Ministério da Educação
Universidade Federal do Pampa
Campus Alegrete

TESTE DE SOFTWARE APLICADO A FERRAMENTA ASE

ÂNGELA SANTINON TOSATTO

MONOGRAFIA DE CONCLUSÃO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO

Orientador(a): Me. Vanessa Gindri Vieira

Alegrete, 07 de dezembro de 2010.

ÂNGELA SANTINON TOSATTO

TESTE DE SOFTWARE APLICADO A FERRAMENTA ASE

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal do Pampa, como requisito parcial para a obtenção do Título de Bacharel em Ciência da Computação.

Área de concentração: Ciências Exatas e da Terra.

Profª orientadora: Me. Vanessa Gindri Vieira

Alegrete, 2010.

ÂNGELA SANTINON TOSATTO

TESTE DE SOFTWARE APLICADO A FERRAMENTA ASE

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal do Pampa, como requisito parcial para a obtenção do Título de Bacharel em Ciência da Computação.

Área de concentração: Ciências Exatas e da Terra.

Monografia defendida e aprovada em: 13 de dezembro de 2010.
Banca Examinadora:

Me. Vanessa Gindri Vieira
Orientadora
Ciência da Computação - UNIPAMPA

Dra. Amanda Meincke Melo
Ciência da Computação - UNIPAMPA

Dr. Cleo Zanella Billa
Ciência da Computação - UNIPAMPA

“Aos meus queridos pais”.

AGRADECIMENTOS

Agradeço a Deus pelas oportunidades que me estão sendo dadas na vida, principalmente por ter conhecido muitas pessoas e lugares interessantes, mas também por ter me dado força nas fases difíceis, que foram matérias-primas de aprendizado.

Não posso deixar de agradecer aos meus pais Juarez e Neiva, por terem me fornecido as condições necessárias para eu conseguir concluir este curso. Obrigada por toda a dedicação e amor durante todos esses anos.

Aos meus irmãos Bruno e Daniel, pela verdadeira amizade que nos une desde pequenos, pelas palavras quando estamos juntos, quando a distância nos separa e quando a saudade é grande.

Ao meu namorado Matheus, por toda a compreensão e carinho.

Ao meu amigo Wagner Reck, que me forneceu bons momentos de discussão sobre o tema abordado, e material de pesquisa.

A minha orientadora Vanessa Gindri Vieira, por toda a ajuda e compreensão nas horas difíceis e pelas boas conversas que tivemos.

Aos demais professores, agradeço pelos tijolinhos que contribuíram para a conclusão deste curso.

Aos meus amigos, pela verdadeira amizade que construímos e por estarem sempre ao meu lado durante essa trajetória.

Agradeço também aos familiares que me deram muito apoio e carinho.

A persistência é o caminho do êxito.

Charles Chaplin

RESUMO

A produção de software de alta qualidade propicia aos usuários uma maior confiança e segurança na utilização do mesmo. Para alcançar a qualidade desejada existe um processo inserido nas fases de desenvolvimento de software chamado teste. A atividade de teste é contínua para avaliar e mensurar a qualidade do trabalho desenvolvido desde a análise de requisitos até a fase de manutenção do software. O teste exercita o programa a fim de localizar os erros para serem corrigidos e posteriormente eliminados. Neste trabalho é apresentada uma estratégia de teste de software, com o intuito de colaborar para a melhoria da qualidade da ferramenta Análise de Sistemas Elétricos, desenvolvida para alocar chaves telecomandas, por sistemas computacionais, em redes de distribuição de energia elétrica. A estratégia adotada para a realização dos casos de teste consiste em combinar as técnicas de testes caixa-preta e caixa-branca, estes últimos exercitam a estrutura interna do sistema enquanto os testes caixa-preta exercitam as funcionalidades, não se preocupando com a maneira utilizada para construção da ferramenta. Ao finalizar esse processo de teste é possível mostrar que foram encontradas falhas de software e isso contribui com a qualidade desse produto, pois depois de encontradas as falhas se faz necessário realizar suas correções e enviá-las em forma de atualização para o cliente.

Palavras-chave: estratégia de teste, testes funcionais, testes unitários, qualidade de software.

ABSTRACT

The high quality software production gives the user a greater confidence and security levels. In order to achieve the desired level of quality, there is a process included in the software development stages called software testing. Software testing is a continuous activity because evaluates and measures the quality of work from requirements analysis to software maintenance stage. Software testing exercises the computer program in order to find the errors to be corrected, and then eliminated. This work presents a software testing strategy for improving the quality of the Electrical Systems Analysis tool. This tool was developed to allocate remotes key, which are controlled by computer systems, in networks of distribution of electricity. The strategy adopted to achieve the test cases is to combine the techniques of black-box testing and white-box testing. The white-box testing exercise the internal structure of the system, while the black-box testing exercise the functionality of the system without worrying about the way used for the construction of the tool. At the end this process of testing is possible to show that were found failures of software. This event has contributed to the quality of the software product, because after found failings, the corrections are made and sent as updates to the client.

Keywords: test strategy, functional testing, unit testing, software quality.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ASE: Análise de Sistemas Elétricos

DEC: Duração Equivalente por consumidor

DIC: Duração Individual por Consumidor

DMIC: Duração Máxima Individual por consumidor

ENS: Energia Não Suprida

FEC: Frequência Equivalente por Consumidor

FIC: Frequência Individual por Consumidor

GESEP: Grupo de Estudos de Sistemas Elétricos de Potência

NA: Chave Normalmente Aberta

NF: Chave Normalmente Fechada

OO: Orientação a Objetos

PN: Pontos Notáveis

V&V: Verificação e Validação

SUMÁRIO

INTRODUÇÃO

Em consequência do aumento da utilização de sistemas de computadores nas mais variadas áreas da atividade humana, há a necessidade de garantir a qualidade dos sistemas computacionais que serão utilizados. Sendo assim, a Engenharia de Software é uma aliada dos desenvolvedores, pois nas últimas décadas, tem evoluído significativamente estabelecendo novas técnicas, ferramentas, critérios e métodos para auxiliar nas diversas fases da produção de software. A produção de software com alta qualidade a baixo custo é o objetