPERAÇÃO EM TEMPO REAL

O processo de operação do sistema elétrico é realizado por meio de monitoramento e supervisão de grandezas, elétricas e não-elétricas do sistema, que visam manter o padrão de qualidade e continuidade no fornecimento de energia elétrica. Esse processo pode ser dividido em duas etapas: Sistêmica e Local.

A sistêmica supervisiona todas as principais grandezas de todas as SEs e usinas que compõem o sistema elétrico, fazendo com que mantenha uma condição segura de operação.

Segundo Ferreira (2007), a operação em nível local é realizada na SE, em complemento às ações definidas pelos centros de operação. São implementadas através de sistemas convencionais, digitais ou híbridos, sendo que as pricipais funções executadas em nível local passíveis de serem automatizadas na SE são:

- Proteção por falha de disjuntor: Ao ocorrer um curto circuito em uma LT, a proteção atua e ocorre a abertura do disjuntor. Por razões mecânicas ou elétricas pode ocorrer que a corrente elétrica não seja interrompida. Neste caso, o relé 51BF conta o tempo desde a ordem de abertura até o tempo de abertura esperado, e detecta a falha de abertura. Neste caso, deve ocorrer as seguintes ações: primeiro, abrir todos os disjuntores dos circuitos ligados a barra onde está ligado o disjuntor que ocorreu a falha, e após, abrir automaticamente as seccionadoras adjascentes a este disjuntor, isolando-o do circuito (FERREIRA, 2007);
- Intertravamento: Essa função visa estabelecer condições à manobra de seccionadoras, visto que não podem operar em carga, porém são utilizados para permitir a ligação de um circuito em uma ou outra barra e para isolar o disjuntor e o autotransformador, de modo que se possa efetuar as manutenção dos mesmos em segurança. Essa função deve implementar uma lógica de intertravamento para as seccionadoras com o objetivo de permitir manobras seguras, do ponto de vista operacional (FERREIRA, 2007);
- Sequência automática de chaveamentos: Essa função deve implementar uma lógica para manobras padronizadas, de forma que uma vez acionado o comando inicial, todas as ações seguintes devem ser realizadas automaticamente até o final (FERREIRA, 2007).

3 IMPLEMENTAÇÃO COMPUTACIONAL

O diagrama unifilar ilustrado na Figura 3.1 é uma representação dos componentes do sistema elétrico da SE utilizada nesse estudo, a qual possui três barramentos de níveis de tensão diferentes, a saber: 138, 230 e 345 kV. Cada um deles possui a configuração de barramento duplo com cinco chaves. Nessa configuração, utiliza-se um disjuntor de amarre para interligar as barras A e B de um mesmo nível de tensão e chaves seccionadoras seletoras de barra para interligar as linhas de transmissão (LTs) e os autotransformadores (ATs) ao barramento. Cada seccionadora possui uma lógica de intertravamento, assim como cada LT, AT e barramento possui uma lógica de proteção, as quais foram consideradas na modelagem desses componentes no simulador. Nas Tabelas 3.1 e 3.2 são apresentadas todas as LTs e os ATs conectados a cada um dos barramentos da SE, indicando a barra (A ou B) na qual são ligados na operação normal da SE. A seguir, descreve-se os detalhes da implementação dos componentes do simulador.

Tabela 3. 1 - LTs conectadas a cada um dos barramentos da SE.

Nível de Tensão (kV)	**Nome**	**Barra de Conexão Na Operação Normal**
345	LT1_345kV	A
	LT2_345kV	B
230	LT1_230kV	A
	LT2_230kV	B
138	LT1_138kV	A
	LT2_138kV	B

Tabela 3. 2 - ATs conectados aos barramentos da SE.

Sigla	**Barramentos de Conexão**	**Barra de Conexão na Operação Normal**
AT01	345 kV	A
	138 kV	A
AT02	345 kV	B
	230 kV	A

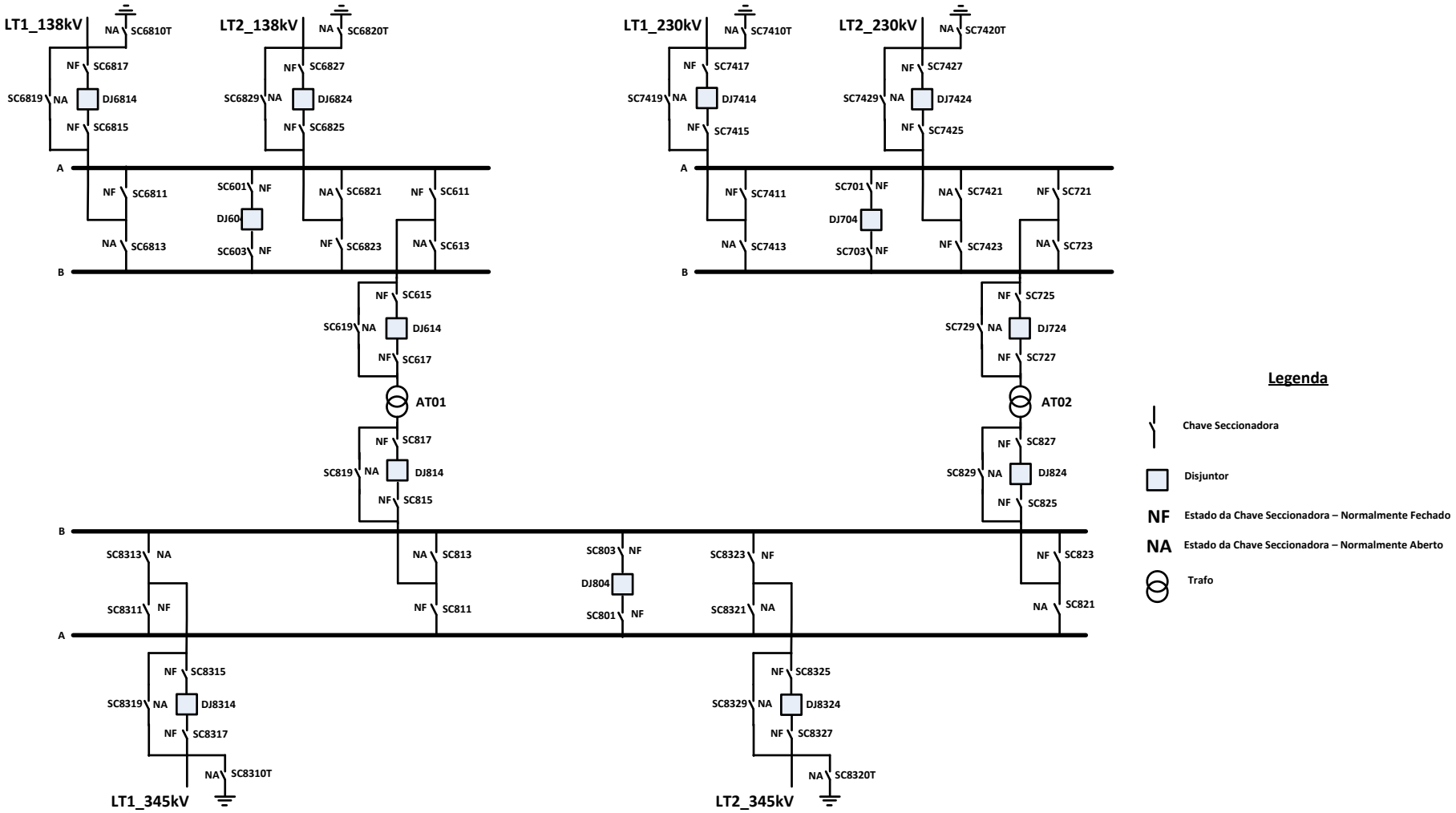

Figura 3. 1 - Diagrama Unifilar da Subestação.

3.1 ESTRUTURA BÁSICA DA IMPLEMENTAÇÃO

A classe fundamental para a implementação do simulador é denominada `ObjetoSimulador`. Todas as demais classes serão derivadas dela através de herança pública (SEBESTA, 2003), conforme diagrama de classe da Figura 3.3. Na Figura 3.2 apresenta-se a declaração desta classe.

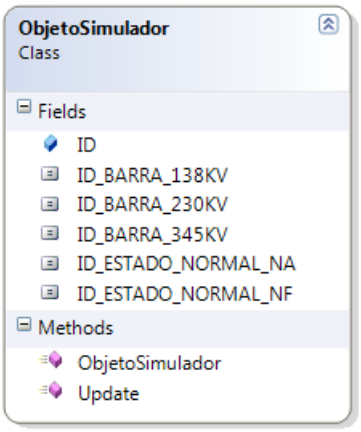

Figura 3. 2 - Classe `ObjetoSimulador`

Como se pode observar, a classe `ObjetoSimulador` é bastante simples. Basicamente, ela define um identificador (ID) e constantes que serão utilizadas ao decorrer da implementação: IDs das barras e do estado normal de operação das chaves seccionadoras. O ID do objeto é passado pelo construtor da classe derivada para o construtor da classe `ObjetoSimulador`.

Como será descrito em seções futuras, a instanciação de todos os objetos que compõem o simulador, bem como a definição de todas as relações entre eles, será feita em uma classe denominada `Simulador`. De fato, esta classe será responsável pela montagem de toda a estrutura do simulador.

Sendo todas as classes derivadas da classe `ObjetoSimulador`, cada objeto instanciado pode ser adicionado a um dicionário com todos os objetos do simulador. Este dicionário é declarado como uma propriedade da classe `Simulador`, com nome `dicionarioCompleto` e tipo `Dictionary`<> da linguagem C# (LIBERTY e XIE, 2009). No construtor da classe base `ObjetoSimulador` é incluída a chamada ao método estático público `AddObjectDicionarioCompleto()` da classe `Simulador`, passando como argumento o ID do objeto. Assim, cada vez que um objeto é instanciado, o

construtor da sua classe faz uma chamada ao construtor da classe base `ObjetoSimulador`, resultando na inclusão desse objeto no `dicionarioCompleto`. Isto garante que o ID de cada objeto seja único, o que facilita sobremaneira a integração do simulador, uma vez que o ID de um objeto pode ser usado como a chave de sua busca no `dicionarioCompleto`.

O método `Update()` da classe `ObjetoSimulador` é definido como virtual, para que ele seja sobrescrito em cada uma das classes derivadas, a fim de particularizar a atualização das propriedades de cada uma delas.

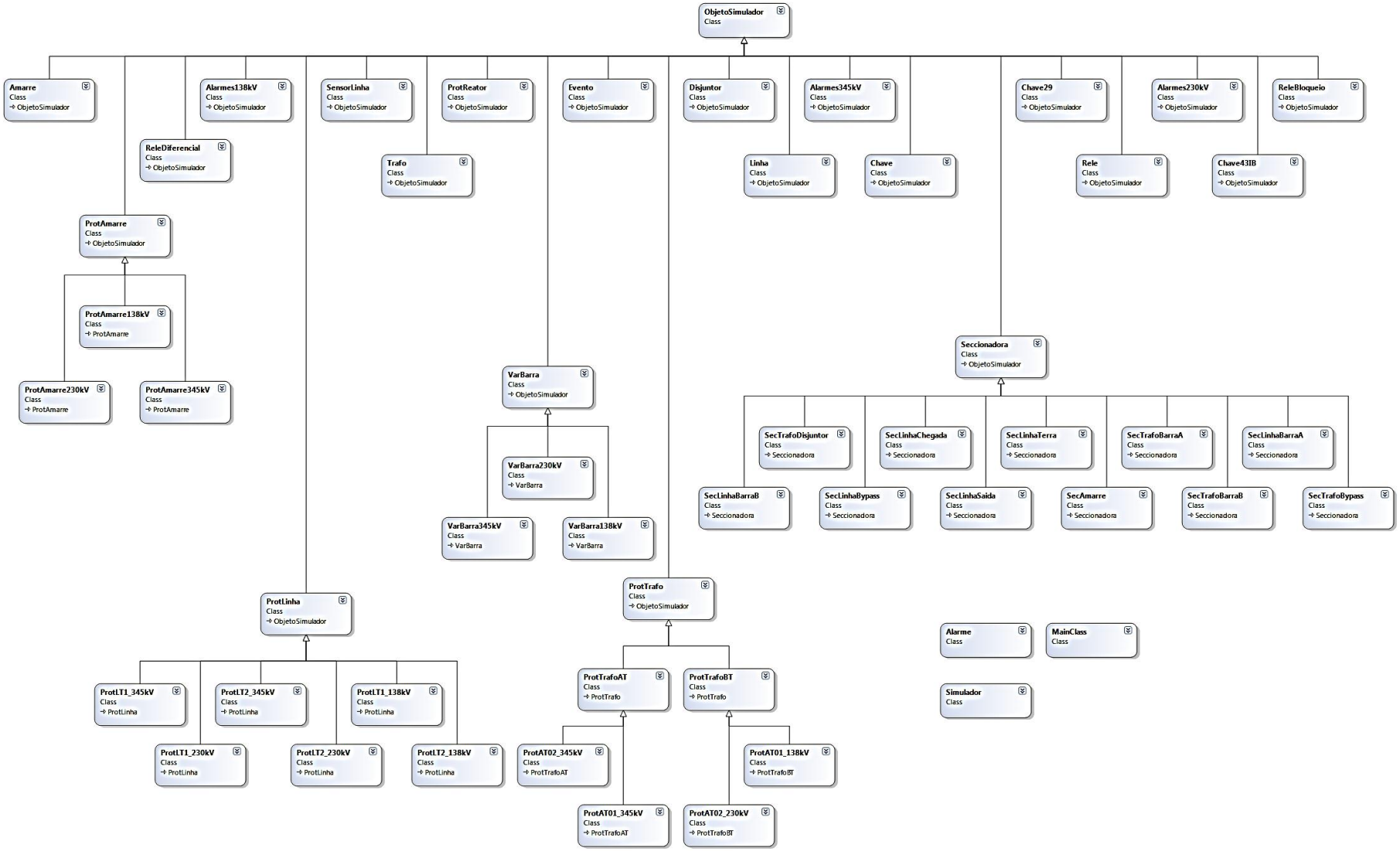

Figura 3. 3 - Diagrama de Classes.

3.1.1 Modelagem da Classe Simulador

Esta classe, conforme Figura 3.4, é responsável pela montagem de toda a estrutura do simulador, contendo todas as informações de estado dos componentes da SE adicionado a um dicionário, como descrito anteriormente.

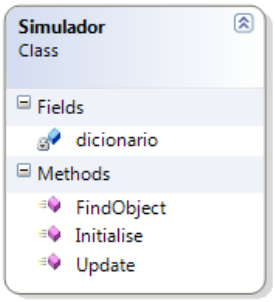

Figura 3. 4 - Classe `Simulador`.

Nesta classe são instanciados os objetos que compõem o dicionário, contendo toda estrutura e estados dos componentes da SE, e criado o primeiro evento, `EV0001`, com os estados iniciais das chaves seccionadoras e dos disjuntores de toda SE.

Os atributos especificados para as seccionadoras das LTs conectados a barra são:

`ID` – identificação da seccionadora.

`estado` – estado da seccionadora, sendo `true` para fechado e `false` para aberto.

`IDLinha` – identificação da LT.

`IDVariaveisBarra` – identificação das variáveis da barra.

Os atributos especificados para os disjuntores das LTS são:

`ID` – identificação do disjuntor.

`estado` – estado do disjuntor, sendo `true` para fechado e `false` para aberto.

Os atributos especificados para os relés de proteção das LTs são:

`ID` – identificação do relé

`estado` – estado do relé, sendo `true` para fechado e `false` para aberto.

A figura 3.5 representa o Diagrama de Estados da classe `Simulador` e a descrição do código no Apêndice A.

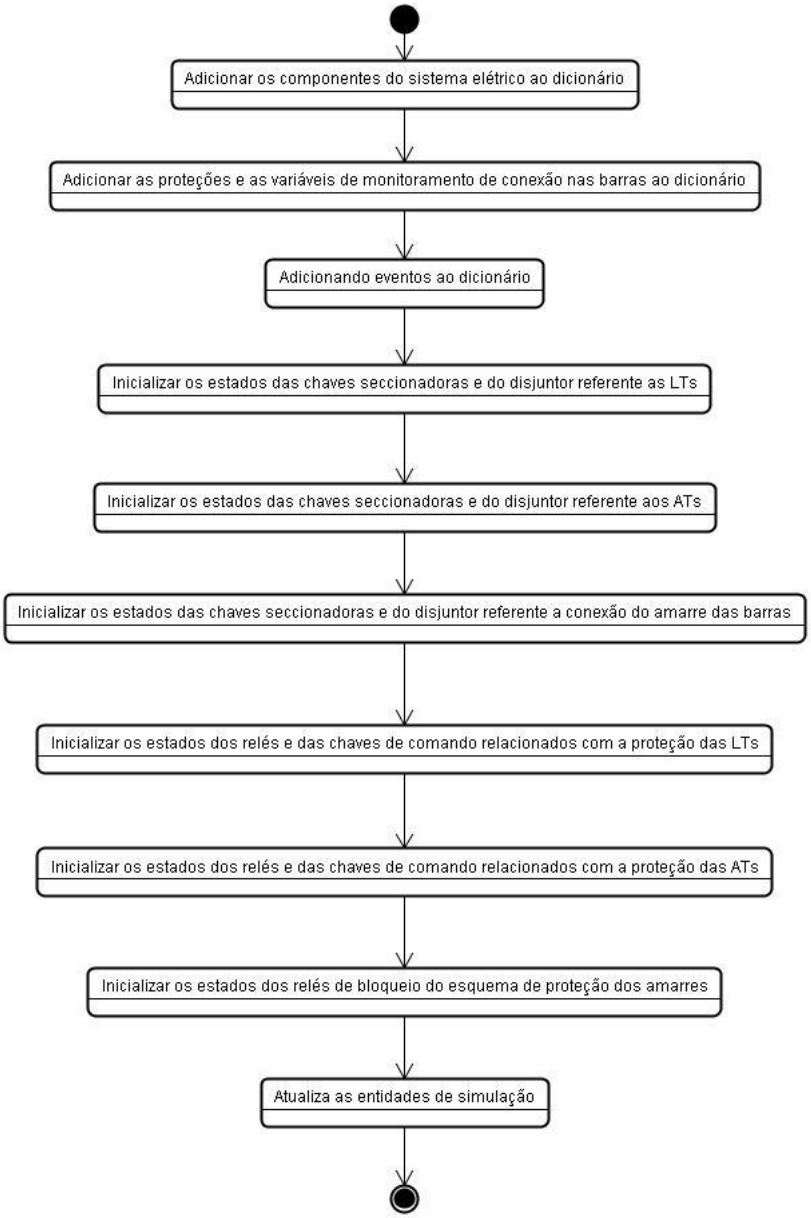

Figura 3. 5 - Diagrama de Estados da classe `Simulador`.

3.1.2 Modelagem da Classe Evento

Na classe `Evento`, conforme Figura 3.6, são modelados os testes de intertravamento e manobras, que também herda da classe `ObjetoSimulador` por herança pública os estados dos componentes da SE. A declaração `case` define o teste ou manobra realizada no trabalho ocorrendo várias condições de mudança de estado simulando operação de proteção da SE, conforme o objeto em estudo.

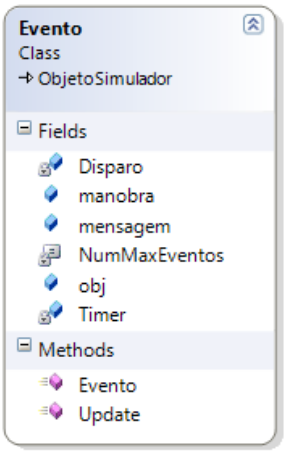

Figura 3. 6 - Classe `Evento`

Para a realização dos testes e manobras foi utilizado uma máquina de estados onde o sistema possui uma quantidade finita de entradas e produz um número finito de diferentes saídas. Cada etapa de um `case` é definido por um `Disparo` e o tempo de execução de cada etapa é proporcional ao valor definido em `NumMaxEventos`. Dessa forma são geradas informações de estados dos componentes da SE no decorrer do tempo e com alterações periódicas ao passar para outra etapa de forma ininterrupta. A utilização da máquina de estados é necessária para que o sistema possa se atualizar a nova condição imposta pela mudança de estado de algum componente.

No Apêndice B, descrevem-se os códigos em C# da máquina de estados e da manobra de atuação da proteção da LT1_345kV e a normalização do sistema, conforme diagrama de estados da Figura 3.7, e do teste de intertravamento de abertura de chaves seccionadoras, com a proteção da LT2_345kV transferida, conforme diagrama de estados da Figura 3.8.

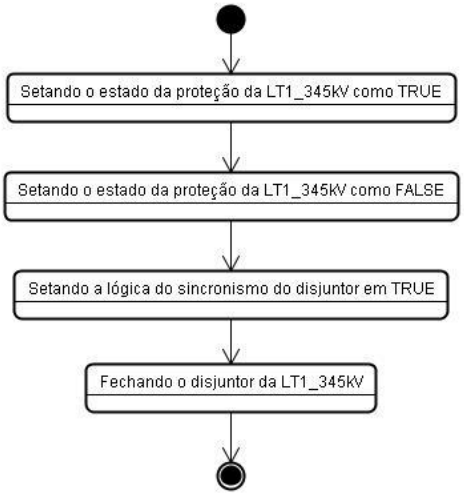

Figura 3. 7 - Diagrama de estados da manobra de atuação da proteção da LT1_345kV e a normalização do sistema, da classe `Evento`.

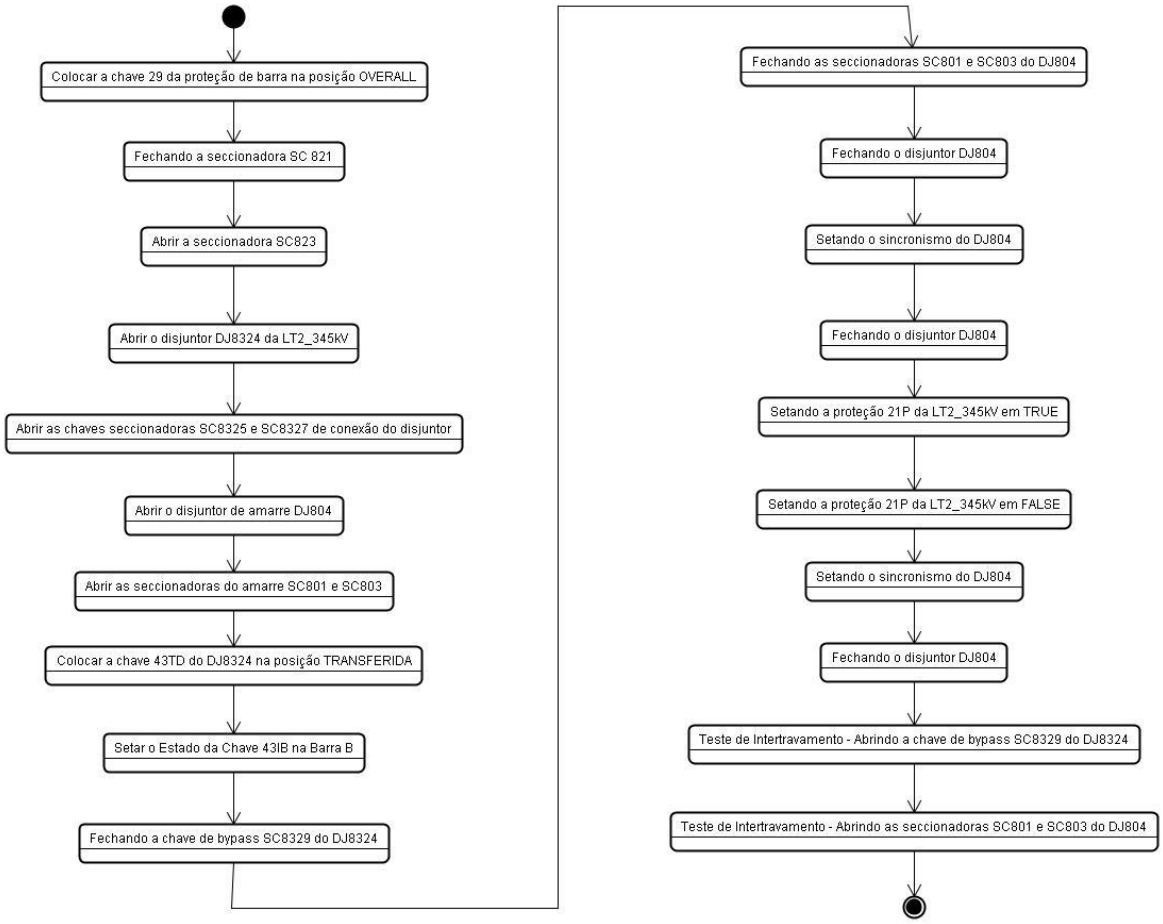

Figura 3. 8 - Diagrama de estados do teste de intertravamento de abertura de chaves seccionadoras, com a proteção da LT2_345kV transferida, da classe `Evento`.

3.1.3 Modelagem da Classe Main

Na classe `Main`, conforme Figura 3.9, são modelados os acessos ao simulador através de linha de comando, conforme Figura 3.10. Estão descritas todas as manobras e testes de intertravamento efetuados e a apresentação do resultado na tela e em arquivo texto.

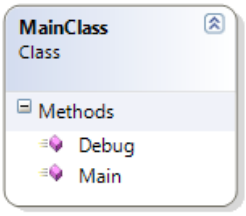

Figura 3. 9 - Classe `Main`.

Figura 3. 10 - Menu de opções de manobras.

No Apêndice C, apresenta-se o código em C# da classe `Main`, com a apresentação de duas declarações apresentadas na classe `Evento,` uma manobra e um teste de intertravamento, sendo que as demais manobras e testes ocorrem de forma similar.

3.1.4 Modelagem para Acréscimo de Manobras

Para a modelagem de novas manobras, será descrito passo-a-passo a declaração `case 0`, da classe `Evento`, que simula a atuação da proteção da LT1_345kV em um curto-circuito na LT e a restauração do sistema para sua condição normal. De modo similar é feito o acréscimo de novas manobras no simulador.

O primeiro passo da manobra consiste em setar o estado da proteção da LT1_345kV como *TRUE*, simulando um curto-circuito na LT. Na modelagem, ocorre a busca do objeto `ProtLT1_345kV` na classe `Simulador` e define o estado da variável `rele21P` como `true`.

```
            case 0:
                if (Timer[0] < 0 && !Disparo[0])
                {
                    //Setando o estado da proteção da LT1_345kV como TRUE
                    (Simulador.FindObject("ProtLT1_345kV") as
ProtLT1_345kV).rele21P.Estado = true;
                    mensagem = "1 - Estado da proteção TRUE";
                    Disparo[0] = true;
                }
```

Uma vez detectado o curto-circuito e consequentemente a abertura do disjuntor, no próximo passo inicia os procedimentos de normalização do sistema. Primeiramente, retornando o estado do `rele21P` para `FALSE`.

```
                if (Timer[1] < 0 && !Disparo[1])
                {
                    //Setando o estado da proteção da LT1_345kV como FALSE
                    (Simulador.FindObject("ProtLT1_345kV") as
ProtLT1_345kV).rele21P.Estado = false;
                    mensagem = "2 - Estado da proteção FALSE";
                    Disparo[1] = true;
                }
```

A seguir, é necessário efetuar o sincronismo do disjuntor, portanto, na modelagem ocorre a busca do `disjuntor` no objeto da `LT1_345kV` da classe `Simulador` e altera o estado do `Sincronismo` para `true`.

```
                if (Timer[2] < 0 && !Disparo[2])
                {
```

```
                    //Setando a lógica do sincronismo do disjuntor em TRUE
                    (Simulador.FindObject("LT1_345kV") as
Linha).disjuntor.Sincronismo = true;
                    mensagem = "3 - Sincronismo em TRUE";
                    Disparo[2] = true;
                }
```

No último passo dessa manobra, ocorre o fechamento do disjuntor, na modelagem ocorre abusca do `disjuntor` no objeto da `LT1_345kV` da classe `Simulador` e efetua-se o comando `Fechar`.

```
                if (Timer[3] < 0 && !Disparo[3])
                {
                    //Fechando o disjuntor da LT1_345kV
                    (Simulador.FindObject("LT1_345kV") as
Linha).disjuntor.Comando = Disjuntor.Comandos.Fechar;
                    mensagem = "4 - Fechando o disjuntor";
                    Disparo[3] = true;
                }
                break;
```

Uma vez efetuada as mudanças na classe `Evento`, é necessário acrescentar as mudanças na classe Main, que é a classe que inicializa o simulador e faz a interface com o usuário, por meio de linha de comando.

Para o `case 0`, primeiramente é necessário criar uma linha no *menu* que será apresentado ao usuário, como descrito abaixo:

```
            System.Console.WriteLine("   1 - Proteção NORMAL + Atuação do relé
21P da LT1_345kV");
```

A linha a seguir, permite que os dados de saída sejam armazenados em arquivo de texto, ao mesmo tempo em que são apresentados na tela do computador.

```
        StreamWriter arqsaida1 = new StreamWriter("C:\\Manobras\\Manobra1.txt",
true, Encoding.ASCII);
```

A seguir, na declaração `case 0`, descreve-se a informação que estará disponível ao usuário e armazenada no arquivo texto, neste caso, o estado do `disjuntor`, o estado do `Sincronismo` e o estado do `rele21p`

```
            case 0:
                linha = Evento.mensagem + " | Disjuntor: |" +
(Simulador.FindObject("LT1_345kV") as Linha).disjuntor.Estado + " | " +
                                        "Sincronismo: |" +
(Simulador.FindObject("LT1_345kV") as Linha).disjuntor.Sincronismo + " | " +
                                        "Rele21P: |" +
(Simulador.FindObject("ProtLT1_345kV") as ProtLT1_345kV).rele21P.Estado;
                arqsaida1.WriteLine(linha);
                System.Console.WriteLine(linha);
                break;
```

E por fim, o fechamento do arquivo texto gerado.

```
arqsaida1.Close();
```

Com esses procedimentos é possível criar novas manobras e também alterar a informação que será disponibilizada ao usuário.

3.2 MODELAGEM DOS COMPONENTES DO SISTEMA ELÉTRICO

A fim de implementar o simulador da SE, faz-se necessário a modelagem de cada componente do seu sistema elétrico utilizando o paradigma da orientação a objetos. A partir da análise do diagrama unifilar ilustrado na Figura 3.1, percebe-se que os componentes principais do sistema elétrico da SE são os barramentos, as LTs e os ATs. Em cada barramento, a conexão entre as barras A e B é feita através de um disjuntor de amarre, conforme ilustrado na Figura 3.11. Há uma configuração padronizada para a conexão de todas LTs e ATs aos barramentos, conforme ilustrado nas Figuras 3.12a e 3.12b. Conforme se pode observar, na configuração de barramento duplo com disjuntor simples, LTs e ATs são conectados aos barramentos através de duas chaves seccionadoras de seleção de barra. No caso das LTs, também são utilizadas uma seccionadora de *bypass* do disjuntor, uma de chegada de LT, uma de saída para o barramento e uma de terra. Na conexão dos ATs, por sua vez, além das seccionadoras de seleção de barra, são utilizadas uma de *bypass* do disjuntor, uma de chegada do AT e uma de saída para o barramento.

Com base nas configurações descritas acima, percebe-se que barramentos, LTs e ATs podem ser modelados a partir da combinação de componentes elementares, que são os disjuntores e as chaves seccionadoras. Nesse sentido, foram implementadas as classes `Amarre`, `Linha` e `Trafo`, como agregações de objetos das classes `Disjuntor` e `Seccionadora`. A seguir, apresenta-se a descrição de cada uma dessas classes.

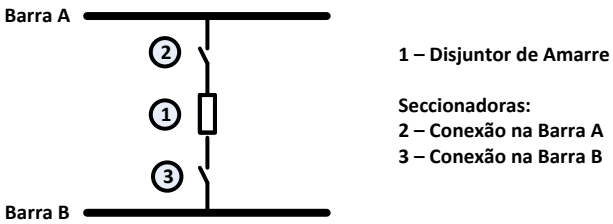

Figura 3. 11 - Conexão das Barras A e B de um mesmo barramento via disjuntor de amarre.

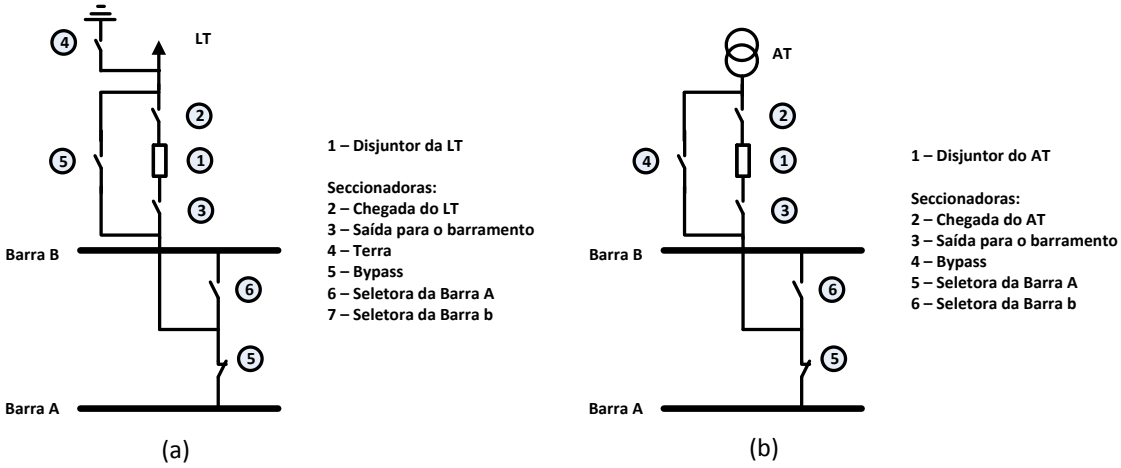

Figura 3. 12 - Conexão de um elemento em um barramento com configuração de barramento duplo com configuração simples: (a)LT e (b)AT.

3.2.1 Modelagem dos Disjuntores

O disjuntor foi modelado a partir de uma classe denominada `Disjuntor`, como sendo, basicamente, uma chave que possui dois estados: aberto ou fechado. O estado pode ser alterado a partir de uma variável de comando. Esta classe, conforme Figura 3.13, possui as seguintes propriedades:

1. ID – Usada para definir um identificador único para o disjuntor. De fato, essa variável é herdada da classe base `ObjetoSimulador`, de modo que o valor atribuído a ela é passado para o construtor da classe `ObjetoSimulador` diretamente no construtor da classe `Disjuntor`.
2. estado – Usada para definir o estado do disjuntor: fechado (`true`) ou aberto (`false`).

3. sincronismo – Usada para monitorar o sincronismo das tensões nos terminais do disjuntor antes do seu fechamento: sincronizado (`true`) ou não sincronizado (`false`).
4. falha – Usada para definir o estado de operação do disjuntor: operação em falha (`true`) ou operação normal (`false`). Essa variável deve ser modificada para `true` quando da avaliação dos cenários de falha do disjuntor.
5. logicaFechamento – Usada para verificar se as condições para o fechamento do disjuntor estão satisfeitas: permissão (`true`) ou bloqueio (`false`).
6. comando – Usada para enviar um comando para o disjuntor: Abrir,Fechar e Neutro. Estes comandos serão enviados ao disjuntor pelas classes `Amarre`, `Linha` e `Trafo`. Assim, não é necessário ter um tipo de disjuntor para cada um desses elementos.

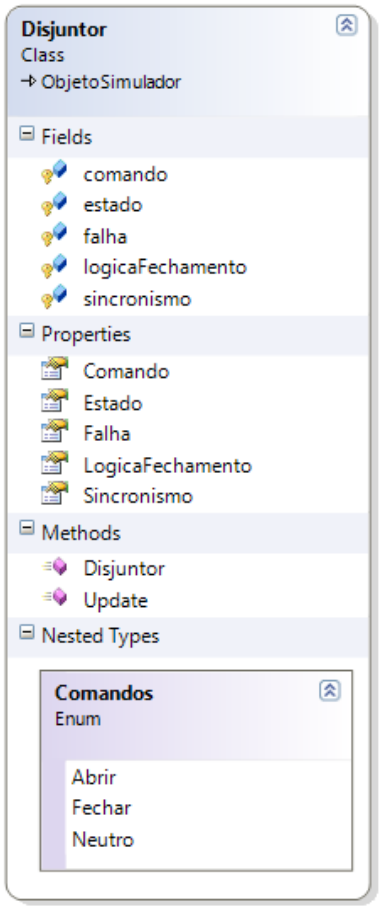

Figura 3. 13 - Classe `Disjuntor`.

A Figura 3.14 representa o Diagrama de Estados da classe `Disjuntor` e a descrição do código no Apêndice D.

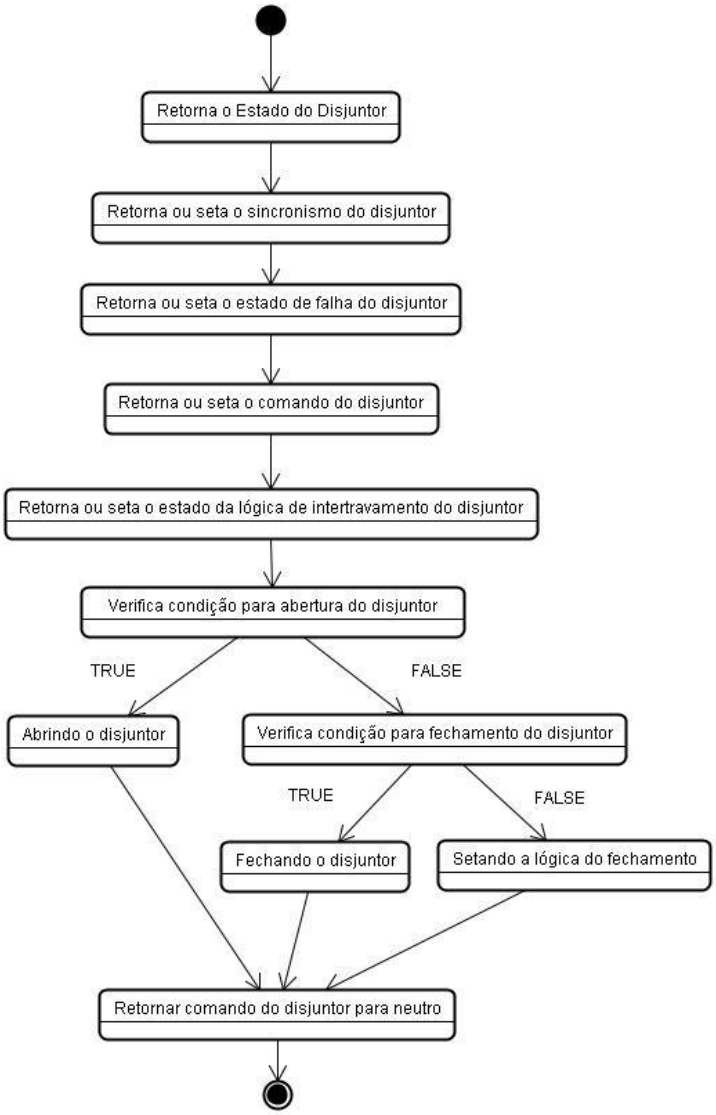

Figura 3. 14 - Diagrama de estados da classe `Disjuntor`.

Observando-se o método `Update()` da classe `Disjuntor`, verifica-se que para abrir o disjuntor é necessário que ele esteja fechado, não esteja em falha e que um comando de abertura seja recebido.

Por outro lado, para fechar o disjuntor, além de receber o comando de fechamento, é preciso que ele esteja aberto, que suas condições de fechamento sejam satisfeitas e que haja sincronismo entre as tensões nos seus terminais.

Caso nenhum comando seja enviado para disjuntor ou as demais condições não sejam satisfeitas, apenas atribui-se `false` à lógica de fechamento do disjuntor.

Após a avaliação do comando de abertura ou fechamento do disjuntor, a variável `comando` volta ao estado `Neutro`. Isso é necessário para representar corretamente o funcionamento da chave de comando do disjuntor localizada nos painéis de comando da SE.

3.2.2 Modelagem das Chaves Seccionadoras

Assim como um disjuntor, uma chave seccionadora pode ser modelada simplesmente como uma chave que possui dois estados: aberto ou fechado, que pode ser alterado a partir de uma variável de comando. Contudo, cada seccionadora instalada na SE possui uma lógica de intertravamento que permite ou não a sua manobra, ou seja, habilita ou não a mudança de seu estado. Esta lógica de intertravamento depende do tipo de seccionadora e, em alguns casos, do barramento ao qual o componente a que ela pertence esteja conectado. Nesse sentido, implementou-se uma classe `Seccionadora` que herda da classe `ObjetoSimulador` por herança pública e que simplesmente modela uma chave simples de dois estados. As particularidades de cada lógica de intertravamento são implementadas em classes derivadas por herança pública da classe `Seccionadora`, conforme será apresentado na descrição da modelagem dos barramentos, LTs e ATs. A classe `Seccionadora`, conforme Figura 3.15, possui as seguintes propriedades:

1. ID – Usada para definir um identificador único para a seccionadora. De fato, essa variável é herdada da classe base `ObjetoSimulador`.
2. estado – Usada para definir o estado da seccionadora: fechada (`true`) ou aberta (`false`).
3. comando – Usada para enviar um comando para a seccionadora: `Abrir`,`Fechar` e `Neutro`. Estes comandos serão enviados à chave pelas classes derivadas, de modo que ela só seja manobrada caso a sua lógica de intertravamento for satisfeita.

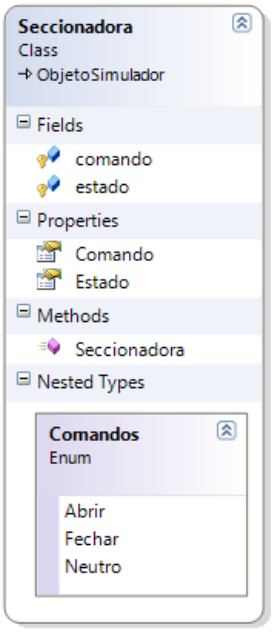

Figura 3. 15 - Classe `Seccionadora`

A Figura 3.16 representa o Diagrama de Estados da classe `Seccionadora` e a descrição do código no Apêndice E .

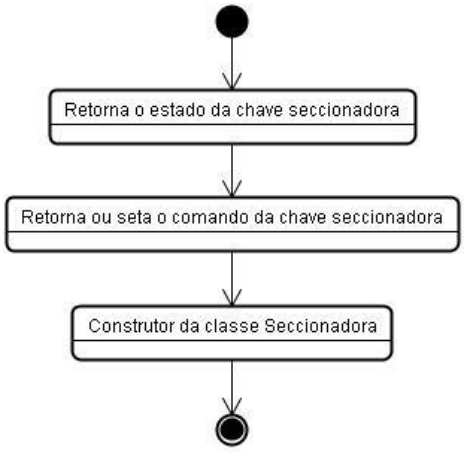

Figura 3. 16 - Diagrama de estados da classe `Seccionadora`.

3.2.3 Modelagem dos Barramentos

Os barramentos da SE são modelados, simplesmente pelo seu amarre, ou seja, pela combinação de um disjuntor de amarre e duas chaves seccionadoras seletoras de barra, conforme ilustrado na Figura 3.11. Para representar a lógica de intertravamento dessas seccionadoras, foi implementada a classe `SecAmarre`, conforme Figura 3.17.

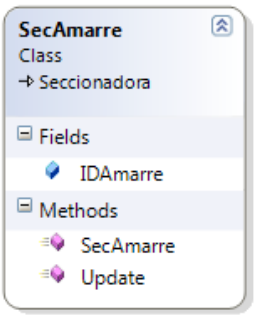

Figura 3. 17 - Classe `SecAmarre`.

A Figura 3.18 representa o Diagrama de Estados da classe `SecAmarre` e a descrição do código no Apêndice F.

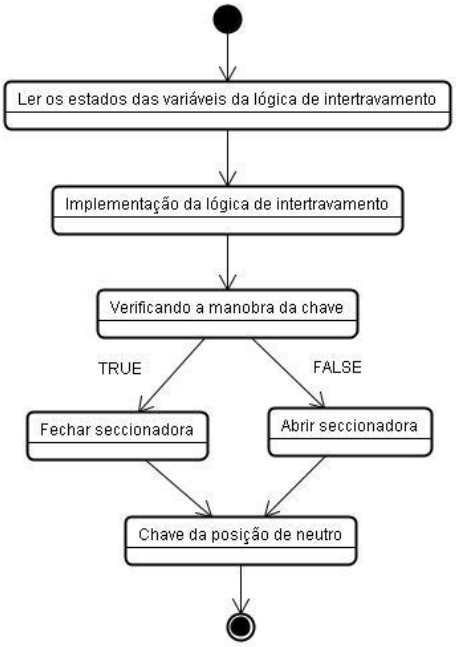

Figura 3. 18 - Diagrama de estados da classe `SecAmarre`.

Percebe-se na classe `SecAmarre` que ela é derivada da classe `Seccionadora` através de uma herança pública. Assim, além das propriedades ID, estado e comando, herdadas da classe `Seccionadora`, a classe `SecAmarre` possui uma propriedade denominada de `IDAmarre`, que armazena o ID do disjuntor de amarre ao qual a seccionadora está associada, sendo este passado como argumento de entrada no construtor da classe. Esse ID é utilizado no método `Update()` para verificar o estado do disjuntor de amarre.

Faz-se uso da função estática pública `FindObject()` da classe `Simulador`. Ela tem como resultado o objeto do dicionário cujo ID é igual ao valor da variável `IDAmarre`. Após encontrado, este objeto será acessado como sendo um objeto do tipo `Amarre`, conforme Figura 3.19. O estado do disjuntor de amarre é então é verificado e atribuído à variável local disjuntor. Esta variável é utilizada na lógica de intertravamento da seccionadora, que nesse caso é simplesmente a negação do estado do disjuntor de amarre.

Em outras palavras, a seccionadora só poderá ser manobrada, ou seja, mudar de estado, caso receba um comando de abertura ou fechamento e o disjuntor de amarre esteja aberto.

Ao final, o comando da chave seccionadora sempre deve voltar para seu estado Neutro.

Isto é necessário para representar o correto funcionamento da chave de comando da seccionadora, que está localizada nos painéis de comando da SE.

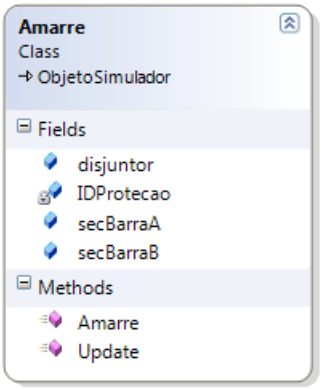

Figura 3. 19 - Classe `Amarre`.

A Figura 3.20 representa o Diagrama de Estados da classe `Amarre` e a descrição do código no Apêndice G.

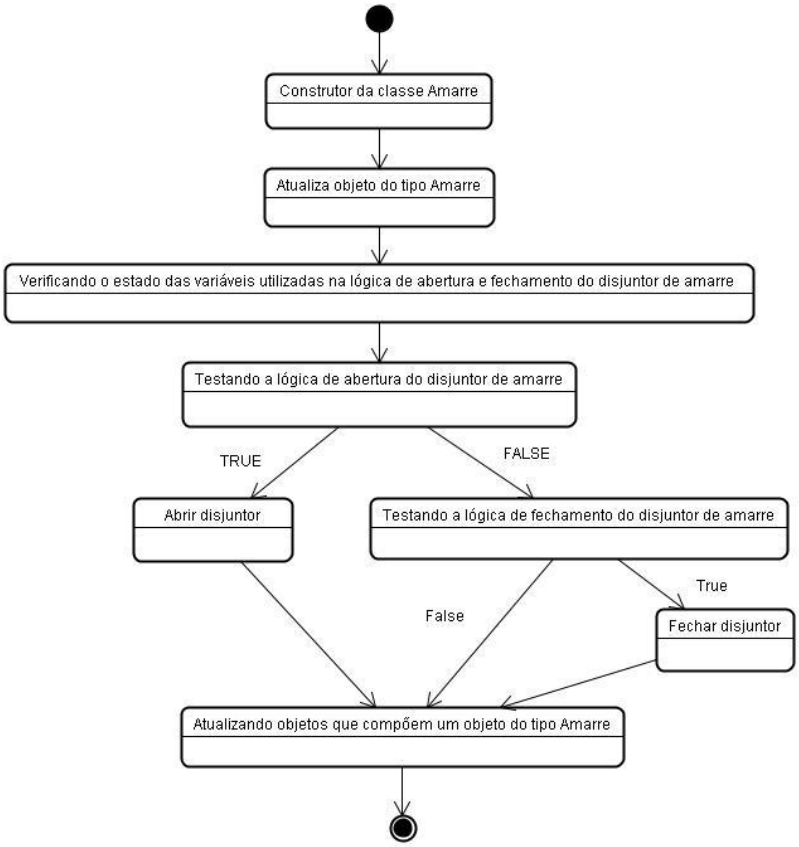

Figura 3. 20 - Diagrama de estados da classe `Amarre`.

Conforme discutido anteriormente, a classe `Amarre` é uma agregação de objetos das classes `Disjuntor` e `SecAmarre`, a fim de representar a configuração ilustrada na Figura 3.2. Isto é feito instanciando os objetos `disjuntor`, `secBarraA` e `secBarraB`.

Outra propriedade da classe `Amarre` é a variável `IDProtecao`, que armazena o ID da proteção do amarre, que deve ser passado como argumento de entrada no construtor da classe, juntamente com o ID do próprio amarre.

Na função `Update()`, inicialmente verifica-se o estado da proteção que atua diretamente no disjuntor de amarre, fazendo-se uma busca no dicionário de objetos pela proteção do amarre e acessando o estado das suas variáveis de `trip` e `bloqueio`.

Em seguida, utilizam-se as variáveis locais `trip` e `bq` para verificar a lógica de disparo do disjuntor de amarre.

Caso o disjuntor esteja aberto e deseja-se fechá-lo, é necessário avaliar a sua lógica de fechamento.

Por fim, faz-se a chamada pelos métodos `Update()` dos objetos `disjuntor`, `secBarraA` e `secBarraB`.

Dessa forma, sempre que o método `Update()` da classe `Amarre` for chamado, ele, ao final, irá chamar os métodos `Update()` dos objetos disjuntor, `secBarraA` e `secBarraB`, atualizando os seus estados. Por exemplo, caso a proteção do amarre envie um sinal de trip para abertura do disjuntor de amarre, primeiramente o valor da propriedade comando do objeto disjuntor é alterado para `Abrir`.

Contudo, o disjuntor só mudará seu estado quando a sua função `Update()` for chamada.

Como será possível perceber, essa estratégia de atualização dos objetos é utilizada nas demais classes no decorrer da implementação do simulador, ou seja, sempre que uma classe é formada por uma agregação de objetos de outras classes, no seu método `Update()` é feita uma chamada ao método `Update()` de cada um dos objetos que a compõe. Assim, é possível garantir a atualização de todos os objetos instanciados no simulador a cada iteração.

3.2.4 Modelagem das LTs

A modelagem das LTs foi feita com a implementação da classe `Linha`, a qual é baseada na configuração ilustrada na Figura 3.12a, ou seja, um disjuntor de linha e as seguintes chaves seccionadoras: seletoras de barra, de chegada de linha, de saída para o barramento, de *bypass* e de terra. O disjuntor de linha é apenas uma instância da classe `Disjuntor`. Por outro lado, foi necessário implementar uma classe para cada tipo de seccionadora, a fim de representar a suas diferentes lógicas de intertravamento. Assim, foram implementadas as classes `SecLinhaBarraA`, `SecLinhaBarraB`, `SecLinhaBypass`, `SecLinhaTerra`, `SecLinhaChegada` e `SecLinhaSaida`. A diferença básica entre essas classes de seccionadoras é justamente a lógica de intertravamento implementada. A título de exemplo, apresenta-se a seguir a declaração da classe `SecLinhaBarraA`, conforme Figura 3.21, e o diagrama de estados que implementa a lógica de intertravamento da seccionadora seletora da barra A, conforme Figura 3.22 e a descrição do código no Apêndice H.

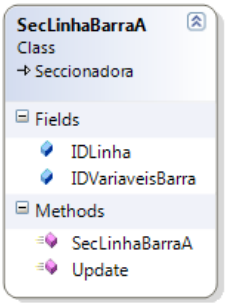

Figura 3. 21 - Classe `SecLinhaBarraA`.

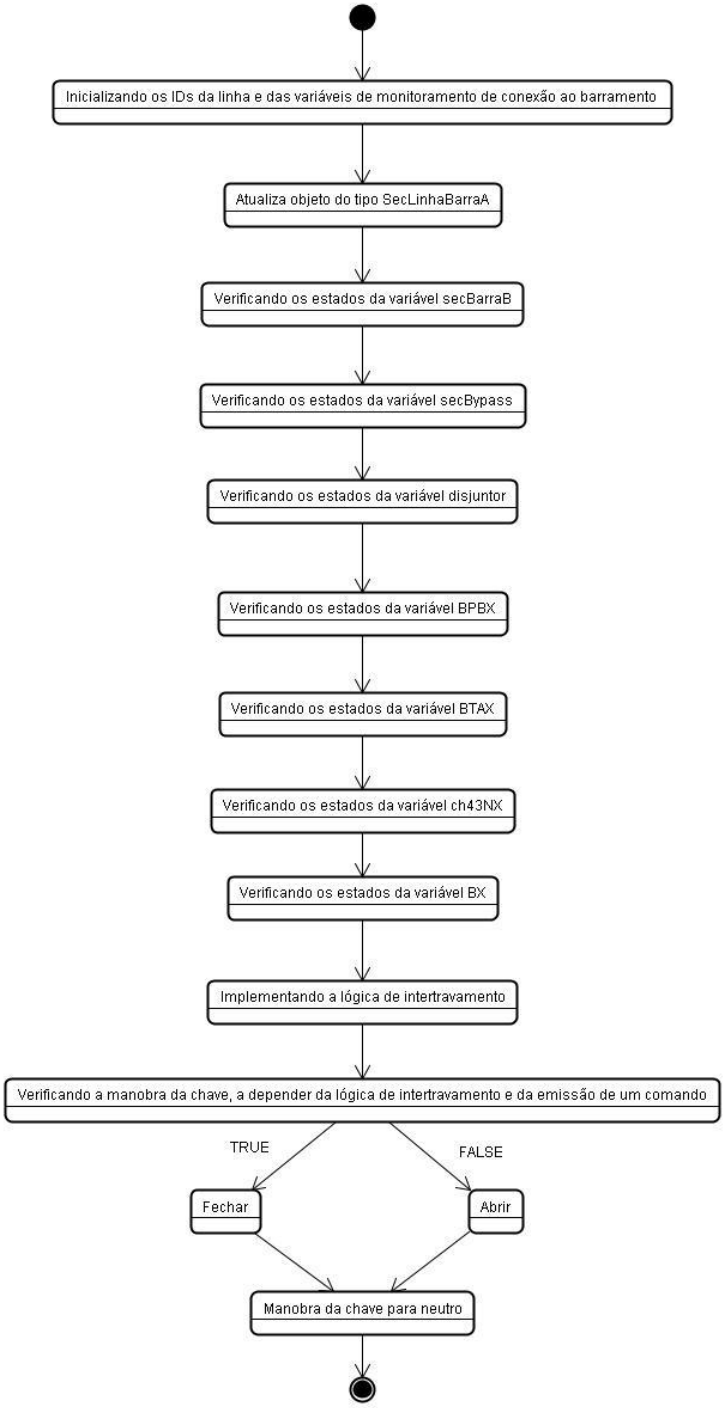

Figura 3. 22 - Diagrama de estados da classe `SecLinhaBarraA`.

Além das propriedades ID, estado e comando, herdadas da classe `Seccionadora`, a classe `SecLinhaBarraA` possui as seguintes propriedades:

1. `IDLinha` – Usada para armazenar ID da LT a qual a seccionadora está associada.
2. `IDVariaveisBarra` – Usada para armazenar o ID das variáveis de monitoramento de conexão à barra que são consultadas na lógica de intertravamento.
3. `IDProtAmarre` – Usada para armazenar o ID da proteção do amarre, necessária para verificar o estado de operação da proteção diferencial de barra, que pode assumir os seguintes valores: `OFF`, `INDIVIDUAL` ou `OVERALL`.
4. `IDLigacaoNormal` – Usada para armazenar o estado normal de conexão da seccionadora na barra A, que pode ser NA ou NF.

Todas essas propriedades são informador como argumento de entrada ao construtor da classe `SecLinhaBarraA`.

As propriedades descritas à cima são utilizadas para verificar o estado das variáveis locais ao método `Update()`, as quais são avaliadas na lógica de intertravamento desse tipo de seccionadora, a saber:

1. `secBarraB` – Estado da seccionadora seletora da barra da B: fechada (`true`) ou aberta (`false`).
2. `secBypass` – Estado da seccionadora de *bypass* do disjuntor de linha: fechada (`true`) ou aberta (`false`).
3. `disjuntor` – Estado do disjuntor de linhas: fechada (`true`) ou aberta (`false`).
4. `overall` – Estado de operação da proteção diferencial de barra: `OFF`, `INDIVIDUAL` ou `OVERALL`.
5. `BX` – Proteção transferida e barra B escolhida (`true`) ou proteção não transferida (`false`).
6. `BPBX` – Chaves de *bypass* de todas as LTs e ATs conectados ao barramento abertas: chaves abertas (`true`) ou alguma chave fechada (`false`).
7. `BTAX` – Paralelo fechado entre as barras A e B do barramento ao qual a LT está conectada: paralelo fechado (`true`) ou paralelo aberto (`false`).

Ao analisar a lógica de intertravamento desse tipo de chave seccionadora usada na SE, verificou-se que a lógica é independente do nível de tensão do barramento, mas depende de qual barra (A ou B) a chave é conectada na configuração normal de operação da SE.

A seccionadora só poderá ser manobrada, ou seja, mudar de estado, caso a variável saida esteja em `true`, um comando de abertura ou fechamento seja enviado e o disjuntor de amarre esteja aberto.

Os diagramas de estados das demais classes `SecLinhaBarraB`, `SecLinhaBypass`, `SecLinhaTerra`, `SecLinhaChegada` e `SecLinhaSaida` são semelhantes às apresentadas anteriormente.

A seguir, apresenta-se a declaração da classe `Linha` ,conforme Figura 3.23:

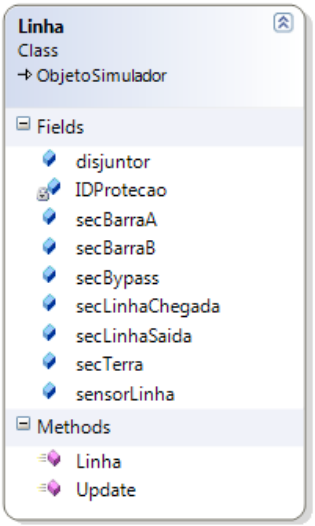

Figura 3. 23 - Classe `Linha`.

A figura 3.24 representa o Diagrama de Estados da classe `Linha` e a descrição do código no Apêndice I.

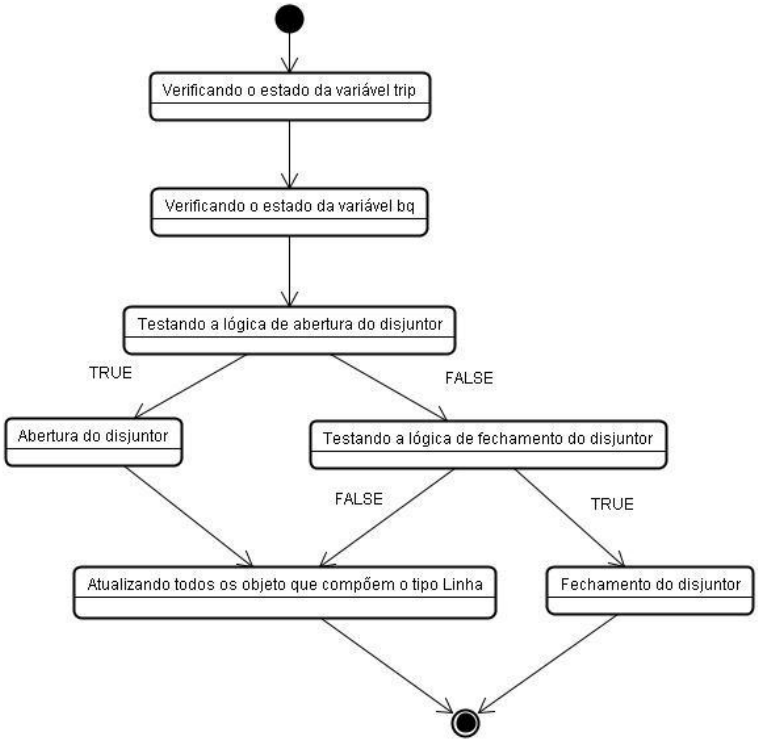

Figura 3. 24 - Diagrama de estados da classe `Linha`.

3.2.5 Modelagem dos ATs

A modelagem dos ATs foi feita com a implementação da classe `Trafo`, a qual é baseada na configuração ilustrada na Figura 3.12b, ou seja, um disjuntor de linha e as seguintes chaves seccionadoras: seletoras de barra, de chegada de linha, de saída para o barramento e de *bypass*. O disjuntor de linha é apenas uma instância da classe `Disjuntor`. Por outro lado, foi necessário implementar uma classe para cada tipo de seccionadora, a fim de representar a suas diferentes lógicas de intertravamento. Assim, foram implementadas as classes `SecTrafoBarraA`, `SecTrafoBarraB`, `SecTrafoBypass` e `SecTrafoDisjuntor`. A diferença básica entre essas classes de seccionadoras é justamente a lógica de intertravamento implementada. A título de exemplo, apresenta-se a seguir a declaração da classe `SecTrafoBarraA`, conforme Figura 3.25, e o diagrama de estados que implementa a lógica de intertravamento da seccionadora seletora da barra A, conforme Figura 3.26 e a descrição do código no Apêndice J.

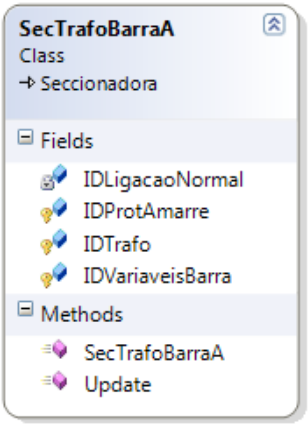

Figura 3. 25 - Classe `SecTrafoBarraA`.

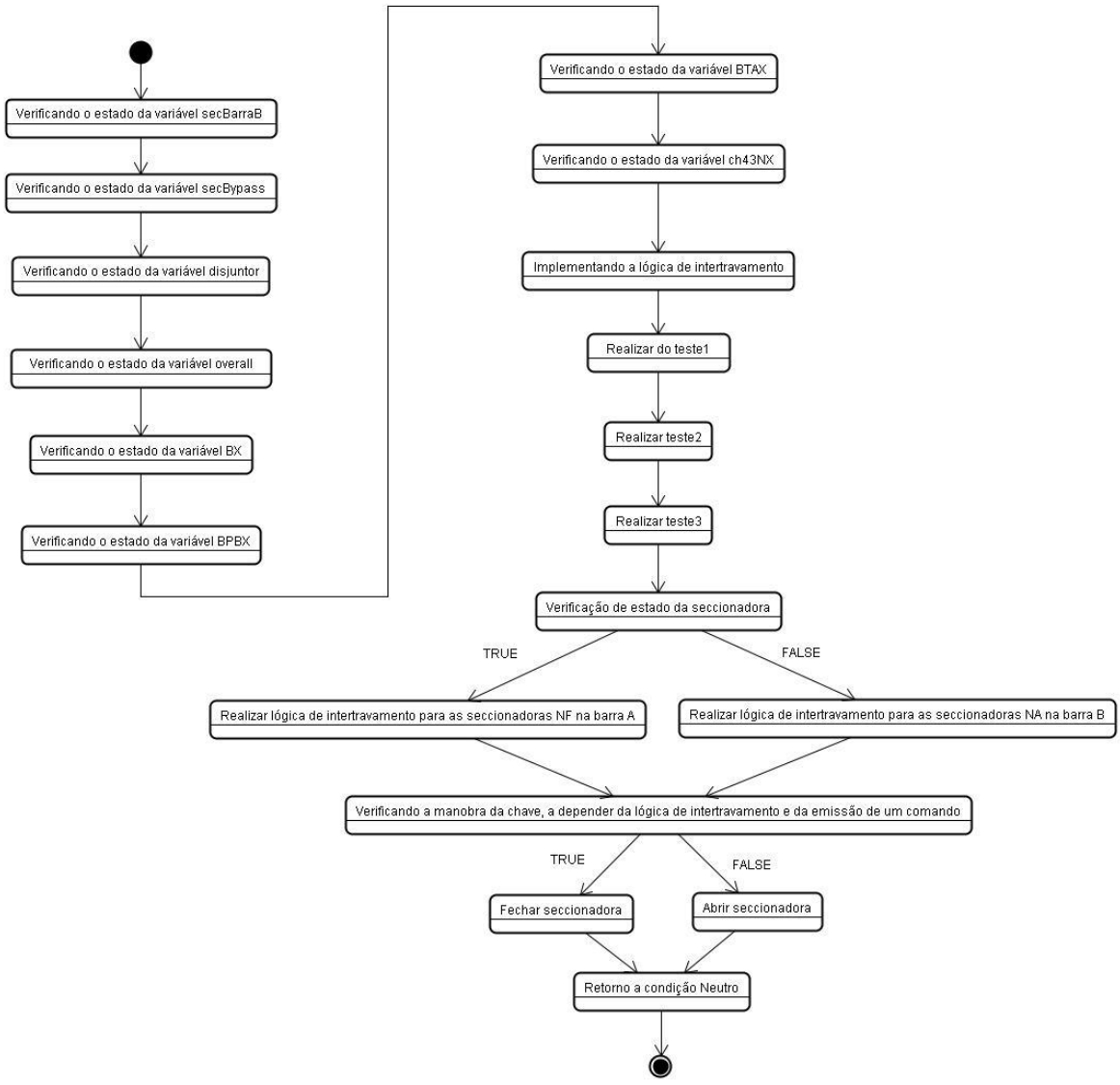

Figura 3. 26 - Diagrama de estados da classe `SecTrafoBarraA`.

Além das propriedades ID, estado e comando, herdadas da classe `Seccionadora`, a classe `SecTrafoBarraA` possui as seguintes propriedades:

1. `IDTrafo` – Usada para armazenar ID do Trafo a qual a seccionadora está associada.
2. `IDVariaveisBarra` – Usada para armazenar o ID das variáveis de monitoramento de conexão à barra que são consultadas na lógica de intertravamento.
3. `IDProtAmarre` – Usada para armazenar o ID da proteção do amarre, necessária para verificar o estado de operação da proteção diferencial de barra, que pode assumir os seguintes valores: `OFF`, `INDIVIDUAL` ou `OVERALL`.
4. `IDLigacaoNormal` – Usada para armazenar o estado normal de conexão da seccionadora na barra A, que pode ser NA ou NF.

Todas essas propriedades são informador como argumento de entrada ao construtor da classe `SecTrafoBarraA`.

As propriedades são utilizadas para verificar o estado das variáveis locais ao método `Update()`, as quais são avaliadas na lógica de intertravamento desse tipo de seccionadora, a saber:

1. `secBarraB` – Estado da seccionadora seletora da barra da B: fechada (`true`) ou aberta (`false`).
2. `secBypass` – Estado da seccionadora de *bypass* do disjuntor de linha: fechada (`true`) ou aberta (`false`).
3. `disjuntor` – Estado do disjuntor de linhas: fechada (`true`) ou aberta (`false`).
4. `overall` – Estado de operação da proteção diferencial de barra: `OFF`, `INDIVIDUAL` ou `OVERALL`.
5. `BX` – Proteção transferida e barra B escolhida (`true`) ou proteção não transferida (`false`).
6. `BPBX` – Chaves de *bypass* de todas as LTs e ATs conectados ao barramento abertas: chaves abertas (`true`) ou alguma chave fechada (`false`).
7. `BTAX` – Paralelo fechado entre as barras A e B do barramento ao qual a LT está conectada: paralelo fechado (`true`) ou paralelo aberto (`false`).

Ao analisar a lógica de intertravamento desse tipo de chave seccionadora usada na SE, verificou-se que a lógica é independente do nível de tensão do barramento, mas depende de qual barra (A ou B) a chave é conectada na configuração normal de operação da SE.

A seccionadora só poderá ser manobrada, ou seja, mudar de estado, caso a variável saida esteja em `true`, um comando de abertura ou fechamento seja enviado e o disjuntor de amarre esteja aberto:

Os diagramas de estados das demais classes `SecTrafoBarraB`, `SecTrafoBypass`, e `SecTrafoDisjuntor` são semelhantes às apresentadas anteriormente.

A seguir, apresenta-se a declaração da classe `Trafo`, conforme Figura 3.27:

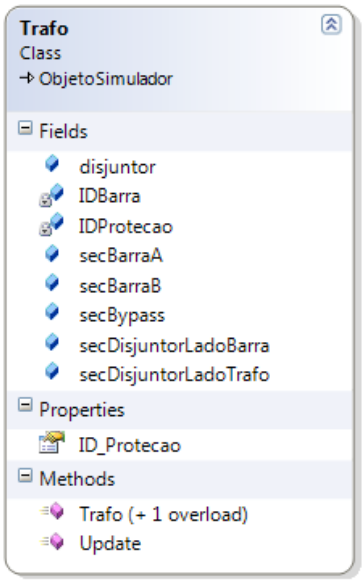

Figura 3. 27 - Classe `Trafo`.

A figura 3.28 representa o Diagrama de Estados da classe `Trafo` e a descrição do código no Apêndice K.

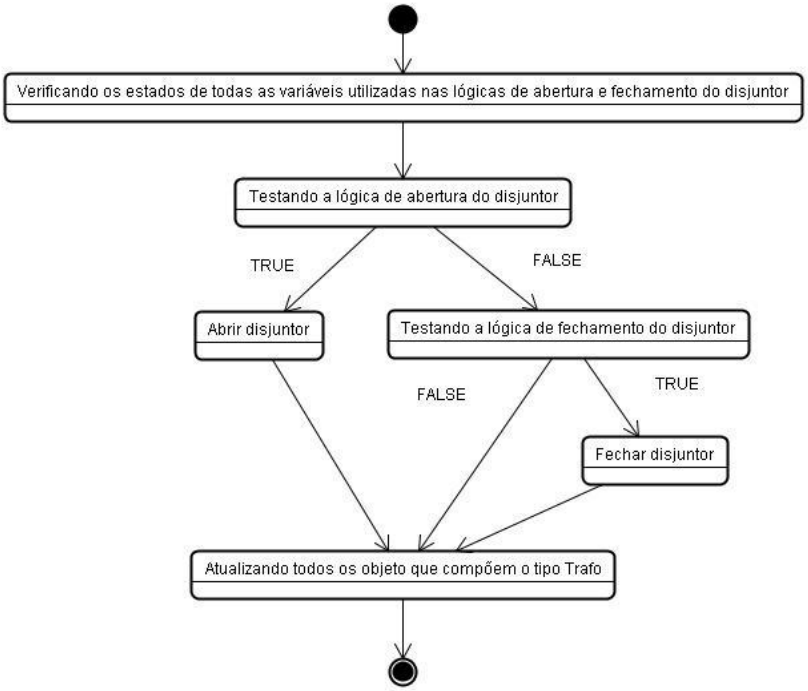

Figura 3. 28 - Diagrama de estados da classe `Trafo`.

3.3 MODELAGEM DAS PROTEÇÕES

Com a modelagem dos componentes de uma SE, faz-se necessário a modelagem da proteção do seu sistema elétrico utilizando o paradigma da orientação a objetos.

Nesse sentido, foram implementadas as classes `ProtLinha`, `ProtAmarre` , `ProtTrafo` e `ProtReator`. A seguir, apresenta-se a descrição de cada uma dessas classes.

3.3.1 Modelagem da Proteção de LTs

A proteção de LTs foi modelado a partir de uma classe denominada `ProtLinha`, que implementa os atributos e métodos comuns a todas as proteções de LTs. Esta classe possui as seguintes propriedades:

1. `IDLinha` – Usada para definir um identificador único para a proteção de LTs. De fato, essa variável é herdada da classe base `ObjetoSimulador`, de modo que o valor atribuído a ela é passado para o construtor da classe `ObjetoSimulador` diretamente no construtor da classe `ProtLinha`.
2. `rele50BF` e `rele62BF` – Usada para declarar o estado dos relés de falha de disjuntor: fechado (`true`) ou aberto (`false`).

3. `rele79` – Usada para declarar o estado dos relés de religamento de linha: fechado (`true`) ou aberto (`false`).
4. `ch43TD` – Usada para declarar o estado da chave de transferência de proteção: não-transferido (`true`) ou transferido (`false`).
5. `trip` – Usada para definir o estado do trip da proteção: acionado (`true`) ou não-acionado (`false`).
6. `bq` – Usada para definir o estado do bloqueio da proteção: acionado (`true`) ou não-acionado (`false`).
7. `bfa` - Usada para definir o estado do bloqueio para falha de disjuntor na barra A: acionado (`true`) ou não-acionado (`false`).
8. `bfb` - Usada para definir o estado do bloqueio para falha de disjuntor na barra B: acionado (`true`) ou não-acionado (`false`).
9. `tbfa` - Usada para definir o estado trip do bloqueio para falha de disjuntor na barra A: acionado (`true`) ou não-acionado (`false`).
10. `tbfb` - Usada para definir o estado trip do bloqueio para falha de disjuntor na barra B: acionado (`true`) ou não-acionado (`false`).

A seguir, apresenta-se a declaração da classe `ProtLinha`, conforme Figura 3.29:

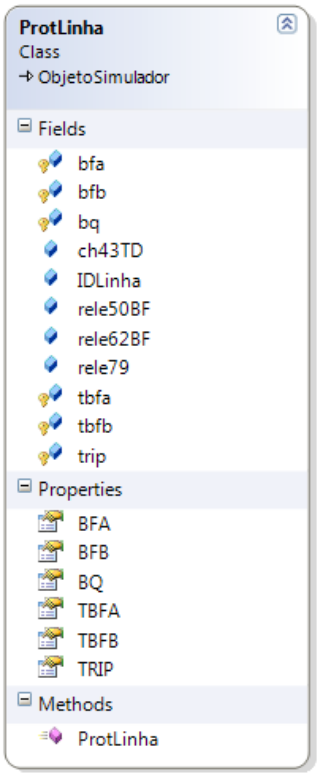

Figura 3. 29 - Classe `ProtLinha`.

A figura 3.30 representa o Diagrama de Estados da classe `ProtLinha` e a descrição do código no Apêndice L.

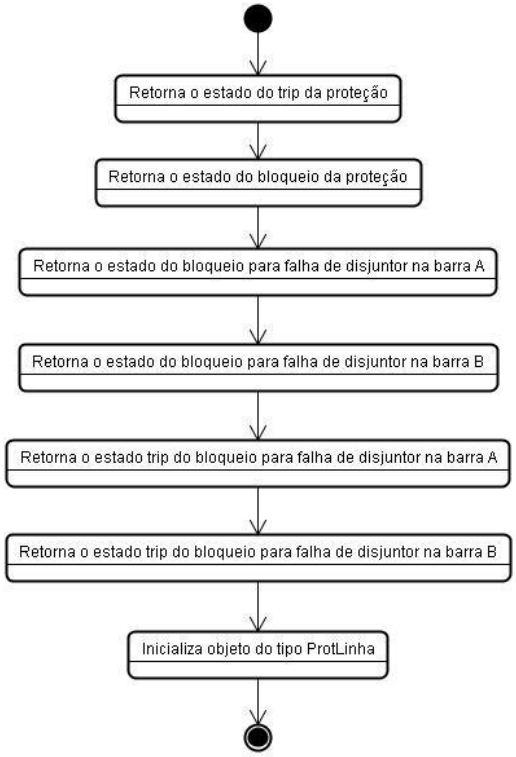

Figura 3. 30 - Diagrama de estados da classe `Protlinha`.

Por outro lado, foi necessário implementar uma classe para cada nível de tensão e barramento, a fim de representar a suas diferentes lógicas de intertravamento. Assim, foram implementadas as classes `ProtLT1_345kV`, `ProtLT2_345kV`, `ProtLT1_230kV`, `ProtLT2_230kV`, `ProtLT1_138kV` e `ProtLT2_138kV`. A diferença básica entre essas classes de proteção de amarre é justamente a lógica de proteção implementada. A título de exemplo, apresenta-se a seguir a declaração da classe `ProtLT1_345kV`, conforme Figura 3.31, e o diagrama de estados que implementam as lógicas de proteções, conforme Figura 3.32 e a descrição do código no Apêndice M.

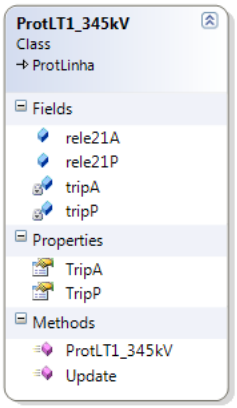

Figura 3. 31 - Classe `ProtLT1_345kV`.

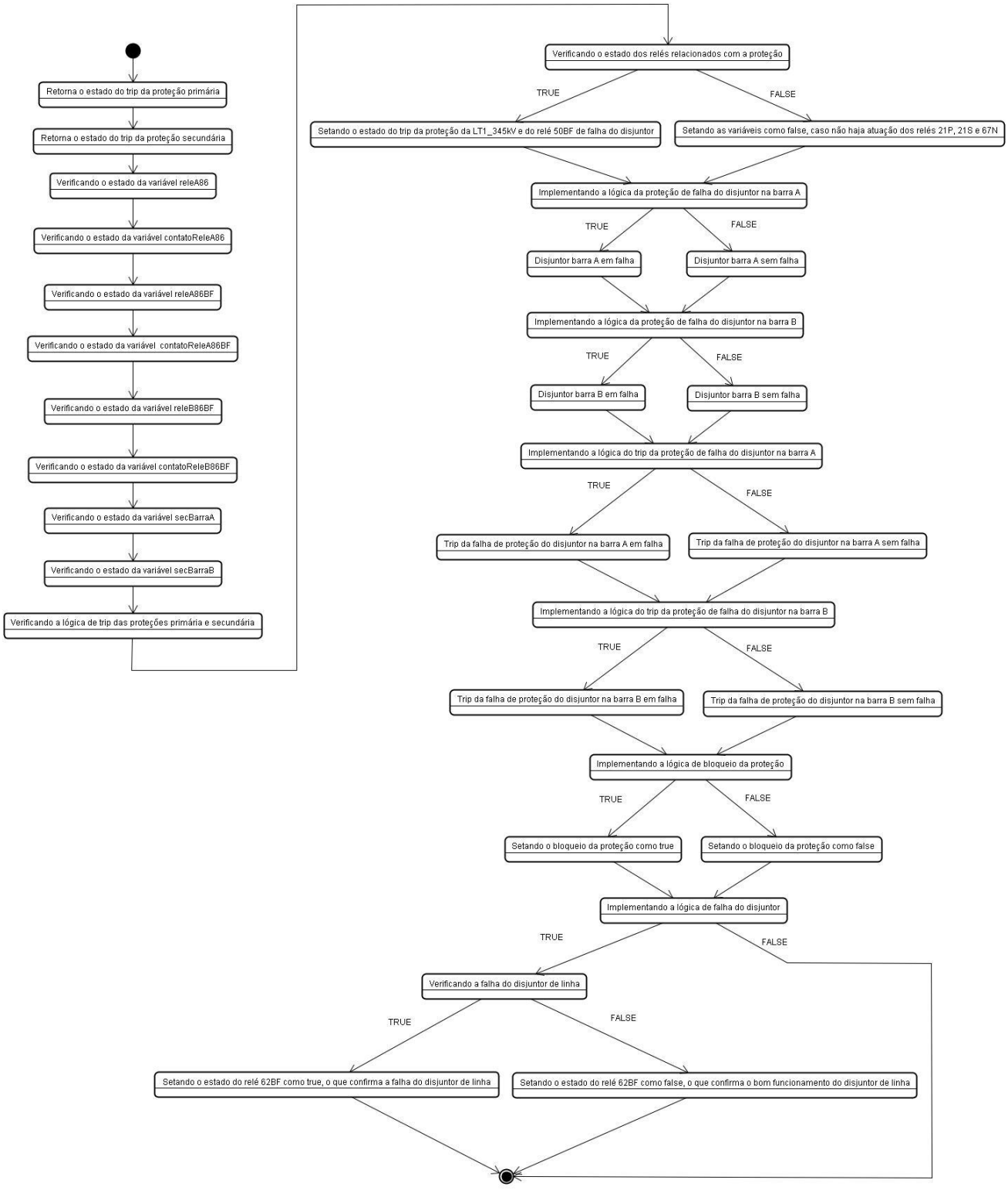

Figura 3. 32 - Diagrama de estados da `ProtLT1_345kV`.

3.3.2 Modelagem da Proteção de Amarre

A proteção de Amarre foi modelado a partir de uma classe denominada `ProtAmarre`, que implementa os atributos e métodos comuns a todas as proteções de Amarre. Esta classe possui as seguintes propriedades:

1. `releA86BF` e `rele86BF` – Usada para definir os Relés de bloqueio de falha do disjuntor: acionado (`true`) ou não-acionado (`false`).
2. `releA86` e `releB86` – Usada para declarar o estado dos Relés de bloqueio da proteção diferencial de barra: acionado (`true`) ou não-acionado (`false`).
3. `releA87A_A` e `releA87B_A` e `releA87C_A` – Usada para declarar o estado dos Relés diferenciais da proteção da barra A: acionado (`true`) ou não-acionado (`false`).
4. `releA87A_B` e `releA87B_B` e `releA87C_B` – Usada para declarar o estado dos Relés diferenciais da proteção da barra B: acionado (`true`) ou não-acionado (`false`).
5. `ch29` – Usada para definir o estado da chave de seleção do modo de operação da proteção diferencial de barra, que pode assumir os valores: `Off, Individual, Overall`.
6. `ch43IB` – Usada para definir o estado da Chave de seleção da barra de supervisão quando da transferência da proteção para o disjuntor de amarre, que podem assumir os valores: `A,B e off`.
7. `trip` – Usada para definir o estado do trip da proteção disjuntor de amarre: acionado (`true`) ou não-acionado (`false`).
8. `bq` – Usada para definir o estado do bloqueio da proteção do disjuntor de amarre: acionado (`true`) ou não-acionado (`false`).
9. `bfa` - Usada para definir o estado do bloqueio para falha de disjuntor na barra A: acionado (`true`) ou não-acionado (`false`).
10. `bfb` - Usada para definir o estado do bloqueio para falha de disjuntor na barra B: acionado (`true`) ou não-acionado (`false`).

A seguir, apresenta-se a declaração da classe `ProtAmarre`, conforme Figura 3.33:

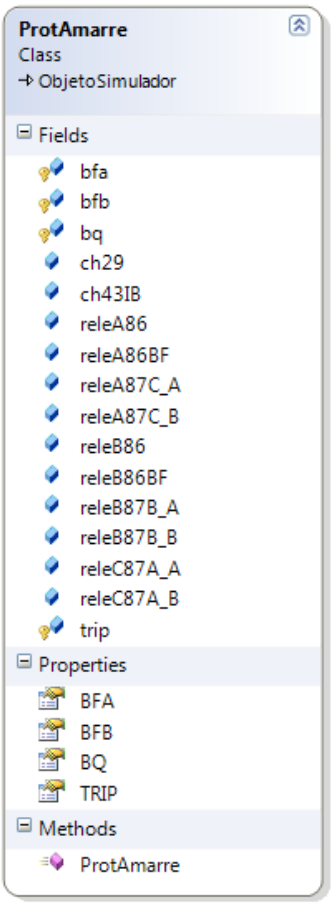

Figura 3. 33 - Classe `ProtAmarre`.

A figura 3.34 representa o Diagrama de Estados da classe `ProtAmarre` e a descrição do código no Apêndice N.

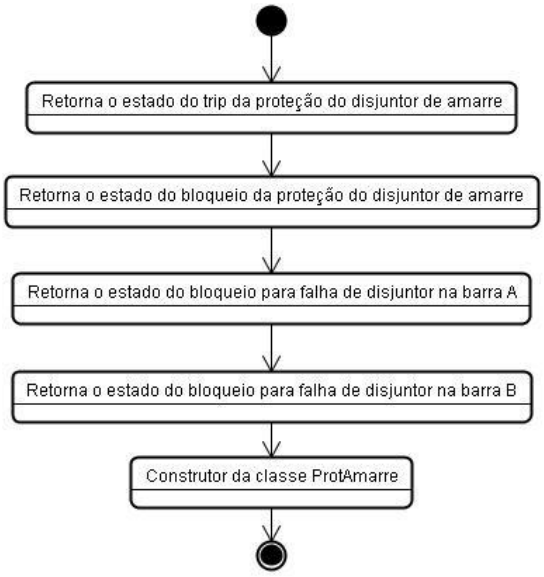

Figura 3. 34 - Diagrama de estados da classe `ProtAmarre`.

Por outro lado, foi necessário implementar uma classe para cada nível de tensão, a fim de representar a suas diferentes lógicas de intertravamento. Assim, foram implementadas as classes `ProtAmarre345kV`, `ProtAmarre230kV` e `ProtAmarre138kV`. A diferença básica entre essas classes de proteção de amarre é justamente a lógica de proteção implementada. A título de exemplo, apresenta-se a seguir a declaração da classe `ProtAmarre345kV`, conforme Figura 3.35, e o diagrama de estados que implementam as lógicas de proteções, conforme Figura 3.36 e a descrição do código no Apêndice O.

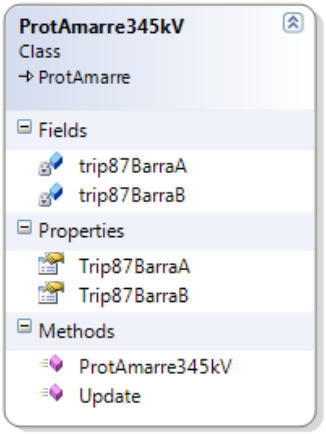

Figura 3. 35 - Classe `ProtAmarre345kV`.

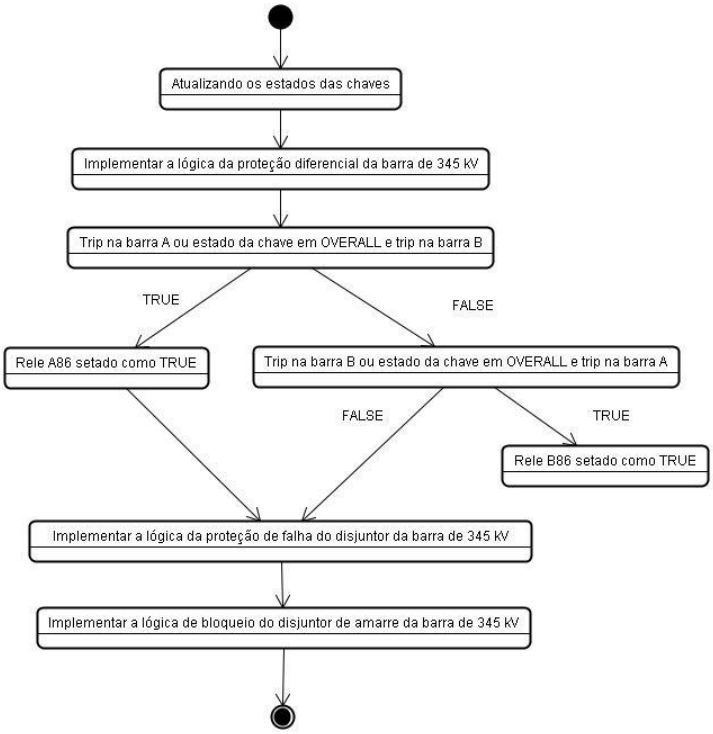

Figura 3. 36 - Diagrama de estados da classe `ProtAmarre345kV`.

3.3.3 Modelagem da Proteção do Trafo

A proteção de LTs foi modelado a partir de uma classe denominada `ProtTrafo`, que implementa os atributos e métodos comuns a todas as proteções de Trafos. Esta classe possui as seguintes propriedades:

1. `IDTrafo` – Usada para definir um identificador único para a proteção de Trafos. De fato, essa variável é herdada da classe base `ObjetoSimulador`, de modo que o valor atribuído a ela é passado para o construtor da classe `ObjetoSimulador` diretamente no construtor da classe `ProtTrafo`.
2. `rele50BF` e `rele62BF` – Usada para declarar o estado dos relés de falha de disjuntor: fechado (`true`) ou aberto (`false`).
3. `ch43TD` – Usada para declarar o estado da chave de transferência de proteção: não-transferido (`true`) ou transferido (`false`).
4. `trip` – Usada para definir o estado do trip da proteção: acionado (`true`) ou não-acionado (`false`).
5. `bq` – Usada para definir o estado do bloqueio da proteção: acionado (`true`) ou não-acionado (`false`).
6. `bfa` - Usada para definir o estado do bloqueio para falha de disjuntor na barra A: acionado (`true`) ou não-acionado (`false`).
7. `bfb` - Usada para definir o estado do bloqueio para falha de disjuntor na barra B: acionado (`true`) ou não-acionado (`false`).
8. `tbfa` - Usada para definir o estado trip do bloqueio para falha de disjuntor na barra A: acionado (`true`) ou não-acionado (`false`).
9. `tbfb` - Usada para definir o estado trip do bloqueio para falha de disjuntor na barra B: acionado (`true`) ou não-acionado (`false`).

A seguir, apresenta-se a declaração da classe `ProtTrafo`, conforme Figura 3.37:

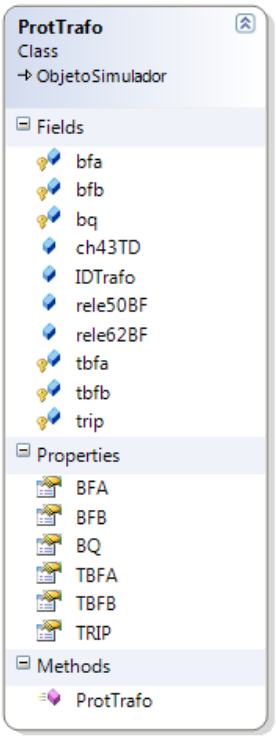

Figura 3. 37 - Classe `ProtTrafo`.

A figura 3.31 representa o Diagrama de Estados da classe `ProtTrafo` e a descrição do código no Apêndice P.

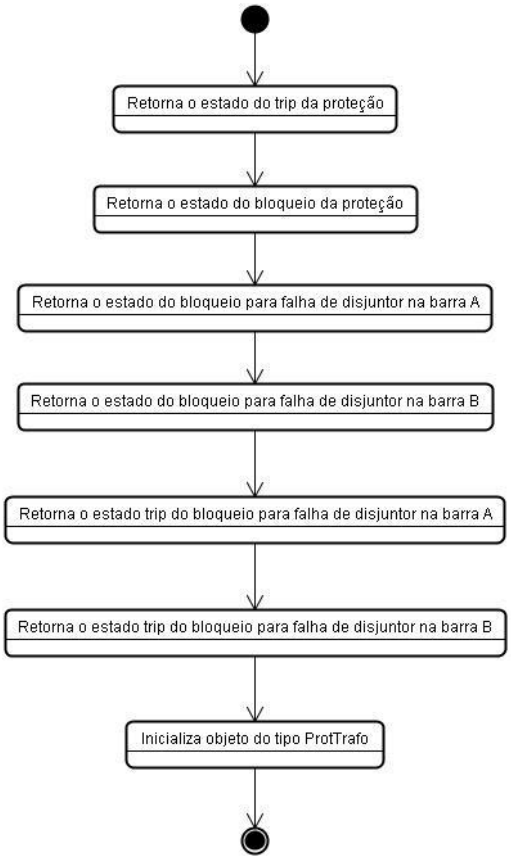

Figura 3. 38 - Diagrama de estados da classe `ProtTrafo`.

Por outro lado, foi necessário implementar uma classe para o lado de baixa-tensão e outra para alta-tensão do trafo, pois no lado de alta-tensão há mais relés de proteção ativos. Assim, foram implementadas as classes `ProtTrafoBT`, conforme Figura 3.39, e `ProtTrafoAT`, conforme Figura 3.40.

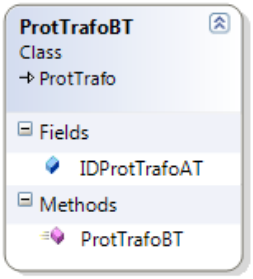

Figura 3. 39 - Classe `ProtTrafoBT`.

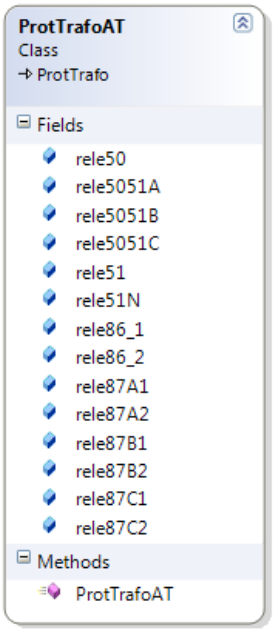

Figura 3. 40 - Classe `ProtTrafoAT`.

3.3.4 Modelagem da Proteção do Reator

A proteção do Reator foi modelada a partir de uma classe denominada `ProtReator`, conforme Figura 3.41, que implementa os atributos e métodos comuns a todas as proteções de Amarre. Esta classe possui as seguintes propriedades:

1. `trip` – Usada para definir o estado do trip da proteção disjuntor de amarre: acionado (`true`) ou não-acionado (`false`).
2. `bq` – Usada para definir o estado do bloqueio da proteção do disjuntor de amarre: acionado (`true`) ou não-acionado (`false`).
3. `bfa` - Usada para definir o estado do bloqueio para falha de disjuntor na barra A: acionado (`true`) ou não-acionado (`false`).
4. `bfb` - Usada para definir o estado do bloqueio para falha de disjuntor na barra B: acionado (`true`) ou não-acionado (`false`).
5. `tbfa` - Usada para definir o estado trip do bloqueio para falha de disjuntor na barra A: acionado (`true`) ou não-acionado (`false`).
6. `tbfb` - Usada para definir o estado trip do bloqueio para falha de disjuntor na barra B: acionado (`true`) ou não-acionado (`false`).

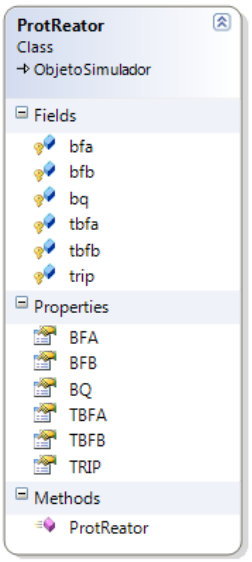

Figura 3. 41 - Classe `ProtReator`.

A figura 3.42 representa o Diagrama de Estados da classe `ProtReator` e a descrição do código no Apêndice Q.

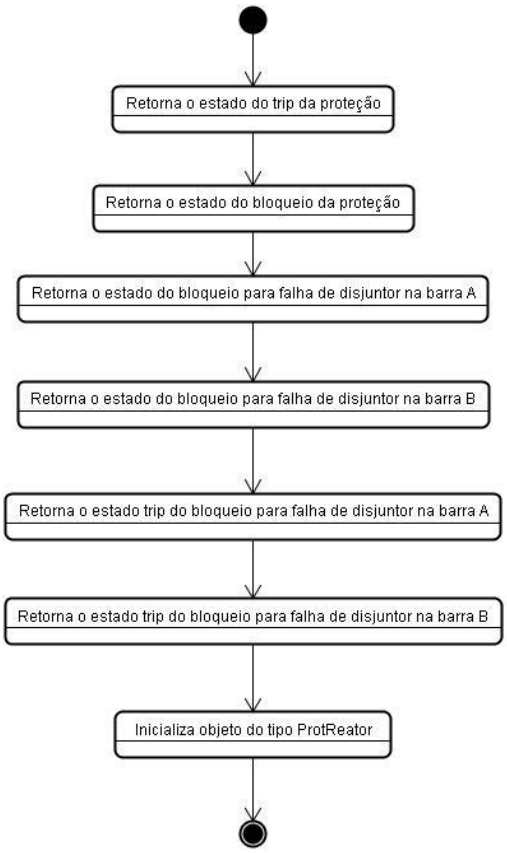

Figura 3. 42 - Diagrama de estados da classe `ProtReator`.

4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Neste capítulo, apresentam-se os resultados obtidos nos testes de intertravamento e manobras realizadas no sistema desenvolvido. O sistema foi desenvolvido em linguagem de programação C#, com a utilização do aplicativo Microsoft Visual Studio 2010©. A implementação dos testes estão localizados na classe `Evento` e a chamada do evento está localizada na classe `Main`. Para a análise dos resultados foram realizados 05 (cinco) testes de intertravamento, e 28 (vinte e oito) manobras, conforme descrição da classe `Main`, Figura 3.10, sendo que 2 testes de intertravamento e 6 manobras serão descritos nas seções que seguem.

4.1 TESTES DE INTERTRAVAMENTO

Os testes de intertravamento visam verificar se os códigos implementados no simulador funcionam devidamente, conforme o funcionamento de uma subestação real, onde o intertravamento assegura a funcionalidade e segurança dos técnicos e equipamentos no caso de uma operação incorreta de abertura ou fechamento de uma chave seccionadora.

4.1.1 Fechamento da Seccionadora Bypass dos Disjuntores das LTs

Neste teste de intertravamento será efetuada a tentativa de fechamento das seccionadoras de *bypass* das LTs, SC8319, SC8329, SC7419, SC7429, SC6819 e SC6829 em diversas etapas, até a realização do fechamento da seccionadora de *bypass* da LT1_345kV.

A seguir estão os passos realizados para a execução do teste:

a) Primeiramente, é realizado uma tentativa de fechamento das seccionadoras de *bypass*, não ocorrendo o fechamento, conforme passos 01 a 06 da Tabela 4.1

   Como as chaves seccionadoras de chegada e saída da linha de transmissão se encontram Fechadas (com seu estado em `TRUE`) as condições do teste 1 de intertravamento não são satisfeitas, conforme fragmento do código e representado pela Figura 4.1.

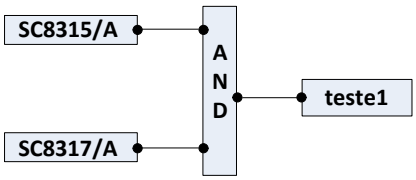

Figura 4. 1 - Teste 1 do Intertravamento da Seccionadora Bypass da LT1_345kV.

b) Abertura dos disjuntores das LTs, ATs e de Amarre, conforme passos 07 a 19 da Tabela 4.1;
c) Nova tentativa de fechamento das seccionadoras de *bypass*, não ocorrendo o fechamento, conforme passos 20 a 25 da Tabela 4.1;

   Apesar da abertura dos disjuntores, não satisfaz o teste2 das condições de intertravamento, conforme código a seguir e representado pela Figura 4.2;

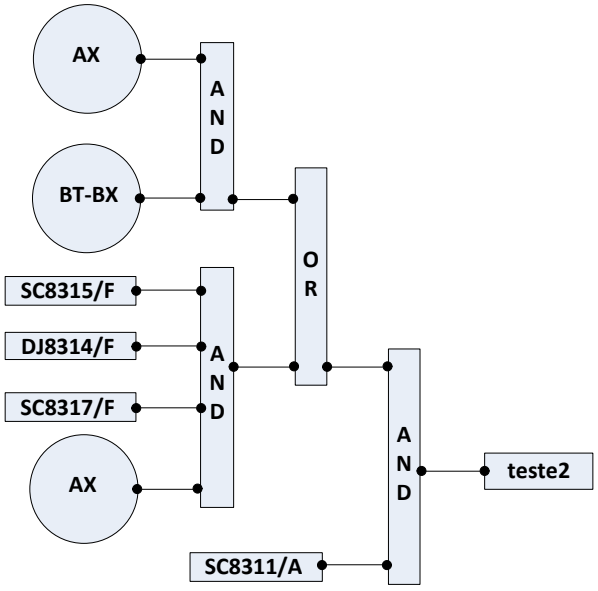

Figura 4. 2 - Teste 2 do Intertravamento da Seccionadora Bypass.

d) Abertura das seccionadoras de amarre, conforme passos 26 a 31 da Tabela 4.1;
e) Nova tentativa de fechamento das seccionadoras de *bypass*, não ocorrendo o fechamento, conforme passos 32 a 37 da Tabela 4.1.

   Apesar da abertura das seccionadoras do amarre, não satisfaz o teste2 e teste3 das condições de intertravamento, conforme códigos e representado pela Figura 4.2 e Figura 4.3;

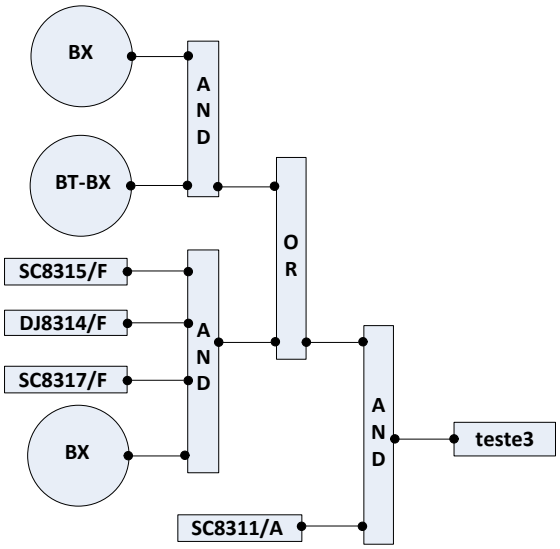

Figura 4. 3 - Teste 3 do Intertravamento da Seccionadora Bypass.

f) Efetuando a transferência da proteção da LT1_345kV para o disjuntor de amarre, conforme os seguintes passos:

- Colocar a chave 29 da proteção de barra na posição OVERALL;
- Abrir as seccionadoras SC8315 e SC8317;
- Colocar a chave 43TD na posição TRANSFERIDA;
- Setar o estado da chave 43IB na barra A;

Conforme realizado nos passos 38 a 42 da Tabela 4.1.

g) Nova tentativa de fechamento das seccionadoras de *bypass*, ocorrendo o fechamento da seccionadora de *bypass* SC8319 da LT1_345kV, conforme passos 43 a 48 da Tabela 4.1.

Com a transferência da proteção realizada na LT1_345kV, torna-se possível o fechamento da seccionadora do *bypass*, satisfazendo todas as condições impostas pelo intertravamento, conforme código a seguir e Figura 4.4.

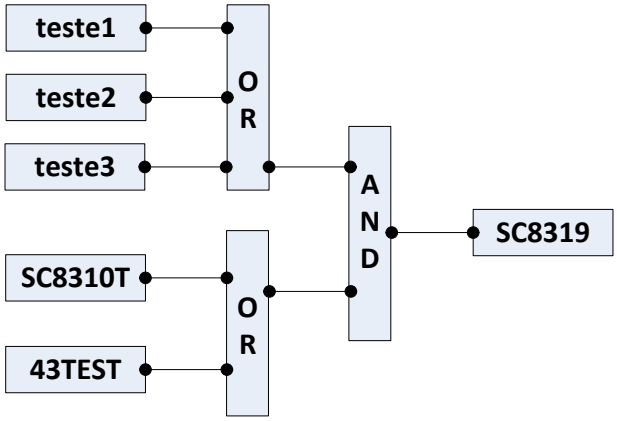

Figura 4. 4 - Condição Completa de Intertravamento da Seccionadora Bypass.

Fica demonstrado neste teste, que o intertravamento da seccionadora impede seu fechamento de maneira "casual", devendo o operador do sistema efetuar uma série de procedimentos para que o fechamento ocorra, no caso é necessário a transferência da proteção da LT para o disjuntor de amarre.

Tabela 4. 1 - Manobra: Teste de Intertravamento – Fechar *Bypass* dos Disjuntores das LTs.

	Sec *Bypass* LT1_345kV:	Sec *Bypass* LT2-345kV:	Sec *Bypass* LT1-230kV:	Sec *Bypass* LT2-230kV:	Sec *Bypass* LT1-138kV:	Sec *Bypass* LT2-138kV:
00 - Condição Inicial	False	False	False	False	False	False
01 - Fechar Sec SC8319	False	False	False	False	False	False
02 - Fechar Sec SC8329	False	False	False	False	False	False
03 - Fechar Sec SC7419	False	False	False	False	False	False
04 - Fechar Sec SC7429	False	False	False	False	False	False
05 - Fechar Sec SC6819	False	False	False	False	False	False
06 - Fechar Sec SC6829	False	False	False	False	False	False
07 - Abrir DJ8314	False	False	False	False	False	False
08 - Abrir DJ8324	False	False	False	False	False	False
09 - Abrir DJ7414	False	False	False	False	False	False
10 - Abrir DJ7424	False	False	False	False	False	False
11 - Abrir DJ6814	False	False	False	False	False	False
12 - Abrir DJ6824	False	False	False	False	False	False
13 - Abrir DJ804 Amarre	False	False	False	False	False	False
14 - Abrir DJ704 Amarre	False	False	False	False	False	False
15 - Abrir DJ604 Amarre	False	False	False	False	False	False
16 - Abrir DJ814 AT01	False	False	False	False	False	False
17 - Abrir D614 AT01	False	False	False	False	False	False

Continuação da Tabela 4.1

	Sec *Bypass* LT1-345kV:	Sec *Bypass* LT2-345kV:	Sec *Bypass* LT1-230kV:	Sec *Bypass* LT2-230kV:	Sec *Bypass* LT1-138kV:	Sec *Bypass* LT2-138kV:
18 - Abrir DJ824 AT02	False	False	False	False	False	False
19 - Abrir DJ724 AT02	False	False	False	False	False	False
20 - Fechar Sec SC8319	False	False	False	False	False	False
21 - Fechar Sec SC8329	False	False	False	False	False	False
22 - Fechar Sec SC7419	False	False	False	False	False	False
23 - Fechar Sec SC7429	False	False	False	False	False	False
24 - Fechar Sec SC6819	False	False	False	False	False	False
25 - Fechar Sec SC6829	False	False	False	False	False	False
26 - Abrir Sec SC8311	False	False	False	False	False	False
27 - Abrir Sec SC8323	False	False	False	False	False	False
28 - Abrir Sec SC7411	False	False	False	False	False	False
29 - Abrir Sec SC7423	False	False	False	False	False	False
30 - Abrir Sec SC6811	False	False	False	False	False	False
31 - Abrir Sec SC6823	False	False	False	False	False	False
32 - Fechar Sec SC8319	False	False	False	False	False	False
33 - Fechar Sec SC8329	False	False	False	False	False	False
34 - Fechar Sec SC7419	False	False	False	False	False	False
35 - Fechar Sec SC7429	False	False	False	False	False	False
36 - Fechar Sec SC6819	False	False	False	False	False	False
37 - Fechar Sec SC6829	False	False	False	False	False	False
38 - Ch29 em OVERALL 345kV	False	False	False	False	False	False
39 - Abrir Sec SC8315	False	False	False	False	False	False
40 - Abrir Sec SC8317	False	False	False	False	False	False
41 - Ch 43TD do DJ8314 LT1_345kV Transferida	False	False	False	False	False	False
42 - Estado Ch 43IB 345kV em A	False	False	False	False	False	False
43 - Fechar Sec SC8319	True	False	False	False	False	False
44 - Fechar Sec SC8329	True	False	False	False	False	False
45 - Fechar Sec SC7419	True	False	False	False	False	False
46 - Fechar Sec SC7429	True	False	False	False	False	False
47 - Fechar Sec SC6819	True	False	False	False	False	False
48 - Fechar Sec SC6829	True	False	False	False	False	False

4.1.2 Abertura de Chaves Seccionadoras dos Disjuntores das LTs

Neste teste de intertravamento será efetuada a tentativa de abertura das chaves seccionadoras dos disjuntores de todas as linhas de transmissão.

A seguir estão os passos realizados para a execução do teste:

a) Tentativa de abertura das seccionadoras de conexão do disjuntor de todas as LTs. Como os disjuntores das LT's estão fechados, ou seja, com a passagem de corrente elétrica, o intertravamento impede a abertura das chaves, conforme demonstrados nos passos 01 a 06 das Tabelas 4.2, 4.3, 4.4, 4.5, 4.6 e 4.7.

   Como os disjuntores estão fechados, não satisfaz a lógica de intertravamento das seccionadoras de chegada e de saída de linha, conforme Figura 4.5.

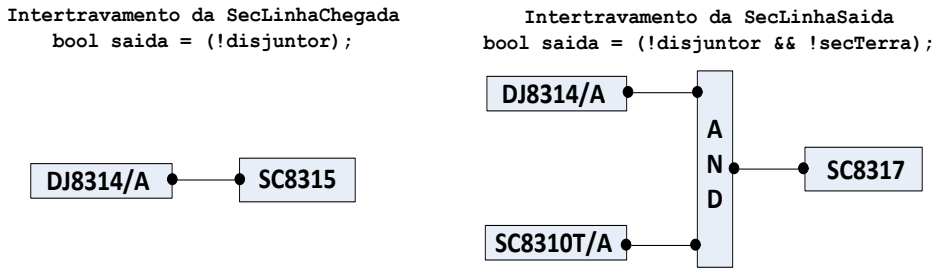

Figura 4. 5 - Lógica de Intertravamento das Seccionadoras de Chegada e Saída de Linha.

b) Abertura dos disjuntores de todas as LTs, conforme demonstrados nos passos 07 a 12 das Tabelas 4.2, 4.3, 4.4, 4.5, 4.6 e 4.7;

c) Nova tentativa de abertura das chaves seccionadoras, uma vez que os disjuntores se encontram em aberto, satisfazendo a lógica de intertravamento, ocorre a abertura das chaves, conforme demonstrados nos passos 13 a 18 das Tabelas 4.2, 4.3, 4.4, 4.5, 4.6 e 4.7;

Tabela 4. 2 - TESTE DE INTERTRAVAMENTO - Abrir chaves dos disjuntores da LT1_345kV.

	Disjuntor LT1-345kV:	Sec Linha Chegada LT1-345kV:	Sec Linha Saida LT1-345kV:
00 - Condição Inicial	True	True	True
01 - Abrir Sec SC8315 e SC8317	True	True	True
02 - Abrir Sec SC8325 e SC8327	True	True	True
03 - Abrir Sec SC7415 e SC7417	True	True	True
04 - Abrir Sec SC7425 e SC7427	True	True	True
05 - Abrir Sec SC6815 e SC6817	True	True	True
06 - Abrir Sec SC6825 e SC6827	True	True	True
07 - Abrir DJ8314	False	True	True
08 - Abrir DJ8324	False	True	True
09 - Abrir DJ7414	False	True	True
10 - Abrir DJ7424	False	True	True
11 - Abrir DJ6814	False	True	True
12 - Abrir DJ6824	False	True	True
13 - Abrir Sec SC8315 e SC8317	False	False	False
14 - Abrir Sec SC8325 e SC8327	False	False	False
15 - Abrir Sec SC7415 e SC7417	False	False	False
16 - Abrir Sec SC7425 e SC7427	False	False	False
17 - Abrir Sec SC6815 e SC6817	False	False	False
18 - Abrir Sec SC6825 e SC6827	False	False	False

Tabela 4. 3 - TESTE DE INTERTRAVAMENTO - Abrir chaves dos disjuntores da LT2_345kV.

	Disjuntor LT2-345kV:	Sec Linha Chegada LT2-345kV:	Sec Linha Saida LT2-345kV:
00 - Condição Inicial	True	True	True
01 - Abrir Sec SC8315 e SC8317	True	True	True
02 - Abrir Sec SC8325 e SC8327	True	True	True
03 - Abrir Sec SC7415 e SC7417	True	True	True
04 - Abrir Sec SC7425 e SC7427	True	True	True
05 - Abrir Sec SC6815 e SC6817	True	True	True

Continuação da Tabela 4.3

	Disjuntor LT2-345kV:	Sec Linha Chegada LT2-345kV:	Sec Linha Saida LT2-345kV:
06 - Abrir Sec SC6825 e SC6827	True	True	True
07 - Abrir DJ8314	True	True	True
08 - Abrir DJ8324	False	True	True
09 - Abrir DJ7414	False	True	True
10 - Abrir DJ7424	False	True	True
11 - Abrir DJ6814	False	True	True
12 - Abrir DJ6824	False	True	True
13 - Abrir Sec SC8315 e SC8317	False	True	True
14 - Abrir Sec SC8325 e SC8327	False	False	False
15 - Abrir Sec SC7415 e SC7417	False	False	False
16 - Abrir Sec SC7425 e SC7427	False	False	False
17 - Abrir Sec SC6815 e SC6817	False	False	False
18 - Abrir Sec SC6825 e SC6827	False	False	False

Tabela 4. 4 - TESTE DE INTERTRAVAMENTO - Abrir chaves dos disjuntores da LT1_230kV.

	Disjuntor LT1-230kV:	Sec Linha Chegada LT1-230kV:	Sec Linha Saida LT1-230kV:
00 - Condição Inicial	True	True	True
01 - Abrir Sec SC8315 e SC8317	True	True	True
02 - Abrir Sec SC8325 e SC8327	True	True	True
03 - Abrir Sec SC7415 e SC7417	True	True	True
04 - Abrir Sec SC7425 e SC7427	True	True	True
05 - Abrir Sec SC6815 e SC6817	True	True	True
06 - Abrir Sec SC6825 e SC6827	True	True	True
07 - Abrir DJ8314	True	True	True
08 - Abrir DJ8324	True	True	True
09 - Abrir DJ7414	False	True	True
10 - Abrir DJ7424	False	True	True
11 - Abrir DJ6814	False	True	True
12 - Abrir DJ6824	False	True	True
13 - Abrir Sec SC8315 e SC8317	False	True	True
14 - Abrir Sec SC8325 e SC8327	False	True	True

Continuação da Tabela 4.4

	Disjuntor LT1-230kV:	Sec Linha Chegada LT1-230kV:	Sec Linha Saida LT1-230kV:
15 - Abrir Sec SC7415 e SC7417	False	False	False
16 - Abrir Sec SC7425 e SC7427	False	False	False
17 - Abrir Sec SC6815 e SC6817	False	False	False
18 - Abrir Sec SC6825 e SC6827	False	False	False

Tabela 4. 5 - TESTE DE INTERTRAVAMENTO - Abrir chaves dos disjuntores da LT2_230kV.

	Disjuntor LT2-230kV:	Sec Linha Chegada LT2-230kV:	Sec Linha Saida LT2-230kV:
00 - Condição Inicial	True	True	True
01 - Abrir Sec SC8315 e SC8317	True	True	True
02 - Abrir Sec SC8325 e SC8327	True	True	True
03 - Abrir Sec SC7415 e SC7417	True	True	True
04 - Abrir Sec SC7425 e SC7427	True	True	True
05 - Abrir Sec SC6815 e SC6817	True	True	True
06 - Abrir Sec SC6825 e SC6827	True	True	True
07 - Abrir DJ8314	True	True	True
08 - Abrir DJ8324	True	True	True
09 - Abrir DJ7414	True	True	True
10 - Abrir DJ7424	False	True	True
11 - Abrir DJ6814	False	True	True
12 - Abrir DJ6824	False	True	True
13 - Abrir Sec SC8315 e SC8317	False	True	True
14 - Abrir Sec SC8325 e SC8327	False	True	True
15 - Abrir Sec SC7415 e SC7417	False	True	True
16 - Abrir Sec SC7425 e SC7427	False	False	False
17 - Abrir Sec SC6815 e SC6817	False	False	False
18 - Abrir Sec SC6825 e SC6827	False	False	False

Tabela 4. 6 - TESTE DE INTERTRAVAMENTO - Abrir chaves dos disjuntores da LT1_138kV.

	Disjuntor LT1-138kV:	Sec Linha Chegada LT1-138kV:	Sec Linha Saida LT1-138kV:
00 - Condição Inicial	True	True	True
01 - Abrir Sec SC8315 e SC8317	True	True	True
02 - Abrir Sec SC8325 e SC8327	True	True	True
03 - Abrir Sec SC7415 e SC7417	True	True	True
04 - Abrir Sec SC7425 e SC7427	True	True	True
05 - Abrir Sec SC6815 e SC6817	True	True	True
06 - Abrir Sec SC6825 e SC6827	True	True	True
07 - Abrir DJ8314	True	True	True
08 - Abrir DJ8324	True	True	True
09 - Abrir DJ7414	True	True	True
10 - Abrir DJ7424	True	True	True
11 - Abrir DJ6814	False	True	True
12 - Abrir DJ6824	False	True	True
13 - Abrir Sec SC8315 e SC8317	False	True	True
14 - Abrir Sec SC8325 e SC8327	False	True	True
15 - Abrir Sec SC7415 e SC7417	False	True	True
16 - Abrir Sec SC7425 e SC7427	False	True	True
17 - Abrir Sec SC6815 e SC6817	False	False	False
18 - Abrir Sec SC6825 e SC6827	False	False	False

Tabela 4. 7 - TESTE DE INTERTRAVAMENTO - Abrir chaves dos disjuntores da LT2_138kV.

	Disjuntor LT2-138kV:	Sec Linha Chegada LT2-138kV:	Sec Linha Saida LT2-138kV:
00 - Condição Inicial	True	True	True
01 - Abrir Sec SC8315 e SC8317	True	True	True
02 - Abrir Sec SC8325 e SC8327	True	True	True
03 - Abrir Sec SC7415 e SC7417	True	True	True
04 - Abrir Sec SC7425 e SC7427	True	True	True
05 - Abrir Sec SC6815 e SC6817	True	True	True

Continuação da Tabela 4.7

	Disjuntor LT2-138kV:	Sec Linha Chegada LT2-138kV:	Sec Linha Saida LT2-138kV:
06 - Abrir Sec SC6825 e SC6827	True	True	True
07 - Abrir DJ8314	True	True	True
08 - Abrir DJ8324	True	True	True
09 - Abrir DJ7414	True	True	True
10 - Abrir DJ7424	True	True	True
11 - Abrir DJ6814	True	True	True
12 - Abrir DJ6824	False	True	True
13 - Abrir Sec SC8315 e SC8317	False	True	True
14 - Abrir Sec SC8325 e SC8327	False	True	True
15 - Abrir Sec SC7415 e SC7417	False	True	True
16 - Abrir Sec SC7425 e SC7427	False	True	True
17 - Abrir Sec SC6815 e SC6817	False	True	True
18 - Abrir Sec SC6825 e SC6827	False	False	False

4.2 MANOBRAS

As manobras representam a simulação de uma operação em uma subestação onde ocorre um estado de emergência, com a atuação de um relé, ocorrendo a abertura dos disjuntores necessários para sanar a falha, e após a restauração do sistema para sua condição normal.

4.2.1 Proteção NORMAL + Atuação do Relé 21P da LT1_345kV

Nesta manobra, o sistema encontra-se funcionando com sua configuração normal com todas as proteções de linha na sua posição NORMAL, conforme Figura 4.6. Neste exemplo tem-se um curto-circuito na Linha de Transmissão LT1_345kV, ocorrendo a abertura do disjuntor DJ8314.

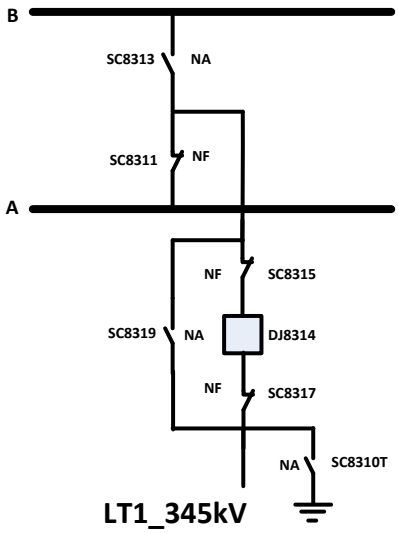

Figura 4. 6 - Sistema em configuração NORMAL.

Com o curto-circuito, tem-se a mudança de estado do relé 21P de FALSE para TRUE, ocorrendo a abertura do disjuntor DJ8314.

Para a restauração do sistema é necessário restaurar o estado do relé 21P para FALSE, efetuar o Sincronismo do disjuntor DJ8314 e a seguir o seu fechamento, ocorrendo as mudanças de estado conforme Tabela 4.8.

Tabela 4. 8 - Manobra: Proteção NORMAL + Atuação do relé 21P da LT1_345kV.

	Disjuntor DJ8314	Sincronismo DJ8314	Rele21P
0 - Condição Inicial	True	False	False
1 - Estado da proteção TRUE	False	False	True
2 - Estado da proteção FALSE	False	False	False
3 - Sincronismo em TRUE	False	True	False
4 - Fechando o disjuntor	True	False	False

4.2.2 Proteção da LT1_345kV TRANSFERIDA + Atuação do Relé 21P

O sistema encontra-se funcionando com sua configuração normal, conforme Figura 4.7. Devido a uma manutenção do disjuntor DJ8314 da LT1_345kV, será necessário a transferência da proteção da LT para o disjuntor de Amarre DJ804.

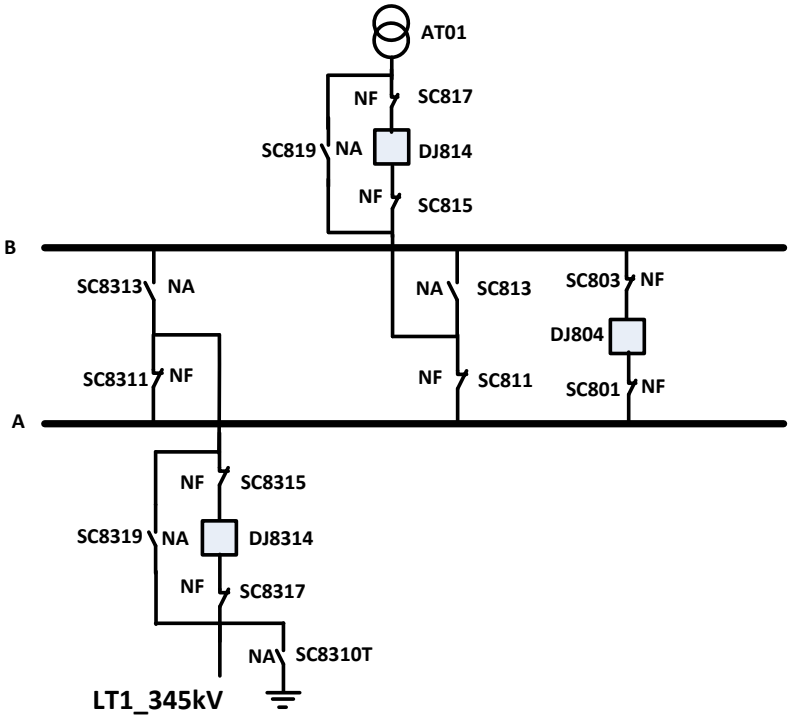

Figura 4. 7 - Sistema em configuração NORMAL.

Para que isso ocorra é necessário seguir uma série de ações, conforme segue:

a) Colocar a chave 29 da proteção de barra de 345kV na posição OVERALL;
b) Fechamento da seccionadora SC 813, conforme Figura 4.8;
c) Abertura da seccionadora SC811, conforme Figura 4.8;

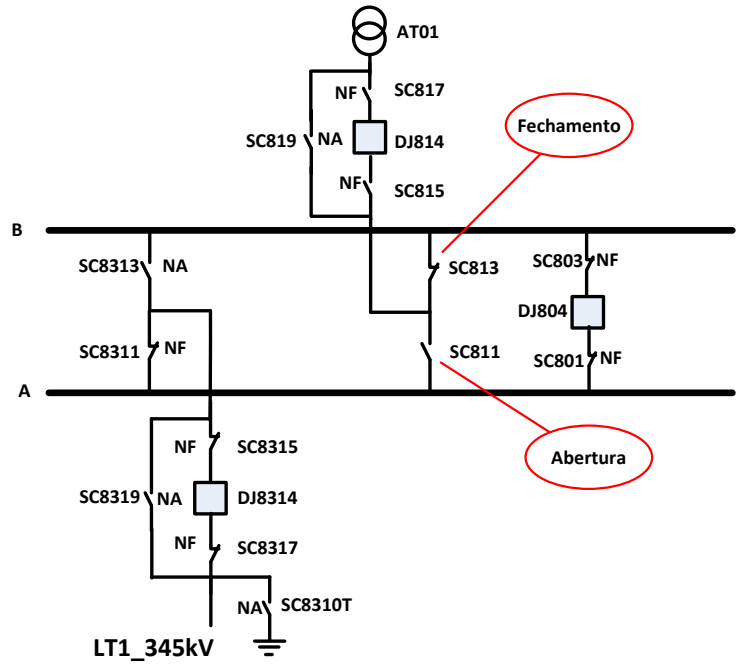

Figura 4. 8 - Fechamento da SC813 e Aberturada SC811.

d) Abertura do disjuntor DJ8314 da LT1_345kV;

e) Abertura das chaves seccionadoras SC8315 e SC8317 de conexão do disjuntor, conforme Figura 4.9;

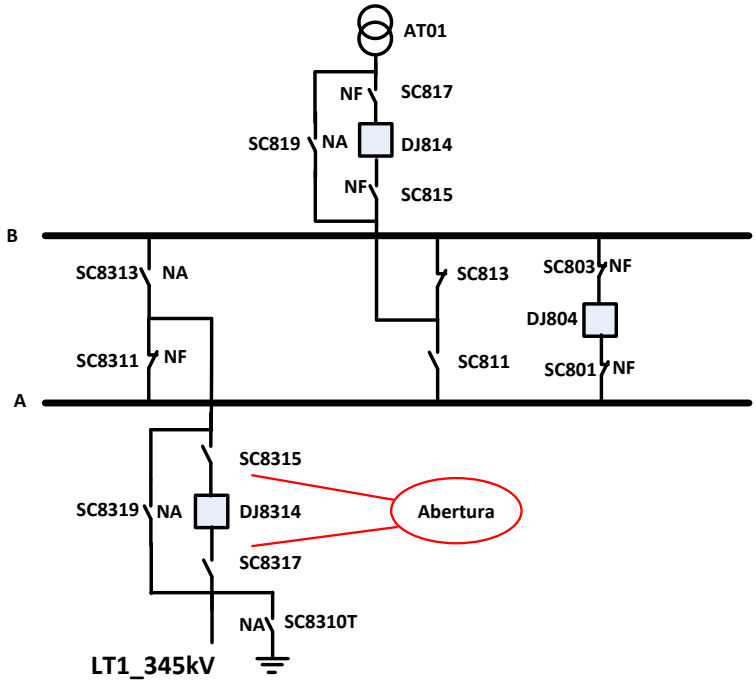

Figura 4. 9 - Abertura das SC8315 e SC8317.

f) Abertura do disjuntor de amarre DJ804;

g) Abertura das seccionadoras do amarre SC801 e SC803, conforme Figura 4.10;

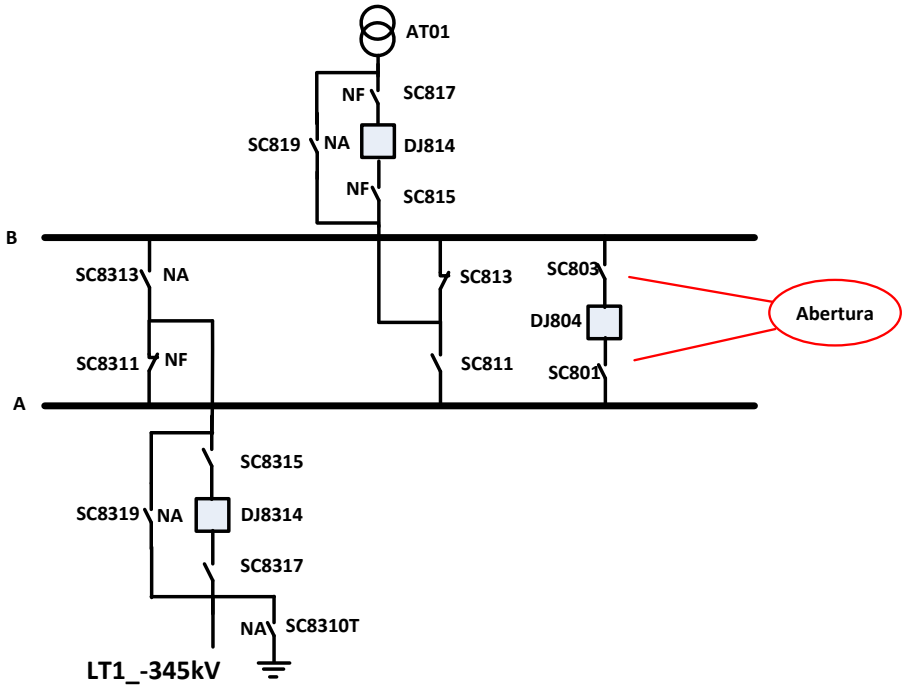

Figura 4. 10 - Abertura das SC801 e SC803.

h) Colocar a chave 43TD do DJ8314 na posição TRANSFERIDA;

i) Setar o Estado da Chave 43IB na Barra A;

j) Fechamento da chave de bypass SC8319 do DJ8314, conforme Figura 4.11;

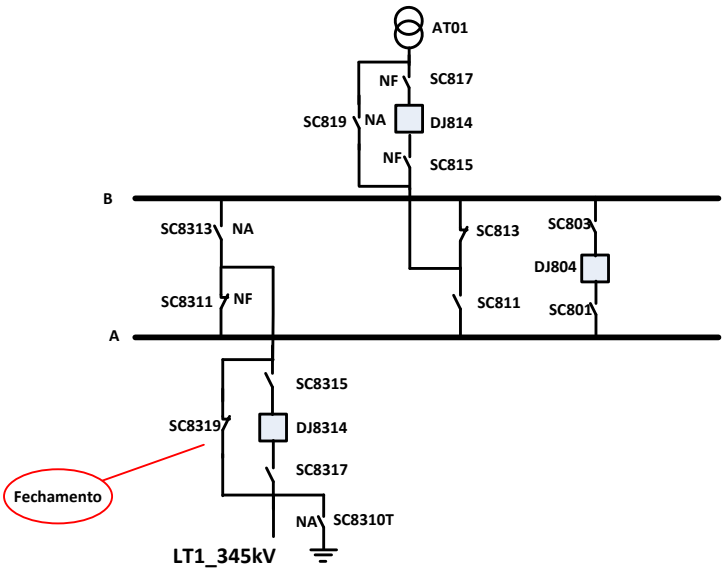

Figura 4. 11 - Fechamento da SC8319.

k) Fechamento das seccionadoras SC803 e SC801 do DJ804, conforme Figura 4.12;

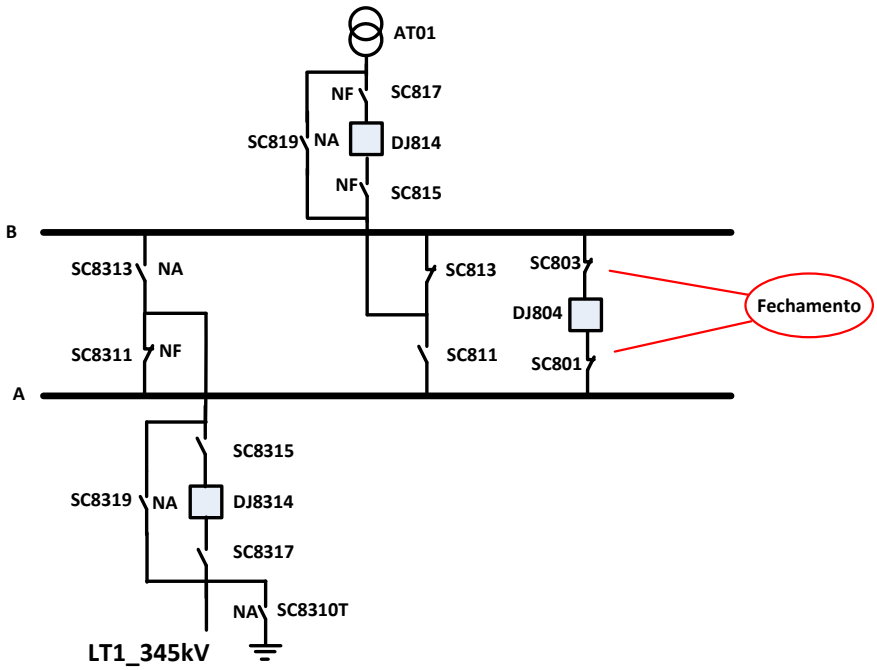

Figura 4. 12 - Fechamento das SC803 e SC801.

l) Setando o sincronismo do disjuntor de amarre DJ804;

m) Fechamento do disjuntor de amarre DJ804;

Desta forma, a LT1_345kV tem a sua proteção transferida para o disjuntor do amarre DJ804. Durante o período de manutenção, nota-se que somente a LT1_345kV está conectado a Barra A, e os demais componentes estão conectados a Barra B.

n) Atuação do Relé 21P na LT1_345kV;

Na atuação do relé 21P na LT1_345kV ocasionará a abertura do disjuntor do amarre, DJ804, conforme demonstrado no passo 13 da Tabela 4.9.

Tabela 4. 9 - Manobra: Proteção da LT1_345kV Transferida + Atuação do relé 21P.

	Disjuntor LT DJ8314	Disjuntor Amarre DJ804	Prot LT Rele21P
00 - Condição Inicial	True	True	False
01 - Ch29 em OVERALL	True	True	False
02 - Fechando Sec Trafo SC813	True	True	False
03 - Abrir Sec SC811 e DJ8314	False	True	False
04 - Abrir Sec SC8315 e SC8317	False	True	False
05 - Abrir DJ804 Amarre	False	False	False
06 - Abrir Sec801 e SC803 Amarre	False	False	False
07 - Ch 43TD do DJ8314 Transferida	False	False	False
08 - Estado Ch 43IB em A	False	False	False
09 - Fechando Bypass SC8319	False	False	False
10 - Fechando Sec SC803 e SC801	False	False	False
11 - Sincronismo do DJ804	False	False	False
12 - Fechando DJ804	False	True	False
13 - Setando 21P em TRUE	False	False	True
14 - Setando 21P em False	False	False	False
15 - Sincronismo DJ804	False	False	False
16 - Fechando DJ804	False	True	False

o) Setando a proteção 21P da LT1_345kV em False;

p) Setando o sincronismo do DJ804;

q) Fechando o disjuntor DJ804.

Com a restauração da proteção para “false”, é possível fechar novamente o disjuntor, demonstrando dessa forma a proteção transferida do disjuntor da LT para o disjuntor de amarre.

4.2.3 Proteção NORMAL + atuação da proteção diferencial de barra da LT1_345kV

O sistema encontra-se funcionando com sua configuração normal com todas as proteções de linha na sua posição NORMAL e a proteção diferencial de barra na posição INDIVIDUAL. Acontece um curto-circuito na barra A de 345 kV. Como a chave 29 está na posição INDIVIDUAL, apenas os disjuntores conectados a barra A serão abertos. Neste caso, o disjuntor de amarre DJ 804, o disjuntor DJ814 do AT01 e o disjuntor de linha DJ8314 da LT1_345kV irão abrir. As seguintes ações são realizadas para a proteção e normalização do sistema:

a) Setando o estado do relé diferencial B87B da barra A, atuando a proteção, conforme passo 01 da Tabela 4.10, observa-se a abertura dos disjuntores que estão conectados a barra A, conforme Figura 4.13.

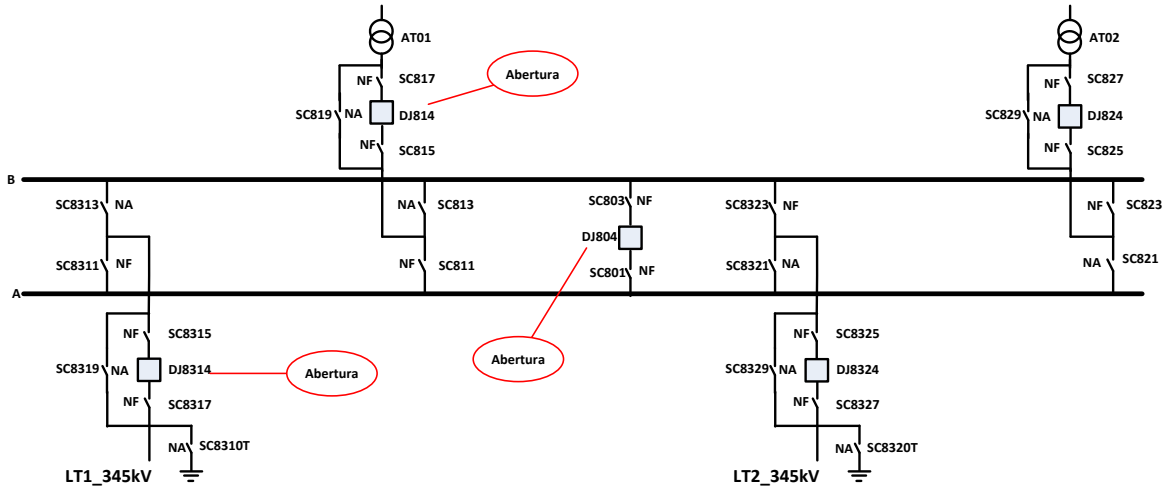

Figura 4. 13 - Abertura dos disjuntores conectados a Barra A.

b) Resetando o estado do rele diferencial B87B da barra A;
c) Resentando o estado do relé de bloqueio A86;
d) Setando o sincronismo do Disjuntor DJ814 do AT01;
e) Fechando o disjuntor DJ814 do AT01;
f) Setando o sincronismo do disjuntor de amarre DJ804;
g) Fechando o disjuntor de amarre DJ804;
h) Setando o sincronismo do disjuntor DJ 8314 da LT1_345kV;

i) Fechando o disjuntor DJ8314 da LT1_345kV.

Conforme a Tabela 4.10, verifica-se a atuação da proteção, no passo 01, e todo o processo de restauração do sistema, passos 02 a 09.

Tabela 4. 10 - Proteção NORMAL + atuação da proteção diferencial de barra da LT1_345kV.

	Disjuntor LT1_345kV (DJ8314)	Disjuntor LT2_345kV (DJ8324)	Disjuntor Amarre (DJ804)	Disjuntor AT01 (DJ814)	Disjuntor AT02 (DJ824)
00 - Condição Inicial	True	True	True	True	True
01 - Setando estado rele B87B na barra A	False	True	False	False	True
02 - Resetando estado rele B87B na barra A	False	True	False	False	True
03 - Resetando estado rele A86	False	True	False	False	True
04 - Sincronismo do DJ DJ814 do AT01	False	True	False	False	True
05 - Fechando o DJ DJ814 do AT01	False	True	False	True	True
06 - Sincronismo do DJ amarre	False	True	False	True	True
07 - Fechando o DJ amarre	False	True	True	True	True
08 - Sincronismo do DJ LT	False	True	True	True	True
09 - Fechando o DJ LT	True	True	True	True	True

4.2.4 Atuação do relé 21P da LT1_345kV + Falha do disjuntor DJ8314 da LT1_345kV

O sistema encontra-se funcionando com sua configuração normal com todas as proteções de linha na sua posição NORMAL e a proteção diferencial de barra na posição INDIVIDUAL. O relé 21P irá atuar para uma falta AB. A princípio, o disjuntor DJ8314 da LT1_345kV deveria abrir, mas ele apresentou falha. Assim, a proteção de falha do disjuntor deve abrir todos os disjuntores que estejam conectados a barra A, uma vez que o disjuntor DJ8314 está conectado a barra A. As seguintes ações são realizadas para a proteção e normalização do sistema:

a) Setando o estado de falha do disjuntor DJ8314 da LT1_345kV;
b) Setando o estado do relé 21P como TRUE, ocasionado falta AB, ocorrendo a abertura dos disjuntores de amarre e do AT01, conforme passo 02 da Tabela 4.11 e Figura 4.14;

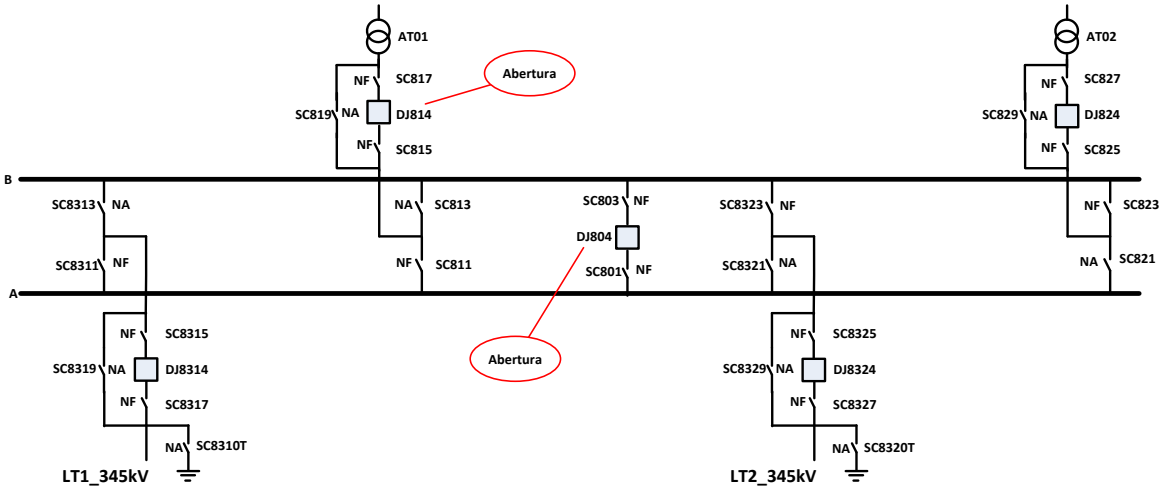

Figura 4. 14 - Abertura dos disjuntores de amarre e do AT01.

c) Resetando os reles 21P da LT1_345kV e setando o estado de falha do disjuntor DJ8314 da LT1_345kV;
d) Resetando os relés 50BF, 62 BF e A86BF da LT1_345kV;
e) Setando o sincronismo do disjuntor de amarre DJ804 e a seguir o seu fechamento, conforme passos 07 e 08 da Tebela 4.11;
f) Setando o sincronismo do disjuntor DJ814 do AT01 e a seguir o seu fechamento, conforme passos 09 e 10 da Tabela 4.11, normalizando o sistema.

Tabela 4. 11 - Atuação do relé 21p da LT1_345kV + Falha do disjuntor DJ8314 da LT1_345kV.

	Disjuntor LT1_345kV (DJ8314):	Disjuntor LT2_345kV (DJ8324):	Disjuntor Amarre (DJ804):	Disjuntor AT01 (DJ814):	Disjuntor AT02 (DJ824):
00 - Condição Inicial	True	True	True	True	True
01 - Setando o estado de falha do disjuntor DJ8314 da LT01_345kV	True	True	True	True	True
02 - Setando o estado do relé 21P como TRUE (Falta AB na Zona 1)	True	True	False	False	True
03 - Resetando os relés 21P da LT1_345kV	False	True	False	False	True
04 - Resetando os relés 50BF da LT1_345kV	False	True	False	False	True
05 - Resetando os relés 62BF da LT1_345kV	False	True	False	False	True
06 - Resetando o relé de bloqueio para falha de disjuntor A86BF	False	True	False	False	True
07 - Setando o sincronismo do disjuntor de amarre DJ804	False	True	False	False	True
08 - Fechando o disjuntor de amarre DJ804	False	True	True	False	True
09 - Setando o sincronismo do disjuntor DJ814 do AT01	False	True	True	False	True
10 - Fechando o disjuntor DJ814 do AT01	False	True	True	True	True

4.2.5 Proteção de barra normal (em Individual) + Atuação da proteção diferencial de barra – 345kV

O sistema encontra-se funcionando com sua configuração normal como todas as proteções de linha na sua posição NORMAL e a proteção diferencial de barra na posição INDIVIDUAL. Acontece um curto-circuito BT na barra A de 345 kV. Como a chave 29 está na posição INDIVIDUAL, apenas os disjuntores conectados à barra A serão abertos. Nesse caso, o disjuntor de amarre DJ 804 e o disjuntor de linha DJ8314 da LT1_345kV irão abrir. As seguintes ações são realizadas para a proteção e normalização do sistema:

a) Setando o estado do relé diferencial B87B da barra A, atuando a proteção, com a abertura dos disjuntores conectados a barra A, conforme passo 01 da Tabela 4.12 e Figura 4.15;

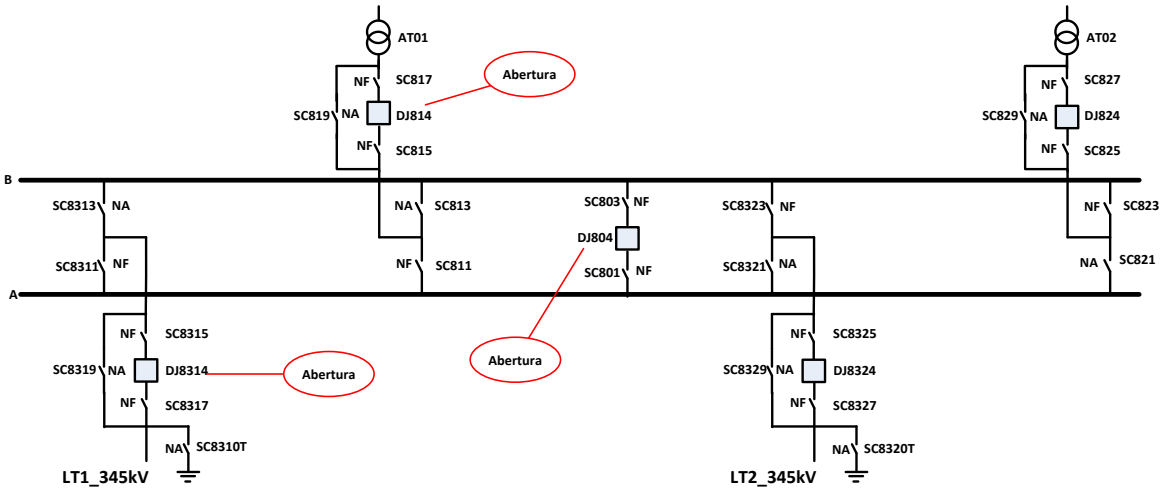

Figura 4. 15 - Abertura dos disjuntores conectados a Barra A.

b) Resetando o estado do relé diferencial B87B da barra A;
c) Resetando o relé A86;
d) Setando o sincronismo do disjuntor de amarre DJ804 e o seu fechamento, conforme passos 04 e 05 da Tabela 4.12;
e) Setando o sincronismo do disjuntor DJ814 do AT01 e o seu fechamento, conforme passos 06 e 07 da Tabela 4.12;
f) Setando o sincronismo do disjuntor DJ8314 da LT1_345kV e o seu fechamento, conforme passos 08 e 09 da Tabela 4.12, normalizando o sistema.

Tabela 4. 12 - Proteção de barra normal (em Individual) + Atuação da proteção diferencial de barra – 345kV.

	Disjuntor LT1_345kV (DJ8314)	Disjuntor LT2_345kV (DJ8324)	Disjuntor Amarre (DJ804)	Disjuntor AT01 (DJ814)	Disjuntor AT02 (DJ824)
00 - Condição Inicial	True	True	True	True	True
01 - Setando o estado do relé diferencial B87B da barra A	False	True	False	False	True
02 - Resetando o estado do relé diferencial B87B da barra A	False	True	False	False	True
03 - Resetar o estado do relé de bloqueio A86	False	True	False	False	True
04 - Setando o sincronismo do disjuntor de amarre DJ804	False	True	False	False	True
05 - Fechando o disjuntor de amarre DJ804	False	True	True	False	True
06 - Setar o sincronismo do disjuntor DJ814 do Trafo AT01	False	True	True	False	True
07 - Fechando o disjuntor DJ814 do Trafo AT01	False	True	True	True	True
08 - Setando o sincronismo do disjuntor DJ8314 da LT1_345kV	False	True	True	True	True
09 - Fechando o disjuntor DJ8314 da LT01_345kV	True	True	True	True	True

4.2.6 Proteção de barra em OVERALL + Atuação da proteção diferencial de barra – 345kV

O sistema encontra-se funcionando com sua configuração normal com todas as proteções de linha na sua posição NORMAL e a proteção diferencial de barra na posição OVERALL. Acontece um curto-circuito na barra A de 345 kV. Como a chave 29 está na posição OVERALL, todos os disjuntores conectados a barra A e B serão abertos. As seguintes ações são realizadas para a proteção e normalização do sistema:

a) Colocando a chave 29 na posição OVERALL;
b) Setando o estado do relé diferencial B87B da barra A, ocasionando a abertura de todos os disjuntores, conforme passo 02 da Tabela 4.13 e Figura 4.16;

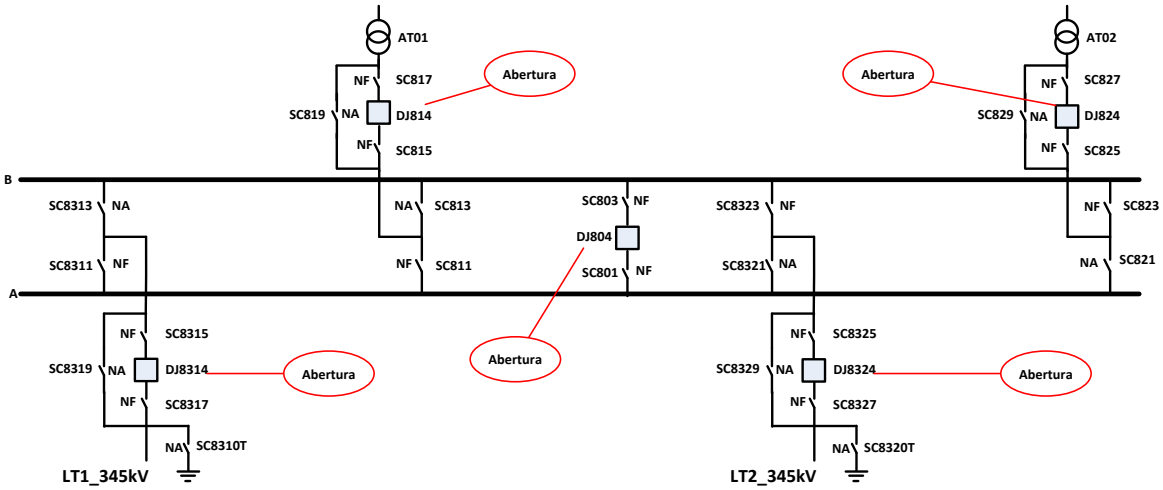

Figura 4. 16 - Abertura de todos os disjuntores.

c) Resetando o estado do relé diferencial B87B da barra A e B;
d) Resetando os estados dos relés de bloqueio A86 e B86;
e) Setando o sincronismo do disjuntor de amarre DJ 804 e o seu fechamento, conforme passo 05 e 06 da Tabela 4.13;
f) Setando o sincronismo do disjuntor DJ814 do AT01 e o seu fechamento, conforme passo 07 e 08 da Tabela 4.13;
g) Setando o sincronismo do disjuntor DJ8314 da LT1_345kV e o seu fechamento, conforme passo 09 e 10 da Tabela 4.13;
h) Setando o sincronismo do disjuntor DJ824 do AT02 e o seu fechamento, conforme passo 11 e 12 da Tabela 4.13;
i) Setando o sincronismo do disjuntor DJ8324 da LT2_345kV e o seu fechamento, conforme passo 13 e 14 da Tabela 4.13.

Tabela 4. 13 - Proteção de barra em OVERALL + Atuação da proteção diferencial de barra – 345kV.

	Disjuntor LT1_345kV (DJ8314)	Disjuntor LT2_345kV (DJ8324)	Disjuntor Amarre (DJ804)	Disjuntor AT01 (DJ814)	Disjuntor AT02 (DJ824)
00 – Condição Inicial	True	True	True	True	True
01 - Colocando a chave 29 na posi??o OVERALL	True	True	True	True	True
02 - Setando o estado do rel? diferencial B87B da barra A	False	False	False	False	False
03 - Resetando o estado do rel? diferencial B87B da barra A e B	False	False	False	False	False
04 - Resetar o estado do rel? de bloqueio A86 e B86	False	False	False	False	False
05 - Setando o sincronismo do disjuntor de amarre DJ804	False	False	False	False	False
06 - Fechando o disjuntor de amarre DJ804	False	False	True	False	False
07 - Setar o sincronismo do disjuntor DJ814 do Trafo AT01	False	False	True	False	False
08 - Fechando o disjuntor DJ814 do Trafo AT01	False	False	True	True	False
09 - Setando o sincronismo do disjuntor DJ8314 da LT1_345kV	False	False	True	True	False
10 - Fechando o disjuntor DJ8314 da LT01_345kV	True	False	True	True	False
11 - Setar o sincronismo do disjuntor DJ824 do Trafo AT02	True	False	True	True	False
12 - Fechando o disjuntor DJ824 do Trafo AT02	True	False	True	True	True
13 - Setando o sincronismo do disjuntor DJ8324 da LT2_345kV	True	False	True	True	True
14 - Fechando o disjuntor DJ8324 da LT02_345kV	True	True	True	True	True

5 CONCLUSÃO

Neste trabalho foi apresentada a implementação de um simulador de intertravamento de chaves seccionadoras de uma subestação, desenvolvido em C#. Primeiramente, fez-se uma descrição dos equipamentos de uma subestação que são relevantes para o desenvolvimento do sistema. A seguir, é determinada a configuração básica da subestação com suas características e forma de operação, que no caso opera com barramento duplo e três níveis de tensão. Então, é apresentado a implementação computacional, onde é demonstrado a classe fundamental e as classes derivadas, bem como a modelagem dos equipamentos que compõem a subestação hipotética em estudo. Por fim, foram efetuados os testes de intertravamento e diversas operações de manobras, onde a ocorrência de falha de algum componente ou ativação de algum relé de proteção aciona a proteção da subestação, ou então, efetua alguma manobra para manutenção de algum equipamento, como o disjuntor ou o autotransformador.

Conclui-se que a implementação computacional para simulação de uma subestação é viável para o treinamento de operadores, por ser facilmente instalado em microcomputadores e também pela facilidade em manipular o código fonte, para tais recursos. Propicia também a oportunidade de se realizar ações que não podem ser realizadas fisicamente e em tempo real. Com isso, melhora-se a atuação de operações no sistema elétrico.

5.1 TRABALHOS FUTUROS

Como continuação dos estudos realizados neste trabalho são sugeridas as seguintes propostas:

- Seria importante dar continuidade ao projeto para o sistema mais amigável para o operador, pois o mesmo se encontra em sua forma mais simples;
- Adaptar o sistema para que se configure de forma mais fácil a outros tipos de subestações, de modo que não se altere o código fonte, ou que se altere o mínimo possível.
- Incrementar o tempo de atuação dos relés e dos equipamentos de proteção, ficando mais próximo da realidade de uma subestação real.

REFERÊNCIAS BIBLIOGRÁFICAS

CLARK, H. K. **Proteção de Sistemas Elétricos de Potência**. 1. ed. Santa Maria - RS: Convênio Eletrobrás/UFSM, v. 7, 1979.

ESPINOZA, R. G. F. **Análise de Proteção de Linhas de Transmissão Através de Relés Numéricos e Uso de Models Externos no ATP**. Dissertação de Mestrado - Unesp - Universidade Estadual Paulista - Ilha Solteira. Ilha Solteira - SP. 2011.

FERREIRA, D. G. **Visão Integrada da Automação da Operação e Manutenção de Sistemas Elétricos de Potência**. Universidade Federal de Minas Gerais - UFMG. Belo Horizonte. 2007.

FILHO, J. M.; MAMEDE, D. R. **Proteção de sistemas elétricos de potência**. 1. ed. Rio de Janeiro: LTC, 2011.

JUNIOR, A. P. et al. **Um sistema de Realidade Virtual para Treinamento de Manutenção e Estudo de uma Unidade Hidrelétrica de Energia**. XII Congreso Argentino de Ciencias de la Computación. Potrero de los Funes, San Luis, Argentina. 2006.

KINDERMANN, G. **Proteção de Sistemas Elétricos de Potência**. 1. ed. Florianópolis - SC: Edição do Autor, v. 2, 2008.

LEITE, M. A. M.; SILVA, D. N.; FILHO, M. R. **Utilização de Realidade Virtual em Treinamento de Operadores e Mantenedores na Usina Hidrelétrica de Tucuruí**. III Simpósio Brasileiro de Sistemas Elétricos. [S.l.]. 2010.

LIBERTY, J.; XIE, D. **Programando C# 3.0**. 5. ed. [S.l.]: Alta Books, 2009.

LIMA, R. T. D. **Desenvolvimento de Software para Medição dos Tempos de Operação Durante Ensaios em Disjuntores de Alta Tensão**. Dissertação de Mestrado - Universidade Federal de Itajubá. Itajubá. 2010.

MACDONALD, M. **Beginning ASP.NET 4.0 in C# 2010**. [S.l.]: APRESS, 2010.

MOUTINHO, J. A. P. **Simulador de Sistemas de Proteção, Controle e Supervisão: Uma aplicação nos Sistemas Elétricos do Tramo Oeste e SE Boa Vista da Eletronorte**. III Simpósio Brasileiro de Sistemas Elétricos. [S.l.]. 2010.

PADMANABHAN, H.; KINDER, R. J. Software Interlocking Topology Schemes for Substation Control Refurbishments. **Developments in Power System Protection, 2008. DPSP 2008. IET 9th International Conference on**, 17-20 March 2008. 584-589.

PAREDES, A. E. R. O. **Integração de Sistemas de Supervisão, Proteção e Automação de Subestações de Energia Elétrica**. Dissertação de Mestrado - Universidade Federal de Itajubá. Itajubá. 2002.

SANTOS, T. C. D. C. **Projeto Básico de Implantação de uma Subestação de 230/138 kV ao Sistema Interligado Nacional**. Projeto de Graduação - Universidade Federal do Rio de Janeiro. Rio de Janeiro. 2010.

SEBESTA, R. W. **Conceitos de Linguagens de Programação**. 5. ed. Porto Alegre: Bookman, 2003.

TAM, E. K. et al. A Web-based virtual environment for operator training. **IEEE Transactions on Power Systems**, v. 14, n. 3, August 1999.

VASQUES, C. M. R. **Interação Entre Autotransformadores de Potência e Solicitações de Alta Frequência do Sistema Elétrico**. Dissertação de Mestrado - Universidade Federal do Rio de Janeiro. Rio de Janeiro. 2011.

APÊNDICES

A. CÓDIGO EM C# DA CLASSE `SIMULADOR`.

No trecho de código a seguir, são instanciados os objetos que compõem o dicionário, contendo toda estrutura e estados dos componentes da SE de 345kV, sendo similares para os outros níveis de tensão.

```
    {
        /// <summary>
        /// Adicionando os componentes do sistema elétrico ao dicionário
        /// </summary>
        dicionario.Add("LT1_345kV", new Linha("LT1_345kV", "ProtLT1_345kV"));
        dicionario.Add("LT2_345kV", new Linha("LT2_345kV", "ProtLT2_345kV"));
        dicionario.Add("AT01_345kV", new Trafo("AT01_345kV",
ObjetoSimulador.ID_BARRA_345KV, "ProtAT01_345kV"));
        dicionario.Add("AT02_345kV", new Trafo("AT02_345kV",
ObjetoSimulador.ID_BARRA_345KV,  "ProtAT02_345kV"));
        dicionario.Add("Amarre345kV", new Amarre("Amarre345kV",
"ProtAmarre345kV"));
        /// <summary>
        /// Adicionando as proteções e as variáveis de monitoramento de conexão
nas barras ao dicionário
        /// </summary>
        dicionario.Add("VarBarra345kV", new VarBarra345kV("VarBarra345kV"));
        dicionario.Add("ProtAmarre345kV", new
ProtAmarre345kV("ProtAmarre345kV"));
        dicionario.Add("ProtLT1_345kV", new
ProtLT1_345kV("ProtLT1_345kV","LT1_345kV"));
        dicionario.Add("ProtLT2_345kV", new
ProtLT2_345kV("ProtLT2_345kV","LT2_345kV"));
        dicionario.Add("ProtAT01_345kV", new ProtAT01_345kV("ProtAT01_345kV",
"AT01_345kV"));
        dicionario.Add("ProtAT02_345kV", new ProtAT02_345kV("ProtAT02_345kV",
"AT02_345kV"));
```

No trecho de código a seguir, é criado o primeiro evento, `EV0001`, com os estados iniciais das chaves seccionadoras e dos disjuntores, para o nível de tensão de 345kV, conforme o diagrama unifilar da SE, ocorrendo de maneira similar aos demais níveis de tensão.

```
        /// <summary>
        /// Adicionando eventos ao dicionário
        /// </summary>
        dicionario.Add("EV0001", new Evento("EV0001"));
        /// <summary>
        /// Inicializando os estados das chaves seccionadoras e do disjuntor
referente a LT1_345kV
        /// </summary>
        (dicionario["LT1_345kV"] as Linha).secBarraA = new
SecLinhaBarraA("SC8311", true, "LT1_345kV", "VarBarra345kV")
        (dicionario["LT1_345kV"] as Linha).secBarraB = new
SecLinhaBarraB("SC8313", false, "LT1_345kV", "VarBarra345kV");
        (dicionario["LT1_345kV"] as Linha).secBypass = new
SecLinhaBypass("SC8319", false, "LT1_345kV", "VarBarra345kV");
        (dicionario["LT1_345kV"] as Linha).secLinhaChegada = new
SecLinhaChegada("SC8315", true, "LT1_345kV");
        (dicionario["LT1_345kV"] as Linha).secLinhaSaida = new
SecLinhaSaida("SC8317", true, "LT1_345kV");
        (dicionario["LT1_345kV"] as Linha).secTerra = new
SecLinhaTerra("SC8310T", false, "LT1_345kV", "VarBarra345kV");
        (dicionario["LT1_345kV"] as Linha).disjuntor = new Disjuntor("DJ8314",
true);
        (dicionario["LT1_345kV"] as Linha).disjuntor.Falha = false;
```

```
        (dicionario["LT1_345kV"] as Linha).sensorLinha = new
SensorLinha("Sensor_LT1_345kV", true);
```

De forma similar ocorre com a declaração dos estados dos componentes dos ATs e da conexão de amarre.

```
        /// <summary>
        /// Inicializando os estados das chaves seccionadoras e do disjuntor
referente a AT01_345kV
        /// </summary>
        (dicionario["AT01_345kV"] as Trafo).secBarraA = new
SecTrafoBarraA("SC811", true, "AT01_345kV", ObjetoSimulador.ID_ESTADO_NORMAL_NF,
"ProtAmarre345kV", "VarBarra345kV");
        (dicionario["AT01_345kV"] as Trafo).secBarraB = new
SecTrafoBarraB("SC813", false, "AT01_345kV", ObjetoSimulador.ID_ESTADO_NORMAL_NA,
"ProtAmarre345kV", "VarBarra345kV");
        (dicionario["AT01_345kV"] as Trafo).secBypass = new
SecTrafoBypass("SC819", false, "AT01_345kV", ObjetoSimulador.ID_BARRA_345KV,
"VarBarra345kV");
        (dicionario["AT01_345kV"] as Trafo).secDisjuntorLadoBarra = new
SecTrafoDisjuntor("SC815", true, "AT01_345kV");
        (dicionario["AT01_345kV"] as Trafo).secDisjuntorLadoTrafo = new
SecTrafoDisjuntor("SC817", true, "AT01_345kV");
        (dicionario["AT01_345kV"] as Trafo).disjuntor = new Disjuntor("DJ814",
true);
        (dicionario["AT01_345kV"] as Trafo).disjuntor.Falha = false;
        /// <summary>
        /// Inicializando os estados das chaves seccionadoras e do disjuntor
referente a conexão do amarre da barra de 345 kV
        /// </summary>
        (dicionario["Amarre345kV"] as Amarre).secBarraA = new SecAmarre("SC801",
true, "Amarre345kV");
        (dicionario["Amarre345kV"] as Amarre).secBarraB = new SecAmarre("SC803",
true, "Amarre345kV");
        (dicionario["Amarre345kV"] as Amarre).disjuntor = new Disjuntor("DJ804",
true);
        (dicionario["Amarre345kV"] as Amarre).disjuntor.Falha = false;
```

A seguir, são declarados os estados dos relés de proteção e das chaves de comando.

```
        /// <summary>
        /// Inicializando os estados dos relés e das chaves de comando
relacionados com a proteção da LT1_345kV
        /// </summary>
        (dicionario["ProtLT1_345kV"] as ProtLT1_345kV).rele21P = new
Rele("21P+68P_LT1_345kV", false);
        (dicionario["ProtLT1_345kV"] as ProtLT1_345kV).rele21A = new
Rele("21S+68P_LT1_345kV", false);
        (dicionario["ProtLT1_345kV"] as ProtLT1_345kV).rele50BF = new
Rele("50BF_LT1_345kV", false);
        (dicionario["ProtLT1_345kV"] as ProtLT1_345kV).rele62BF = new
Rele("62BF_LT1_345kV", false);
        (dicionario["ProtLT1_345kV"] as ProtLT1_345kV).rele79 = new
Rele("79_LT1_345kV", false);
        (dicionario["ProtLT1_345kV"] as ProtLT1_345kV).ch43TD = new
Chave("43TD_LT1_345kV", false);
        /// <summary>
        /// Variável utilizada para inicializar os contatos dos relés de bloqueio
        /// </summary>
        bool[] contatos = { true, true, true, true, true, true, true, true, true,
true };

        /// <summary>
        /// Inicializando os estados dos relés e das chaves de comando
relacionados com a proteção da AT01_345kV
        /// </summary>
```

```
        (dicionario["ProtAT01_345kV"] as ProtAT01_345kV).rele5051A = new
Rele("rele5051A_AT01_345kV", false);
        (dicionario["ProtAT01_345kV"] as ProtAT01_345kV).rele5051B = new
Rele("rele5051B_AT01_345kV", false);
        (dicionario["ProtAT01_345kV"] as ProtAT01_345kV).rele5051C = new
Rele("rele5051C_AT01_345kV", false);
        (dicionario["ProtAT01_345kV"] as ProtAT01_345kV).rele51N = new
Rele("rele51N_AT01_345kV", false);
        (dicionario["ProtAT01_345kV"] as ProtAT01_345kV).rele51 = new
Rele("rele51_AT01_345kV", false);
        (dicionario["ProtAT01_345kV"] as ProtAT01_345kV).rele50 = new
Rele("rele50_AT01_345kV", false);
        (dicionario["ProtAT01_345kV"] as ProtAT01_345kV).rele50BF = new
Rele("rele50BF_AT01_345kV", false);
        (dicionario["ProtAT01_345kV"] as ProtAT01_345kV).rele62BF = new
Rele("rele62BF_AT01_345kV", false);
        (dicionario["ProtAT01_345kV"] as ProtAT01_345kV).rele87A1 = new
Rele("rele87A1_AT01_345kV", false);
        (dicionario["ProtAT01_345kV"] as ProtAT01_345kV).rele87B1 = new
Rele("rele87B1_AT01_345kV", false);
        (dicionario["ProtAT01_345kV"] as ProtAT01_345kV).rele87C1 = new
Rele("rele87C1_AT01_345kV", false);
        (dicionario["ProtAT01_345kV"] as ProtAT01_345kV).rele87A2 = new
Rele("rele87A2_AT01_345kV", false);
        (dicionario["ProtAT01_345kV"] as ProtAT01_345kV).rele87B2 = new
Rele("rele87B2_AT01_345kV", false);
        (dicionario["ProtAT01_345kV"] as ProtAT01_345kV).rele87C2 = new
Rele("rele87C2_AT01_345kV", false);
        (dicionario["ProtAT01_345kV"] as ProtAT01_345kV).rele86_1 = new
ReleBloqueio("rele86_1_AT01_345kV", false, contatos);
        (dicionario["ProtAT01_345kV"] as ProtAT01_345kV).rele86_2 = new
ReleBloqueio("rele86_2_AT01_345kV", false, contatos);
        (dicionario["ProtAT01_345kV"] as ProtAT01_345kV).ch43TD = new
Chave("43TD_AT01_345kV", false);
        /// <summary>
        /// Inicializando os estados dos relés e das chaves de comando
relacionados com a proteção da AT02_345kV
        /// </summary>
        (dicionario["ProtAT02_345kV"] as ProtAT02_345kV).rele5051A = new
Rele("rele5051A_AT02_345kV", false);
        (dicionario["ProtAT02_345kV"] as ProtAT02_345kV).rele5051B = new
Rele("rele5051B_AT02_345kV", false);
        (dicionario["ProtAT02_345kV"] as ProtAT02_345kV).rele5051C = new
Rele("rele5051C_AT02_345kV", false);
        (dicionario["ProtAT02_345kV"] as ProtAT02_345kV).rele51N = new
Rele("rele51N_AT02_345kV", false);
        (dicionario["ProtAT02_345kV"] as ProtAT02_345kV).rele51 = new
Rele("rele51_AT02_345kV", false);
        (dicionario["ProtAT02_345kV"] as ProtAT02_345kV).rele50 = new
Rele("rele50_AT02_345kV", false);
        (dicionario["ProtAT02_345kV"] as ProtAT02_345kV).rele50BF = new
Rele("rele50BF_AT02_345kV", false);
        (dicionario["ProtAT02_345kV"] as ProtAT02_345kV).rele62BF = new
Rele("rele62BF_AT02_345kV", false);
        (dicionario["ProtAT02_345kV"] as ProtAT02_345kV).rele87A1 = new
Rele("rele87A1_AT02_345kV", false);
        (dicionario["ProtAT02_345kV"] as ProtAT02_345kV).rele87B1 = new
Rele("rele87B1_AT02_345kV", false);
        (dicionario["ProtAT02_345kV"] as ProtAT02_345kV).rele87C1 = new
Rele("rele87C1_AT02_345kV", false);
        (dicionario["ProtAT02_345kV"] as ProtAT02_345kV).rele87A2 = new
Rele("rele87A2_AT02_345kV", false);
        (dicionario["ProtAT02_345kV"] as ProtAT02_345kV).rele87B2 = new
Rele("rele87B2_AT02_345kV", false);
        (dicionario["ProtAT02_345kV"] as ProtAT02_345kV).rele87C2 = new
Rele("rele87C2_AT02_345kV", false);
        (dicionario["ProtAT02_345kV"] as ProtAT02_345kV).rele86_1 = new
ReleBloqueio("rele86_1_AT02_345kV", false, contatos);
```

```
        (dicionario["ProtAT02_345kV"] as ProtAT02_345kV).rele86_2 = new
ReleBloqueio("rele86_2_AT02_345kV", false, contatos);
        (dicionario["ProtAT02_345kV"] as ProtAT02_345kV).ch43TD = new
Chave("43TD_AT02_345kV", false);
        /// <summary>
        /// Inicializando os estados dos relés e das chaves de comando
relacionados com a proteção da AT01_138kV
        /// </summary>
        (dicionario["ProtAT01_138kV"] as ProtAT01_138kV).ch43TD = new
Chave("43TD_AT01_138kV", false);
        (dicionario["ProtAT01_138kV"] as ProtAT01_138kV).rele50BF = new
Rele("rele50BF_AT01_138kV", false);
        (dicionario["ProtAT01_138kV"] as ProtAT01_138kV).rele62BF = new
Rele("rele62BF_AT01_138kV", false);
        /// <summary>
        /// Inicializando os estados dos relés e das chaves de comando
relacionados com a proteção da AT02_230kV
        /// </summary>
        (dicionario["ProtAT02_230kV"] as ProtAT02_230kV).ch43TD = new
Chave("43TD_AT02_230kV", false);
        (dicionario["ProtAT02_230kV"] as ProtAT02_230kV).rele50BF = new
Rele("rele50BF_AT02_230kV", false);
        (dicionario["ProtAT02_230kV"] as ProtAT02_230kV).rele62BF = new
Rele("rele62BF_AT02_230kV", false);
        /// <summary>
        /// Inicializando os estados dos relés de bloqueio do esquema de proteção
de amarre 345kV
        /// </summary>
        (dicionario["ProtAmarre345kV"] as ProtAmarre345kV).releA87C_A = new
ReleDiferencial("A87C_A_345kV", false, false);
        (dicionario["ProtAmarre345kV"] as ProtAmarre345kV).releB87B_A = new
ReleDiferencial("B87B_A_345kV", false, false);
        (dicionario["ProtAmarre345kV"] as ProtAmarre345kV).releC87A_A = new
ReleDiferencial("C87A_A_345kV", false, false);
        (dicionario["ProtAmarre345kV"] as ProtAmarre345kV).releA87C_B = new
ReleDiferencial("A87C_B_345kV", false, false);
        (dicionario["ProtAmarre345kV"] as ProtAmarre345kV).releB87B_B = new
ReleDiferencial("B87B_B_345kV", false, false);
        (dicionario["ProtAmarre345kV"] as ProtAmarre345kV).releC87A_B = new
ReleDiferencial("C87A_B_345kV", false, false);
        (dicionario["ProtAmarre345kV"] as ProtAmarre345kV).releA86BF = new
ReleBloqueio("A86BF_345kV", false, contatos);
        (dicionario["ProtAmarre345kV"] as ProtAmarre345kV).releB86BF = new
ReleBloqueio("B86BF_345kV", false, contatos);
        (dicionario["ProtAmarre345kV"] as ProtAmarre345kV).releA86 = new
ReleBloqueio("A86_345kV", false, contatos);
        (dicionario["ProtAmarre345kV"] as ProtAmarre345kV).releB86 = new
ReleBloqueio("B86_345kV", false, contatos);
        (dicionario["ProtAmarre345kV"] as ProtAmarre345kV).ch29 = new
Chave29("CH29_345kV", Chave29.Estados.Individual);
        (dicionario["ProtAmarre345kV"] as ProtAmarre345kV).ch43IB = new
Chave43IB("43IB_345kV", Chave43IB.Estados.Off);
    }
    /// <summary>
    /// Atualiza as entidades de simulacao.
    /// </summary>
    /// <param name="dt">
    /// A <see cref="System.Single"/>
    /// </param>
    public static void Update(float dt)
    {
        foreach (ObjetoSimulador os in dicionario.Values)
        {
            os.Update(dt);
        }
    }
}
```

B. CÓDIGO EM C# DA CLASSE EVENTO.

```
public class Evento : ObjetoSimulador
{
    public ObjetoSimulador obj;
    public static string mensagem;
    private const int NumMaxEventos = 150;
    private bool[] Disparo = new bool[NumMaxEventos];
    private float[] Timer = new float[NumMaxEventos];
    public static int manobra;
    public Evento(string ID) : base(ID)
    {
        for (int i = 0; i < NumMaxEventos; i++)
        {
            Disparo[i] = false;
            Timer[i] = (i + 1) * (2.0f);
        }
    }
    public override void Update(float dt)
    {
        for (int i = 0; i < NumMaxEventos; i++)
        {
            Timer[i] -= dt;
        }
        switch (manobra)
```

A seguir, apresenta-se a declaração da classe Evento, com a apresentação de duas declarações representando uma manobra e um teste de intertravamento, sendo que as demais manobras e testes ocorrem de forma similar.

```
  {

            case 0:
                /// <summary>
                /// MANOBRA: Atuação da Proteção da LT1_345kV e normalização do
sistema
                /// DESCRIÇÃO: O sistema encontra-se funcionando com sua
configuração normal com todas
                /// as proteções de linha na sua posição NORMAL. Acontece um
curto-circuito na Linha de Transmissão
                /// LT1_345kV, ocorrendo a abertura do disjuntor DJ8314. As
seguintes ações são realizadas para
                /// normalização do sistema:
                ///                         1 - Setar o relé 21P como FALSE
                ///                         2 - Setar o sincronismo do disjuntor de
linha DJ8314 da LT1_345kV
                ///                         3 - Fechar o disjuntor DJ8314 da
LT1_345kV
                /// </summary>
                if (Timer[0] < 0 && !Disparo[0])
                {
                    //Setando o estado da proteção da LT1_345kV como TRUE
                    (Simulador.FindObject("ProtLT1_345kV") as
ProtLT1_345kV).rele21P.Estado = true;
                    mensagem = "1 - Estado da proteção TRUE";
                    Disparo[0] = true;
                }
                if (Timer[1] < 0 && !Disparo[1])
                {
                    //Setando o estado da proteção da LT1_345kV como FALSE
                    (Simulador.FindObject("ProtLT1_345kV") as
ProtLT1_345kV).rele21P.Estado = false;
                    mensagem = "2 - Estado da proteção FALSE";
                    Disparo[1] = true;
                }
```

```
                if (Timer[2] < 0 && !Disparo[2])
                {
                    //Setando a lógica do sincronismo do disjuntor em TRUE
                    (Simulador.FindObject("LT1_345kV") as
Linha).disjuntor.Sincronismo = true;
                    mensagem = "3 - Sincronismo em TRUE";
                    Disparo[2] = true;
                }
                if (Timer[3] < 0 && !Disparo[3])
                {
                    //Fechando o disjuntor da LT1_345kV
                    (Simulador.FindObject("LT1_345kV") as
Linha).disjuntor.Comando = Disjuntor.Comandos.Fechar;
                    mensagem = "4 - Fechando o disjuntor";
                    Disparo[3] = true;
                }
                break;


            case 99:
                /// <summary>
                /// MANOBRA: Proteção da LT2_345kV TRANSFERIDA + Atuação do relé
21P - TESTE DE INTERTRAVAMENTO
                /// </summary>
                if (Timer[0] < 0 && !Disparo[0])
                {
                    //Colocar a chave 29 da proteção de barra na posição OVERALL
                    (Simulador.FindObject("ProtAmarre345kV") as
ProtAmarre).ch29.Comando = Chave29.Estados.Overall;
                    mensagem = "Ch29 em OVERALL";
                    Disparo[0] = true;
                }
                if (Timer[1] < 0 && !Disparo[1])
                {
                    //Fechando a seccionadora SC 821
                    (Simulador.FindObject("AT02_345kV") as
Trafo).secBarraA.Comando = Seccionadora.Comandos.Fechar;
                    mensagem = "Fechando SC821";
                    Disparo[1] = true;
                }
                if (Timer[2] < 0 && !Disparo[2])
                {
                    //Abrir a seccionadora SC823
                    (Simulador.FindObject("AT02_345kV") as
Trafo).secBarraB.Comando = Seccionadora.Comandos.Abrir;
                    //Abrir o disjuntor DJ8324 da LT2_345kV
                    (Simulador.FindObject("LT2_345kV") as
Linha).disjuntor.Comando = Disjuntor.Comandos.Abrir;
                    mensagem = "Abrir SC823 e DJ8324";
                    Disparo[2] = true;
                }
                if (Timer[3] < 0 && !Disparo[3])
                {
                    //Abrir as chaves seccionadoras SC8325 e SC8327 de conexão do
disjuntor
                    (Simulador.FindObject("LT2_345kV") as
Linha).secLinhaChegada.Comando = Seccionadora.Comandos.Abrir;
                    (Simulador.FindObject("LT2_345kV") as
Linha).secLinhaSaida.Comando = Seccionadora.Comandos.Abrir;
                    mensagem = "Abrir Sec SC8325 e SC8327";
                    Disparo[3] = true;
                }
                if (Timer[4] < 0 && !Disparo[4])
                {
                    //Abrir o disjuntor de amarre DJ804
                    (Simulador.FindObject("Amarre345kV") as
Amarre).disjuntor.Comando = Disjuntor.Comandos.Abrir;
                    mensagem = "Abrir DJ804";
```

```
                    Disparo[4] = true;
                }
                if (Timer[5] < 0 && !Disparo[5])
                {
                    //Abrir as seccionadoras do amarre SC801 e SC803
                    (Simulador.FindObject("Amarre345kV") as
Amarre).secBarraA.Comando = Seccionadora.Comandos.Abrir;
                    (Simulador.FindObject("Amarre345kV") as
Amarre).secBarraB.Comando = Seccionadora.Comandos.Abrir;
                    mensagem = "Abrir SC801 e SC803";
                    Disparo[5] = true;
                }
                if (Timer[6] < 0 && !Disparo[6])
                {
                    //Colocar a chave 43TD do DJ8324 na posição TRANSFERIDA
                    (Simulador.FindObject("ProtLT2_345kV") as
ProtLinha).ch43TD.Comando = true;
                    mensagem = "Ch 43TD do DJ8324 TRANSFERIDA";
                    Disparo[6] = true;
                }
                if (Timer[7] < 0 && !Disparo[7])
                {
                    //Setar o Estado da Chave 43IB na Barra B
                    (Simulador.FindObject("ProtAmarre345kV") as
ProtAmarre).ch43IB.Comando = Chave43IB.Estados.B;
                    mensagem = "Setar Ch 43IB em B";
                    Disparo[7] = true;
                }
                if (Timer[8] < 0 && !Disparo[8])
                {
                    //Fechando a chave de bypass SC8329 do DJ8324
                    (Simulador.FindObject("LT2_345kV") as
Linha).secBypass.Comando = Seccionadora.Comandos.Fechar;
                    mensagem = "Fechando Bypass SC8329 do DJ8324";
                    Disparo[8] = true;
                }
                if (Timer[9] < 0 && !Disparo[9])
                {
                    //Fechando as seccionadoras SC801 e SC803 do DJ804
                    (Simulador.FindObject("Amarre345kV") as
Amarre).secBarraA.Comando = Seccionadora.Comandos.Fechar;
                    (Simulador.FindObject("Amarre345kV") as
Amarre).secBarraB.Comando = Seccionadora.Comandos.Fechar;
                    mensagem = "Fechando SC801 e SC803 do DJ804";
                    Disparo[9] = true;
                }
                if (Timer[10] < 0 && !Disparo[10])
                {
                    //Fechando o disjuntor DJ804
                    (Simulador.FindObject("Amarre345kV") as
Amarre).disjuntor.Comando = Disjuntor.Comandos.Fechar;
                    mensagem = "TESTE IT - Fechando DJ804";
                    Disparo[10] = true;
                }
                if (Timer[11] < 0 && !Disparo[11])
                {
                    //Setando o sincronismo do DJ804
                    (Simulador.FindObject("Amarre345kV") as
Amarre).disjuntor.Sincronismo = true;
                    mensagem = "Sincronismo DJ804";
                    Disparo[11] = true;
                }
                if (Timer[12] < 0 && !Disparo[12])
                {
                    //Fechando o disjuntor DJ804
                    (Simulador.FindObject("Amarre345kV") as
Amarre).disjuntor.Comando = Disjuntor.Comandos.Fechar;
                    mensagem = "Fechando DJ804";
```

```
                    Disparo[12] = true;
                }
                if (Timer[13] < 0 && !Disparo[13])
                {
                    //Setando a proteção 21P da LT2_345kV em TRUE
                    (Simulador.FindObject("ProtLT2_345kV") as
ProtLT2_345kV).rele21P.Estado = true;
                    mensagem = "Protecao 21P em TRUE";
                    Disparo[13] = true;
                }
                if (Timer[14] < 0 && !Disparo[14])
                {
                    //Setando a proteção 21P da LT2_345kV em FALSE
                    (Simulador.FindObject("ProtLT2_345kV") as
ProtLT2_345kV).rele21P.Estado = false;
                    mensagem = "Protecao 21P em FALSE";
                    Disparo[14] = true;
                }
                if (Timer[15] < 0 && !Disparo[15])
                {
                    //Setando o sincronismo do DJ804
                    (Simulador.FindObject("Amarre345kV") as
Amarre).disjuntor.Sincronismo = true;
                    mensagem = "Sincronismo DJ804";
                    Disparo[15] = true;
                }
                if (Timer[16] < 0 && !Disparo[16])
                {
                    //Fechando o disjuntor DJ804
                    (Simulador.FindObject("Amarre345kV") as
Amarre).disjuntor.Comando = Disjuntor.Comandos.Fechar;
                    mensagem = "Fechando DJ804";
                    Disparo[16] = true;
                }

                if (Timer[17] < 0 && !Disparo[17])
                {
                    //Abrindo a chave de bypass SC8329 do DJ8324
                    (Simulador.FindObject("LT2_345kV") as
Linha).secBypass.Comando = Seccionadora.Comandos.Abrir;
                    mensagem = "TESTE IT - Abrindo Bypass SC8329 do DJ8324";
                    Disparo[17] = true;
                }
                if (Timer[18] < 0 && !Disparo[18])
                {
                    //Abrindo as seccionadoras SC801 e SC803 do DJ804
                    (Simulador.FindObject("Amarre345kV") as
Amarre).secBarraA.Comando = Seccionadora.Comandos.Abrir;
                    (Simulador.FindObject("Amarre345kV") as
Amarre).secBarraB.Comando = Seccionadora.Comandos.Abrir;
                    mensagem = "TESTE de IT - Abrindo SC801 e SC803 do DJ804";
                    Disparo[18] = true;
                }
                break;
```

C. CÓDIGO EM C# DA CLASSE MAIN.

```
class MainClass
{
    /// <summary>
    /// Permite acesso ao simulador atraves da linha de comando.
    /// </summary>
    /// <param name="args">
    /// A <see cref="System.String[]"/>
    /// </param>
    public static void Main(string[] args)
    {
        //
        //
        Simulador.Initialise();
        int numManobras = 150;
        do
        {
            System.Console.WriteLine("Escolha uma das manobras:");
            System.Console.WriteLine("   1 - Proteção NORMAL + Atuação do relé
21P da LT1_345kV");
            System.Console.WriteLine("   2 - Proteção NORMAL + Atuação do relé
21P da LT2_345kV");
            System.Console.WriteLine("   3 - Proteção NORMAL + Atuação do relé
21P da LT1_230kV");
            System.Console.WriteLine("   4 - Proteção NORMAL + Atuação do relé
21P da LT2_230kV");
            System.Console.WriteLine("   5 - Proteção NORMAL + Atuação do relé
21P da LT1_138kV");
            System.Console.WriteLine("   6 - Proteção NORMAL + Atuação do relé
21P da LT2_138kV");
            System.Console.WriteLine("   7 - Proteção TRANSFERIDA + Atuação do
relé 21P da LT1_345kV");
            System.Console.WriteLine("   8 - Proteção TRANSFERIDA + Atuação do
relé 21P da LT2_345kV");
            System.Console.WriteLine("   9 - Proteção TRANSFERIDA + Atuação do
relé 21P da LT1_230kV");
            System.Console.WriteLine("  10 - Proteção TRANSFERIDA + Atuação do
relé 21P da LT2_230kV");
            System.Console.WriteLine("  11 - Proteção TRANSFERIDA + Atuação do
relé 21P da LT1_138kV");
            System.Console.WriteLine("  12 - Proteção TRANSFERIDA + Atuação do
relé 21P da LT2_138kV");
            System.Console.WriteLine("  13 - Proteção NORMAL + Atuação da
proteção diferencial de barra da LT1_345kV");
            System.Console.WriteLine("  14 - Proteção NORMAL + Atuação da
proteção diferencial de barra da LT2_345kV");
            System.Console.WriteLine("  15 - Proteção NORMAL + Atuação da
proteção diferencial de barra da LT1_230kV");
            System.Console.WriteLine("  16 - Proteção NORMAL + Atuação da
proteção diferencial de barra da LT2_230kV");
            System.Console.WriteLine("  17 - Proteção NORMAL + Atuação da
proteção diferencial de barra da LT1_138kV");
            System.Console.WriteLine("  18 - Proteção NORMAL + Atuação da
proteção diferencial de barra da LT2_138kV");
            System.Console.WriteLine("  19 - Atuação do relé 21P da LT1_345kV
+ Falha do disjuntor DJ8314 da LT1_345kV");
            System.Console.WriteLine("  20 - Atuação do relé 21P da LT2_345kV
+ Falha do disjuntor DJ8324 da LT2_345kV");
            System.Console.WriteLine("  21 - Atuação do relé 21P da LT1_230kV
+ Falha do disjuntor DJ8314 da LT1_230kV");
            System.Console.WriteLine("  22 - Atuação do relé 21P da LT2_230kV
+ Falha do disjuntor DJ8324 da LT2_230kV");
            System.Console.WriteLine("  23 - Atuação do relé 21P da LT1_138kV
+ Falha do disjuntor DJ8314 da LT1_138kV");
            System.Console.WriteLine("  24 - Atuação do relé 21P da LT2_138kV
+ Falha do disjuntor DJ8324 da LT2_138kV");
```

```
            System.Console.WriteLine("  25 - Proteção de barra Normal (em
INDIVIDUAL) + Atuação da proteção diferencial de barra - 345kV");
            System.Console.WriteLine("  26 - Proteção de barra em OVERALL +
Atuação da proteção diferencial de barra - 345kV");
            System.Console.WriteLine("  27 - Proteção do AT01 TRANSFERIDA +
Atuação da proteção do AT01 - 345kV");
            System.Console.WriteLine("  28 - Atuação do relé 5051A do
AT01_345kV + Falha do disjuntor DJ814 do AT01_345kV");
            System.Console.WriteLine("  100 - TESTE DE INTERTRAVAMENTO -
Proteção TRANSFERIDA + Atuação do relé 21P da LT2_345kV");
            System.Console.WriteLine("  101 - TESTE DE INTERTRAVAMENTO -
Proteção TRANSFERIDA + Atuação do relé 21P da LT1_230kV");
            System.Console.WriteLine("  102 - TESTE DE INTERTRAVAMENTO - Abrir
Chaves do Amarre");
            System.Console.WriteLine("  103 - TESTE DE INTERTRAVAMENTO - Abrir
chaves dos disjuntores das LT's");
            System.Console.WriteLine("  104 - TESTE DE INTERTRAVAMENTO -
Fechar Bypass dos disjuntores das LT's");

            System.Console.Write("Manobra: ");
            Evento.manobra = Convert.ToInt32(System.Console.ReadLine());
            if ((Evento.manobra < 0) || (Evento.manobra > numManobras))
            {
                System.Console.Clear();
            }
        } while ((Evento.manobra < 0) || (Evento.manobra > numManobras));

        System.Console.Clear();
        Evento.manobra--;

        DateTime currentTime, previousTime;
        previousTime = DateTime.Now;
        // contador
        // int nrVezes = 0;
       // while (nrVezes < 100)
        while (true)
        {
            currentTime = DateTime.Now;
            Simulador.Update((float)(currentTime -
previousTime).TotalSeconds);
            Debug();
            previousTime = currentTime;

         //   nrVezes++;
        }
    }
    /// <summary>
    /// Funcao usada para fins de depuracao.
    /// Qualquer acesso ao System.Console deve ser feito neste metodo.
    /// </summary>
    public static void Debug()
    {
 //
        string linha = "";
        StreamWriter arqsaida1 = new
StreamWriter("C:\\Manobras\\Manobra1.txt", true, Encoding.ASCII);
        StreamWriter arqsaida2 = new
StreamWriter("C:\\Manobras\\Manobra2.txt", true, Encoding.ASCII);
        StreamWriter arqsaida3 = new
StreamWriter("C:\\Manobras\\Manobra3.txt", true, Encoding.ASCII);
        StreamWriter arqsaida4 = new
StreamWriter("C:\\Manobras\\Manobra4.txt", true, Encoding.ASCII);
        StreamWriter arqsaida5 = new
StreamWriter("C:\\Manobras\\Manobra5.txt", true, Encoding.ASCII);
        StreamWriter arqsaida6 = new
StreamWriter("C:\\Manobras\\Manobra6.txt", true, Encoding.ASCII);
        StreamWriter arqsaida7 = new
StreamWriter("C:\\Manobras\\Manobra7.txt", true, Encoding.ASCII);
```

```
        StreamWriter arqsaida8 = new
StreamWriter("C:\\Manobras\\Manobra8.txt", true, Encoding.ASCII);
        StreamWriter arqsaida9 = new
StreamWriter("C:\\Manobras\\Manobra9.txt", true, Encoding.ASCII);
        StreamWriter arqsaida10 = new
StreamWriter("C:\\Manobras\\Manobra10.txt", true, Encoding.ASCII);
        StreamWriter arqsaida11 = new
StreamWriter("C:\\Manobras\\Manobra11.txt", true, Encoding.ASCII);
        StreamWriter arqsaida12 = new
StreamWriter("C:\\Manobras\\Manobra12.txt", true, Encoding.ASCII);
        StreamWriter arqsaida13 = new
StreamWriter("C:\\Manobras\\Manobra13.txt", true, Encoding.ASCII);
        StreamWriter arqsaida14 = new
StreamWriter("C:\\Manobras\\Manobra14.txt", true, Encoding.ASCII);
        StreamWriter arqsaida15 = new
StreamWriter("C:\\Manobras\\Manobra15.txt", true, Encoding.ASCII);
        StreamWriter arqsaida16 = new
StreamWriter("C:\\Manobras\\Manobra16.txt", true, Encoding.ASCII);
        StreamWriter arqsaida17 = new
StreamWriter("C:\\Manobras\\Manobra17.txt", true, Encoding.ASCII);
        StreamWriter arqsaida18 = new
StreamWriter("C:\\Manobras\\Manobra18.txt", true, Encoding.ASCII);
        StreamWriter arqsaida19 = new
StreamWriter("C:\\Manobras\\Manobra19.txt", true, Encoding.ASCII);
        StreamWriter arqsaida20 = new
StreamWriter("C:\\Manobras\\Manobra20.txt", true, Encoding.ASCII);
        StreamWriter arqsaida21 = new
StreamWriter("C:\\Manobras\\Manobra21.txt", true, Encoding.ASCII);
        StreamWriter arqsaida22 = new
StreamWriter("C:\\Manobras\\Manobra22.txt", true, Encoding.ASCII);
        StreamWriter arqsaida23 = new
StreamWriter("C:\\Manobras\\Manobra23.txt", true, Encoding.ASCII);
        StreamWriter arqsaida24 = new
StreamWriter("C:\\Manobras\\Manobra24.txt", true, Encoding.ASCII);
        StreamWriter arqsaida25 = new
StreamWriter("C:\\Manobras\\Manobra25.txt", true, Encoding.ASCII);
        StreamWriter arqsaida26 = new
StreamWriter("C:\\Manobras\\Manobra26.txt", true, Encoding.ASCII);
        StreamWriter arqsaida27 = new
StreamWriter("C:\\Manobras\\Manobra27.txt", true, Encoding.ASCII);
        StreamWriter arqsaida28 = new
StreamWriter("C:\\Manobras\\Manobra28.txt", true, Encoding.ASCII);
        StreamWriter arqsaida100 = new
StreamWriter("C:\\Manobras\\Manobra100.txt", true, Encoding.ASCII);
        StreamWriter arqsaida101 = new
StreamWriter("C:\\Manobras\\Manobra101.txt", true, Encoding.ASCII);
        StreamWriter arqsaida102 = new
StreamWriter("C:\\Manobras\\Manobra102.txt", true, Encoding.ASCII);
        StreamWriter arqsaida103 = new
StreamWriter("C:\\Manobras\\Manobra103.txt", true, Encoding.ASCII);
        StreamWriter arqsaida104 = new
StreamWriter("C:\\Manobras\\Manobra104.txt", true, Encoding.ASCII);
        switch (Evento.manobra)
```

Segue, a descrição das manobras do `case 0` e do `case 99`, sendo as demais manobras são similares.

```
            case 0:
                linha = Evento.mensagem + " | Disjuntor: |" +
(Simulador.FindObject("LT1_345kV") as Linha).disjuntor.Estado + " | " +
                                        "Sincronismo: |" +
(Simulador.FindObject("LT1_345kV") as Linha).disjuntor.Sincronismo + " | " +
                                        "Rele21P: |" +
(Simulador.FindObject("ProtLT1_345kV") as ProtLT1_345kV).rele21P.Estado;
                arqsaida1.WriteLine(linha);
                System.Console.WriteLine(linha);
                break;
```

```
            case 99:
                linha = Evento.mensagem + " | Disjuntor: |" +
(Simulador.FindObject("LT2_345kV") as Linha).disjuntor.Estado + " | " +
                                        "Disjuntor Amarre: |" +
(Simulador.FindObject("Amarre345kV") as Amarre).disjuntor.Estado + " | " +
                                        "Sec LT Linha Chegada: |" +
(Simulador.FindObject("LT2_345kV") as Linha).secLinhaChegada.Estado + " | " +
                                        "Sec LT Linha Saida: |" +
(Simulador.FindObject("LT2_345kV") as Linha).secLinhaSaida.Estado + " | " +
                                        "Sec LT ByPass: |" +
(Simulador.FindObject("LT2_345kV") as Linha).secBypass.Estado + " | " +
                                        "Sec AT Barra A: |" +
(Simulador.FindObject("AT02_345kV") as Trafo).secBarraA.Estado + " | " +
                                        "Sec AT Barra B: |" +
(Simulador.FindObject("AT02_345kV") as Trafo).secBarraB.Estado + " | " +
                                        "Sec Amarre Barra A: |" +
(Simulador.FindObject("Amarre345kV") as Amarre).secBarraA.Estado + " | " +
                                        "Sec Amarre Barra B: |" +
(Simulador.FindObject("Amarre345kV") as Amarre).secBarraB.Estado + " | " +
                                        "Sec DJ Amarre: |" +
(Simulador.FindObject("Amarre345kV") as Amarre).disjuntor.Estado + " | " +
                                        "Prot Amarre CH43IB: |" +
(Simulador.FindObject("ProtAmarre345kV") as ProtAmarre).ch43IB.Estado + " | "
+
                                        "Prot LT CH43TD: |" +
(Simulador.FindObject("ProtLT2_345kV") as ProtLT2_345kV).ch43TD.Estado + " | "
+
                                        "Prot LT Rele21P: |" +
(Simulador.FindObject("ProtLT2_345kV") as ProtLT2_345kV).rele21P.Estado;
                arqsaida100.WriteLine(linha);
                System.Console.WriteLine(linha);
                break;

            default:
                break;
        }

    arqsaida1.Close();
        arqsaida2.Close();
        arqsaida3.Close();
        arqsaida4.Close();
        arqsaida5.Close();
        arqsaida6.Close();
        arqsaida7.Close();
        arqsaida8.Close();
        arqsaida9.Close();
        arqsaida10.Close();
        arqsaida11.Close();
        arqsaida12.Close();
        arqsaida13.Close();
        arqsaida14.Close();
        arqsaida15.Close();
        arqsaida16.Close();
        arqsaida17.Close();
        arqsaida18.Close();
        arqsaida19.Close();
        arqsaida20.Close();
        arqsaida21.Close();
        arqsaida22.Close();
        arqsaida23.Close();
        arqsaida24.Close();
        arqsaida25.Close();
        arqsaida26.Close();
        arqsaida27.Close();
```

```
            arqsaida28.Close();
            arqsaida100.Close();
            arqsaida101.Close();
            arqsaida102.Close();
            arqsaida103.Close();
            arqsaida104.Close();
                }
}
```

D. CÓDIGO EM C# DA CLASSE DISJUNTOR.

```
public class Disjuntor : ObjetoSimulador
{
    // Declarando os tipos de comando do disjuntor
    public enum Comandos
    {
        Abrir,
        Fechar,
        Neutro
    };
    // Declarando as variáveis de estado e comando do disjuntor
    protected bool estado;
    protected bool sincronismo;
    protected bool falha;
    protected bool logicaFechamento;
    protected Comandos comando;
    /// Retorna o estado do disjuntor
    public bool Estado
    {
        get { return estado; }
    }
    // Retorna ou seta o sincronismo do disjuntor
    public bool Sincronismo
    {
        get { return sincronismo; }
        set { sincronismo = value; }
    }
    // Retorna ou seta o estado de falha do disjuntor
    public bool Falha
    {
        get { return falha; }
        set { falha = value; }
    }
    // Retorna ou seta o comando do disjuntor
    public Comandos Comando
    {
        get { return comando; }
        set { comando = value; }
    }
    // Retorna ou seta o estado da lógica de fechamento do disjuntor
    public bool LogicaFechamento
    {
        get { return logicaFechamento; }
        set { logicaFechamento = value; }
    }
    // Construtor da classe Disjuntor
    public Disjuntor(string ID, bool estado) : base(ID)
    {
        this.estado = estado;
        sincronismo = false;
        comando = Comandos.Neutro;
    }
    // Atualiza objeto do tipo Disjuntor
    public override void Update(float dt)
    {
        if ((comando == Comandos.Abrir) && (estado == true) && !falha)
        {
            // Abrindo o disjuntor, caso as condições sejam satisfeitas
            estado = false;
        }
        else if ((comando == Comandos.Fechar) && (estado == false) &&
logicaFechamento && sincronismo)
        {
            // Fechando o disjuntor, caso as condições sejam satisfeitas
            estado = true;
            sincronismo = false;
```

```
            }
            else
            {
                // Setando a lógica do fechamento como false
                logicaFechamento = false;
            }
            comando = Comandos.Neutro;
        }
    }
```

E. CÓDIGO EM C# DA CLASSE SECCIONADORA.

```
public class Seccionadora : ObjetoSimulador
{
    // Declarando os tipos de comando de uma chave seccionadora
    public enum Comandos
    {
        Abrir,
        Fechar,
        Neutro
    };
    // Declarando as variáveis de estado e comando da chave seccionadora
    protected bool estado;
    protected Comandos comando;
    // Retorna o estado da chave seccionadora
    public bool Estado
    {
        get { return estado; }
    }
    // Retorna ou seta o comando da chave seccionadora
    public Comandos Comando
    {
        get { return comando;}
        set { comando = value;}
    }
    // Construtor da classe Seccionadora
    public Seccionadora(string ID, bool estado) : base(ID)
       {
    // Inicializando o estado e o comando da chave
        this.estado = estado;
        comando = Comandos.Neutro;
       }
}
```

F. CÓDIGO EM C# DA CLASSE SECAMARRE.

```
public class SecAmarre : Seccionadora
{
    // Declarando a variável que armazena o ID do amarre ao qual a
seccionadora pertence
    public string IDAmarre;
    // Construtor da classe SecAmarre
    public SecAmarre(string ID, bool estado, string IDAmarre) : base(ID,
estado)
    {
        this.IDAmarre = IDAmarre;
    }
    // Atualiza objeto do tipo SecAmarre
    public override void Update(float dt)
    {
        // Lendo os estados das variáveis da lógica de intertravamento
        bool disjuntor = (Simulador.FindObject(IDAmarre) as
Amarre).disjuntor.Estado;
        // Implementação da lógica de intertravamento
        bool saida = (!disjuntor);
        // Verificando a manobra da chave
        if (saida && (comando != Comandos.Neutro))
        {
            estado = (comando == Comandos.Fechar) ? true : false;
        }
        comando = Comandos.Neutro;
    }
}
```

G. CÓDIGO EM C# DA CLASSE AMARRE.

```
public class Amarre : ObjetoSimulador
{
    // Declarando os componentes de um elemento amarre
    public Disjuntor disjuntor;
    public SecAmarre secBarraA;
    public SecAmarre secBarraB;
    // Declarando a variável que armazena o nome da proteção do amarre
    private string IDProtecao;

    // Construtor da classe Amarre
    public Amarre(string ID, string IDProtecao) : base(ID)
    {
        this.IDProtecao = IDProtecao;
    }
    // Atualiza objeto do tipo Amarre
    public override void Update(float dt)
    {
     // Verificando o estado das variáveis utilizadas na lógica de abertura e
     // fechamento do disjuntor de amarre
        bool trip = (Simulador.FindObject(IDProtecao) as ProtAmarre).TRIP;
        bool bq = (Simulador.FindObject(IDProtecao) as ProtAmarre).BQ;
        // Testando a lógica de abertura do disjuntor de amarre
        if (disjuntor.Estado && (trip || bq))
        {
            disjuntor.Comando = Disjuntor.Comandos.Abrir;
        }
        // Testando a lógica de fechamento do disjuntor de amarre
        if (!disjuntor.Estado && !(trip || bq) && (disjuntor.Comando ==
Disjuntor.Comandos.Fechar))
        {
            disjuntor.LogicaFechamento = true;
        }
        // Atualizando objetos que compõem um objeto do tipo Amarre
        disjuntor.Update(dt);
        secBarraA.Update(dt);
        secBarraB.Update(dt);
    }
}
```

H. CÓDIGO EM C# DA CLASSE SECLINHABARRAA.

```
public class SecLinhaBarraA : Seccionadora
{
    // Declarando as variáveis que armazenam o ID da linha e do grupo de
comandos ao quais a seccionadora pertence
    public string IDLinha;
    public string IDVariaveisBarra;
    // Construtor da classe SecLinhaBarraA
    public SecLinhaBarraA(string ID, bool estado, string IDLinha, string
IDVariaveisBarra) : base(ID, estado)
    {
        // Inicializando os IDs da linha e das variáveis de monitoramento de
        // conexão ao barramento
        this.IDLinha = IDLinha;
        this.IDVariaveisBarra = IDVariaveisBarra;
    }
    // Atualiza objeto do tipo SecLinhaBarraA
    public override void Update(float dt)
    {
        // Verificando os estados de todas as variáveis utilizadas na lógica
de
        // intertravamento
        bool secBarraB = (Simulador.FindObject(IDLinha) as
Linha).secBarraB.Estado;
        bool secBypass = (Simulador.FindObject(IDLinha) as
Linha).secBypass.Estado;
        bool disjuntor = (Simulador.FindObject(IDLinha) as
Linha).disjuntor.Estado;
        bool BPBX = (Simulador.FindObject(IDVariaveisBarra) as VarBarra).BPBX;
        bool BTAX = (Simulador.FindObject(IDVariaveisBarra) as VarBarra).BTAX;
        bool ch43NX = (Simulador.FindObject(IDVariaveisBarra) as
VarBarra).Ch43NX;
        bool BX = (Simulador.FindObject(IDVariaveisBarra) as VarBarra).BX;
        // Implementando a lógica de intertravamento
        bool teste1, teste2, teste3, saida;
        teste1 = (BTAX && secBarraB && ch43NX && BPBX);
        teste2 = (BPBX && !secBarraB && !disjuntor);
        teste3 = (!secBarraB && !disjuntor && !secBypass && BX);
        saida = (teste1 || teste2 || teste3);
        // Verificando a manobra da chave, a depender da lógica de
intertravamento e da emissão de um comando
        if (saida && (comando != Comandos.Neutro))
        {
            estado = (comando == Comandos.Fechar) ? true : false;
        }
        comando = Comandos.Neutro;
    }
}
```

I. CÓDIGO EM C# DA CLASSE LINHA.

```
public class Linha : ObjetoSimulador
{
    // Declarando os componentes de um elemento linha
    public SecLinhaBarraA secBarraA;
    public SecLinhaBarraB secBarraB;
    public SecLinhaBypass secBypass;
    public SecLinhaChegada secLinhaChegada;
    public SecLinhaSaida secLinhaSaida;
    public SecLinhaTerra secTerra;
    public Disjuntor disjuntor;
    // Declarando a variável que armazena o nome da proteção da linha
    private string IDProtecao;
    // Declarando a variável sensorLinha de linha, utilizada principalmente para
identificar
    // se os terminais remotos estão energizados. Seu estado é utilizado nas
lógicas de
    // proteção e de intertravamento das seccionadoras
    public SensorLinha sensorLinha;
    // Construtor da classe Linha
    public Linha(string ID, string IDProtecao) : base(ID)
    {
        this.IDProtecao = IDProtecao;
    }
    // Atualiza objeto do tipo Linha
    public override void Update(float dt)
    {
        // Verificando os estados de todas as variáveis utilizadas nas lógicas de
abertura e fechamento do disjuntor
        bool trip = (Simulador.FindObject(IDProtecao) as ProtLinha).TRIP;
        bool bq = (Simulador.FindObject(IDProtecao) as ProtLinha).BQ;
        // Testando a lógica de abertura do disjuntor
        if (disjuntor.Estado && (trip || bq))
        {
            disjuntor.Comando = Disjuntor.Comandos.Abrir;
        }
        // Testando a lógica de fechamento do disjuntor
        if (!disjuntor.Estado && !(trip || bq) && (!secTerra.Estado ||
secTerra.chFlexTestTerra.Estado) && (disjuntor.Comando ==
Disjuntor.Comandos.Fechar))
        {
            disjuntor.LogicaFechamento = true;
        }
        // Atualizando todos os objeto que compõem o tipo Linha
        secBarraA.Update(dt);
        secBarraB.Update(dt);
        secLinhaChegada.Update(dt);
        secLinhaSaida.Update(dt);
        secTerra.Update(dt);
        secBypass.Update(dt);
        disjuntor.Update(dt);
    }
}
```

J. CÓDIGO EM C# DA CLASSE SECTRAFOBARRAA.

```
public class SecTrafoBarraA : Seccionadora
{
    /// <summary>
    /// Declarando a variável que armazena o ID do transformador ao qual a
secccionadora pertence
    /// </summary>
    protected string IDTrafo;
    /// <summary>
    /// Declarando a variável que armazena o ID do conjunto de variáveis de
monitoramento
    /// de conexão à barra que são consultadas pelas lógicas de intertravamento
    /// </summary>
    protected string IDVariaveisBarra;
    /// <summary>
    /// Declarando a variável que armazena o ID da proteção do amarre, necessária
para verificar
    /// o estado de operação da proteção diferencial de barra (OFF, INDIVIDUAL,
OVERALL)
    /// </summary>
    protected string IDProtAmarre;
    /// <summary>
    /// Declarando o a variável de armazena o estado normal de conexão da
seccionadora (NA ou NF)
    /// </summary>
    private string IDLigacaoNormal;
    /// <summary>
    /// Construtor da classe CSecTrafoBarraA
    /// <summary>
    public SecTrafoBarraA(string ID, bool estado, string IDTrafo, string
IDLigacaoNormal, string IDProtAmarre, string IDVariaveisBarra)
        : base(ID, estado)
    {
        /// <summary>
        ///  Inicializando os atributos da classe
        /// </summary>
        this.IDTrafo = IDTrafo;
        this.IDVariaveisBarra = IDVariaveisBarra;
        this.IDProtAmarre = IDProtAmarre;
        this.IDLigacaoNormal = IDLigacaoNormal;
    }
    /// <summary>
    /// Atualiza objeto do tipo SecTrafoBarraA
    /// </summary>
    /// <param name="dt">Tempo desde a última atualização</param>
    public override void Update(float dt)
    {
        /// <summary>
        /// Verificando os estados de todas as variáveis utilizadas na lógica de
intertravamento
        /// </summary>
        bool secBarraB = (Simulador.FindObject(IDTrafo) as
Trafo).secBarraB.Estado;
        bool secBypass = (Simulador.FindObject(IDTrafo) as
Trafo).secBypass.Estado;
        bool disjuntor = (Simulador.FindObject(IDTrafo) as
Trafo).disjuntor.Estado;
        bool overall = (Simulador.FindObject(IDProtAmarre) as
ProtAmarre).ch29.Estado == Chave29.Estados.Overall ? true : false;
        bool BX = (Simulador.FindObject(IDVariaveisBarra) as VarBarra).BX;
        bool BPBX = (Simulador.FindObject(IDVariaveisBarra) as VarBarra).BPBX;
        bool BTAX = (Simulador.FindObject(IDVariaveisBarra) as VarBarra).BTAX;
        bool ch43NX = (Simulador.FindObject(IDVariaveisBarra) as
VarBarra).Ch43NX;
        /// <summary>
        /// Implementando a lógica de intertravamento
```

```
        /// </summary>
        bool teste1, teste2, teste3, saida = false;
        teste1 = (BX && !secBypass && !secBarraB && !disjuntor);
        teste2 = (BPBX && !secBarraB && !disjuntor);
        teste3 = (BPBX && ch43NX && BTAX && secBarraB);
        if (IDLigacaoNormal == ObjetoSimulador.ID_ESTADO_NORMAL_NF)
        {
            /// <summary>
            /// Lógica de intertravamento para as seccionadoras NF na barra A
            /// </summary>
            saida = (teste1 || teste2 || teste3);
        }
        else
        {
            /// <summary>
            /// Lógica de intertravamento para as seccionadoras NA na barra B
            /// </summary>
            saida = (overall && (teste1 || teste2 || teste3));
        }
        /// <summary>
        /// Verificando a manobra da chave, a depender da lógica de
intertravamento e da emissão de um comando
        /// </summary>
        if (saida && (comando != Comandos.Neutro))
        {
            estado = (comando == Comandos.Fechar) ? true : false;
        }
        comando = Comandos.Neutro;
    }
}
```

K. CÓDIGO EM C# DA CLASSE TRAFO.

```
public class Trafo : ObjetoSimulador
{
    /// <summary>
    /// Declarando os componentes de um elemento transformador
    /// </summary>
    public SecTrafoBarraA secBarraA;
    public SecTrafoBarraB secBarraB;
    public SecTrafoBypass secBypass;
    public SecTrafoDisjuntor secDisjuntorLadoBarra;
    public SecTrafoDisjuntor secDisjuntorLadoTrafo;
    public Disjuntor disjuntor;
    /// <summary>
    /// Declarando a variável que armazena o nome da barra
    /// </summary>
    private string IDBarra;

    /// <summary>
    /// Declarando a variável que armazena o nome da proteção da linha
    /// </summary>
    private string IDProtecao;

    public string ID_Protecao
    {
        get { return IDProtecao; }
    }
    /// <summary>
    /// Construtor da classe Trafo
    /// </summary>
    /// <param name="ID">ID do objeto</param>
    /// <param name="IDBarra">ID da barra</param>
    /// <param name="IDProtecao">ID da proteção do transformador</param>
    public Trafo(string ID, string IDBarra, string IDProtecao) : base(ID)
    {
        this.IDBarra = IDBarra;
        this.IDProtecao = IDProtecao;
    }
    public Trafo(string ID) : base(ID)
    {
        // TODO: Complete member initialization
        this.ID = ID;
    }
    /// <summary>
    /// Atualiza objeto do tipo Trafo
    /// </summary>
    /// <param name="dt">Tempo desde a última atualização</param>
    public override void Update(float dt)
    {
        /// <summary>
        /// Verificando os estados de todas as variáveis utilizadas nas
lógicas de abertura e fechamento do disjuntor
        /// </summary>
        bool trip = (Simulador.FindObject(IDProtecao) as ProtTrafo).TRIP;
        bool bq = (Simulador.FindObject(IDProtecao) as ProtTrafo).BQ;
        /// <summary>
        /// Testando a lógica de abertura do disjuntor
        /// </summary>
        if (disjuntor.Estado && (trip || bq))
        {
            disjuntor.Comando = Disjuntor.Comandos.Abrir;
        }

        /// <summary>
        /// Testando a lógica de fechamento do disjuntor
        /// </summary>
```

```
        if (!disjuntor.Estado && !(trip || bq) && (disjuntor.Comando ==
Disjuntor.Comandos.Fechar))
        {
            disjuntor.LogicaFechamento = true;
        }
        /// <summary>
        /// Atualizando todos os objeto que compõem o tipo Trafo
        /// </summary>
        secBarraA.Update(dt);
        secBarraB.Update(dt);
        secDisjuntorLadoBarra.Update(dt);
        secDisjuntorLadoTrafo.Update(dt);
        secBypass.Update(dt);
        disjuntor.Update(dt);
    }
}
```

L. CÓDIGO EM C# DA CLASSE PROTLINHA.

```
public class ProtLinha : ObjetoSimulador
{
    /// <summary>
    /// Declarando a variável que armazena o nome da linha protegida
    /// </summary>
    public string IDLinha;

    /// <summary>
    /// Declarando os relés de falha do disjuntor
    /// </summary>
    public Rele rele50BF;
    public Rele rele62BF;

    /// <summary>
    /// Declarando o relé de religamento de linha
    /// </summary>
    public Rele rele79;

    /// <summary>
    /// Declarando a chave de transferência da proteção
    /// </summary>
    public Chave ch43TD;

        /// <summary>
    /// Variáveis utilizadas na proteção da linha
    /// </summary>
    protected bool trip;
    protected bool bq;
    protected bool bfa;
    protected bool bfb;
    protected bool tbfa;
    protected bool tbfb;

        /// <summary>
    /// Retorna o estado do trip da proteção
    /// </summary>
    public bool TRIP
    {
        get { return trip; }
    }

    /// <summary>
    /// Retorna o estado do bloqueio da proteção
    /// </summary>
    public bool BQ
    {
        get { return bq; }
    }

    /// <summary>
    /// Retorna o estado do bloqueio para falha de disjuntor na barra A
    /// </summary>
    public bool BFA
    {
        get { return bfa; }
    }

    /// <summary>
    /// Retorna o estado do bloqueio para falha de disjuntor na barra B
    /// </summary>
    public bool BFB
    {
        get { return bfb; }
    }
```

```
        /// <summary>
        /// Retorna o estado trip do bloqueio para falha de disjuntor na barra A
        /// </summary>
        public bool TBFA
        {
            get { return tbfa; }
        }

        /// <summary>
        /// Retorna o estado trip do bloqueio para falha de disjuntor na barra B
        /// </summary>
        public bool TBFB
        {
            get { return tbfb; }
        }

       /// <summary>
        /// Inicializa objeto do tipo ProtLinha
        /// </summary>
        /// <param name="ID">ID do objeto</param>
        public ProtLinha(string ID) : base(ID)
        {
        }
}
```

M. CÓDIGO EM C# DA CLASSE PROTLT1_345KV.

```
public class ProtLT1_345kV : ProtLinha
{
    /// <summary>
    /// Declarando os relés
    /// </summary>
    public Rele rele21P;
    public Rele rele21A;

    /// <summary>
    /// Declarando as variáveis de trip das proteções primária e secundária
    /// </summary>
    private bool tripP;
    private bool tripA;

    /// <summary>
    /// Retorna o estado do trip da proteção primária
    /// </summary>
    public bool TripP
    {
        get { return tripP; }
    }

    /// <summary>
    /// Retorna o estado do trip da proteção secundária
    /// </summary>
    public bool TripA
    {
        get { return tripA; }
    }

    /// <summary>
    /// Construtor da classe ProtLT1_345kV
    /// </summary>
    /// <param name="ID">ID da proteção da linha</param>
    /// <param name="IDLinha">ID da linha protegida</param>
    public ProtLT1_345kV(string ID, string IDLinha) : base(ID)
    {
        this.IDLinha = IDLinha;
    }

    /// <summary>
    /// Atualiza objeto do tipo ProtLT1_345kV
    /// </summary>
    /// <param name="dt">Tempo desde a última atualização</param>
    public override void Update(float dt)
    {
        /// <summary>
        /// Atualizando o estado da chave 43TD
        /// </summary>
        ch43TD.Update(dt);

        /// <summary>
        /// Verificando o estado das variáveis utilizadas na lógica da
proteção da LT1_345kV
        /// </summary>
        bool releA86 = (Simulador.FindObject("ProtAmarre345kV") as
ProtAmarre345kV).releA86.Estado;
        bool contatoReleA86 = (Simulador.FindObject("ProtAmarre345kV") as
ProtAmarre345kV).releA86.Contato[1];
        bool releA86BF = (Simulador.FindObject("ProtAmarre345kV") as
ProtAmarre345kV).releA86BF.Estado;
        bool contatoReleA86BF = (Simulador.FindObject("ProtAmarre345kV") as
ProtAmarre345kV).releA86BF.Contato[1];
        bool releB86BF = (Simulador.FindObject("ProtAmarre345kV") as
ProtAmarre345kV).releB86BF.Estado;
```

```
        bool contatoReleB86BF = (Simulador.FindObject("ProtAmarre345kV") as
ProtAmarre345kV).releB86BF.Contato[1];
        bool secBarraA = (Simulador.FindObject(IDLinha) as
Linha).secBarraA.Estado;
        bool secBarraB = (Simulador.FindObject(IDLinha) as
Linha).secBarraB.Estado;

        /// <summary>
        /// Verificando a lógica de trip das proteções primária e secundária
        /// </summary>
        tripP = rele21P.Estado;
        tripA = rele21A.Estado;

        /// <summary>
        /// Verificando o estado dos relés relacionados com a proteção
        /// </summary>
        if (tripP || tripA)
        {
            /// <summary>
            /// Setando o estado do trip da proteção da LT1_345kV e do relé
50BF de falha do disjuntor
            /// </summary>
            trip = true;
            rele50BF.Estado = true;
        }
        else
        {
            /// <summary>
            /// Setando as variáveis como false, caso não haja atuação dos
relés 21P, 21S e 67N
            /// </summary>
            trip = false;
            rele50BF.Estado = true;
        }

        /// <summary>
        /// Implementando a lógica da proteção de falha do disjuntor na barra
A
        /// </summary>
        if (((rele50BF.Estado && rele62BF.Estado) && secBarraA)
            || ((secBarraA || secBarraB) && ch43TD.Estado && (rele50BF.Estado
&& rele62BF.Estado)))
        {
            bfa = true;
        }
        else
        {
            bfa = false;
        }

        /// <summary>
        /// Implementando a lógica da proteção de falha do disjuntor na barra
B
        /// </summary>
        if (((rele50BF.Estado && rele62BF.Estado) && secBarraB)
            || ((secBarraA || secBarraB) && ch43TD.Estado && (rele50BF.Estado
&& rele62BF.Estado)))
        {
            bfb = true;
        }
        else
        {
            bfb = false;
        }

        /// <summary>
        /// Implementando a lógica do trip da proteção de falha do disjuntor
na barra A
```

```
        /// </summary>
        if (secBarraA && (releA86BF && contatoReleA86BF))
        {
            tbfa = true;
        }
        else
        {
            tbfa = false;
        }

        /// <summary>
        /// Implementando a lógica do trip da proteção de falha do disjuntor
na barra B
        /// </summary>
        if (secBarraB && (releB86BF && contatoReleB86BF))
        {
            tbfb = true;
        }
        else
        {
            tbfb = false;
        }

        /// <summary>
        /// Implementando a lógica de bloqueio da proteção
        /// </summary>
        if (tbfa || tbfb || (releA86 && contatoReleA86))
        {
            bq = true;
        }
        else
        {
            bq = false;
        }

        /// <summary>
        /// Implementando a lógica de falha do disjuntor
        /// </summary>
        if (trip || bq)
        {
            /// <summary>
            /// Verificando a falha do disjuntor de linha
            /// </summary>
            if ((Simulador.FindObject(IDLinha) as Linha).disjuntor.Falha)
            {
                /// <summary>
                /// Setando o estado do relé 62BF como true, o que confirma a
falha do disjuntor de linha
                /// </summary>
                rele62BF.Estado = true;
            }
            else
            {
                /// <summary>
                /// Setando o estado do relé 62BF como false, o que confirma o
bom funcionamento do disjuntor de linha
                /// </summary>
                rele62BF.Estado = false;
            }
        }
    }
}
```

N. CÓDIGO EM C# DA CLASSE PROTAMARRE.

```
public class ProtAmarre : ObjetoSimulador
{
    /// <summary>
    /// Relés de bloqueio de falha do disjuntor
    /// </summary>
    public ReleBloqueio releA86BF;
    public ReleBloqueio releB86BF;

    /// <summary>
    /// Relés de bloqueio da proteção diferencial de barra
    /// </summary>
    public ReleBloqueio releA86;
    public ReleBloqueio releB86;

    /// <summary>
    /// Relés diferenciais da proteção da barra A
    /// </summary>
    public ReleDiferencial releA87C_A;
    public ReleDiferencial releB87B_A;
    public ReleDiferencial releC87A_A;

    /// <summary>
    /// Relés diferenciais da proteção da barra B
    /// </summary>
    public ReleDiferencial releA87C_B;
    public ReleDiferencial releB87B_B;
    public ReleDiferencial releC87A_B;

    /// <summary>
    /// Chave de seleção do modo de operação da proteção diferencial de barra,
    /// que pode assumir os valores: Off, Individual, Overall.
    /// </summary>
    public Chave29 ch29;

    /// <summary>
    /// Chave de seleção da barra de supervisão quando da transferência da
    /// proteção para o disjuntor de amarre
    /// </summary>
    public Chave43IB ch43IB;

    /// <summary>
    /// Variáveis utilizadas na proteção do disjuntor de amarre
    /// </summary>
    protected bool trip;
    protected bool bq;
    protected bool bfa;
    protected bool bfb;

    /// <summary>
    /// Retorna o estado do trip da proteção do disjuntor de amarre
    /// </summary>
    public bool TRIP
    {
        get { return trip; }
    }

    /// <summary>
    /// Retorna o estado do bloqueio da proteção do disjuntor de amarre
    /// </summary>
    public bool BQ
    {
        get { return bq; }
    }

    /// <summary>
```

```
        /// Retorna o estado do bloqueio para falha de disjuntor na barra A
        /// </summary>
        public bool BFA
        {
            get { return bfa; }
        }

        /// <summary>
        /// Retorna o estado do bloqueio para falha de disjuntor na barra B
        /// </summary>
        public bool BFB
        {
            get { return bfb; }
        }

        /// <summary>
        /// Construtor da classe ProtAmarre
        /// </summary>
        /// <param name="ID">ID do objeto</param>
        public ProtAmarre(string ID) : base (ID)
        {
        }
}
```

O. CÓDIGO EM C# DA CLASSE PROTAMARRE345KV.

```
public class ProtAmarre345kV : ProtAmarre
{

    /// <summary>
    /// Declarando as variáveis de trip das proteções diferenciais
    /// </summary>
    private bool trip87BarraA;
    private bool trip87BarraB;

    /// <summary>
    /// Retorna o estado do trip da proteção diferencial
    /// </summary>
    public bool Trip87BarraA
    {
       get { return trip87BarraA; }
    }

    /// <summary>
    /// Retorna o estado do trip da proteção diferencial
    /// </summary>
    public bool Trip87BarraB
    {
      get { return trip87BarraB; }
    }


    /// <summary>
    /// Construtor da classe ProtAmarre345kV
    /// </summary>
    /// <param name="ID">ID do objeto</param>
    public ProtAmarre345kV(string ID) : base(ID)
    {
    }

    /// <summary>
    /// Atualiza objeto do tipo ProtAmarre345kV
    /// </summary>
    /// <param name="dt">Tempo desde a última atualização</param>
    public override void Update(float dt)
    {
        /// <summary>
        /// Atualizando os estados das chaves
        /// </summary>
        ch29.Update(dt);
        ch43IB.Update(dt);

        /// <summary>
        /// Implementando a lógica da proteção diferencial da barra de 345 kV
        /// </summary>
        trip87BarraA = (releA87C_A.Estado || releB87B_A.Estado ||
releC87A_A.Estado);
        trip87BarraB = (releA87C_B.Estado || releB87B_B.Estado ||
releC87A_B.Estado);

        if (trip87BarraA || ((ch29.Estado == Chave29.Estados.Overall) &&
trip87BarraB))
        {
            releA86.Estado = true;
        }
        if (trip87BarraB || ((ch29.Estado == Chave29.Estados.Overall) &&
trip87BarraA))
        {
            releB86.Estado = true;
        }
```

```
        /// <summary>
        /// Implementando a lógica da proteção de falha do disjuntor da barra
de 345 kV
        /// </summary>
        bfa =      (Simulador.FindObject("ProtLT1_345kV") as ProtLinha).BFA
                || (Simulador.FindObject("ProtLT2_345kV") as ProtLinha).BFA
                || (Simulador.FindObject("ProtAT01_345kV") as ProtTrafo).BFA
                || (Simulador.FindObject("ProtAT02_345kV") as ProtTrafo).BFA;
//para adicionar nova linha, usa ||

        bfb =      (Simulador.FindObject("ProtLT1_345kV") as ProtLinha).BFB
                || (Simulador.FindObject("ProtLT2_345kV") as ProtLinha).BFB
                || (Simulador.FindObject("ProtAT01_345kV") as ProtTrafo).BFB
                || (Simulador.FindObject("ProtAT02_345kV") as ProtTrafo).BFB;
//para adicionar nova linha, usa ||

        releA86BF.Estado = !releA86BF.Estado ? bfa : releA86BF.Estado;
        releA86BF.Estado = !releA86BF.Estado ? bfa : releA86BF.Estado;
        releB86BF.Estado = !releB86BF.Estado ? bfb : releB86BF.Estado;
        releB86BF.Estado = !releB86BF.Estado ? bfb : releB86BF.Estado;

        /// <summary>
        /// Implementando a lógica de bloqueio do disjuntor de amarre da barra
de 345 kV
        /// </summary>
        bool bq1 =     (releA86BF.Estado && releA86BF.Contato[0])
                    || (releB86BF.Estado && releA86BF.Contato[0])
                    || (releA86.Estado && releA86.Contato[0])
                    || (releB86.Estado && releB86.Contato[0]);

        bool bq2 =     ((Simulador.FindObject("ProtLT1_345kV") as ProtLinha).BQ
&& (Simulador.FindObject("ProtLT1_345kV") as ProtLinha).ch43TD.Estado)
                    || ((Simulador.FindObject("ProtLT2_345kV") as ProtLinha).BQ
&& (Simulador.FindObject("ProtLT2_345kV") as ProtLinha).ch43TD.Estado)
                    || ((Simulador.FindObject("ProtAT01_345kV") as
ProtTrafo).BQ && (Simulador.FindObject("ProtAT01_345kV") as
ProtTrafo).ch43TD.Estado)
                    || ((Simulador.FindObject("ProtAT02_345kV") as
ProtTrafo).BQ && (Simulador.FindObject("ProtAT02_345kV") as
ProtTrafo).ch43TD.Estado); //para adicionar nova linha, usa ||

        bq = (bq1 || bq2);

        /// <summary>
        /// Implementando a lógica de trip do disjuntor de amarre da barra de
345 kV
        /// </summary>
        trip = bq || ((Simulador.FindObject("ProtLT1_345kV") as
ProtLinha).TRIP && (Simulador.FindObject("ProtLT1_345kV") as
ProtLinha).ch43TD.Estado)
                  || ((Simulador.FindObject("ProtLT2_345kV") as
ProtLinha).TRIP && (Simulador.FindObject("ProtLT2_345kV") as
ProtLinha).ch43TD.Estado)
                  || ((Simulador.FindObject("ProtAT01_345kV") as
ProtTrafo).TRIP && (Simulador.FindObject("ProtAT01_345kV") as
ProtTrafo).ch43TD.Estado)
                  || ((Simulador.FindObject("ProtAT02_345kV") as
ProtTrafo).TRIP && (Simulador.FindObject("ProtAT02_345kV") as
ProtTrafo).ch43TD.Estado); //para adicionar nova linha, usa ||
    }
}
```

P. CÓDIGO EM C# DA CLASSE PROTTRAFO.

```
public class ProtTrafo : ObjetoSimulador
{
    /// <summary>
    /// Declarando a variável que armazena o nome do trafo protegida
    /// </summary>
    public string IDTrafo;
    /// <summary>
    /// Declarando os relés de falha do disjuntor
    /// </summary>
    public Rele rele50BF;
    public Rele rele62BF;
    /// <summary>
    /// Declarando a chave de transferência da proteção
    /// </summary>
    public Chave ch43TD;
    /// <summary>
    /// Variáveis utilizadas na proteção do transformador
    /// </summary>
    protected bool trip;
    protected bool bq;
    protected bool bfa;
    protected bool bfb;
    protected bool tbfa;
    protected bool tbfb;
    /// <summary>
    /// Retorna o estado do trip da proteção
    /// </summary>
    public bool TRIP
    {
        get { return trip; }
    }
    /// <summary>
    /// Retorna o estado do bloqueio da proteção
    /// </summary>
    public bool BQ
    {
        get { return bq; }
    }
    /// <summary>
    /// Retorna o estado do bloqueio para falha de disjuntor na barra A
    /// </summary>
    public bool BFA
    {
        get { return bfa; }
    }
    /// <summary>
    /// Retorna o estado do bloqueio para falha de disjuntor na barra B
    /// </summary>
    public bool BFB
    {
        get { return bfb; }
    }
    /// <summary>
    /// Retorna o estado trip do bloqueio para falha de disjuntor na barra A
    /// </summary>
    public bool TBFA
    {
        get { return tbfa; }
    }
    /// <summary>
    /// Retorna o estado trip do bloqueio para falha de disjuntor na barra B
    /// </summary>
    public bool TBFB
    {
        get { return tbfb; }
```

```
    }
    /// <summary>
    /// Inicializa objeto do tipo ProtTrafo
    /// </summary>
    /// <param name="ID">ID do objeto</param>
    public ProtTrafo(string ID): base(ID)
    {
    }
}
```

Q. CÓDIGO EM C# DA CLASSE PROTREATOR.

```
public class ProtReator : ObjetoSimulador
{
    /// <summary>
    /// Variáveis utilizadas na proteção do reator
    /// </summary>
    protected bool trip;
    protected bool bq;
    protected bool bfa;
    protected bool bfb;
    protected bool tbfa;
    protected bool tbfb;
    /// <summary>
    /// Retorna o estado do trip da proteção
    /// </summary>
    public bool TRIP
    {
        get { return trip; }
    }
    /// <summary>
    /// Retorna o estado do bloqueio da proteção
    /// </summary>
    public bool BQ
    {
        get { return bq; }
    }
    /// <summary>
    /// Retorna o estado do bloqueio para falha de disjuntor na barra A
    /// </summary>
    public bool BFA
    {
        get { return bfa; }
    }
    /// <summary>
    /// Retorna o estado do bloqueio para falha de disjuntor na barra B
    /// </summary>
    public bool BFB
    {
        get { return bfb; }
    }
    /// <summary>
    /// Retorna o estado trip do bloqueio para falha de disjuntor na barra A
    /// </summary>
    public bool TBFA
    {
        get { return tbfa; }
    }
    /// <summary>
    /// Retorna o estado trip do bloqueio para falha de disjuntor na barra B
    /// </summary>
    public bool TBFB
    {
        get { return tbfb; }
    }
    /// <summary>
    /// Inicializa objeto do tipo ProtReator
    /// </summary>
    /// <param name="ID">ID do objeto</param>
    public ProtReator(string ID) : base(ID)
    {
    }
}
```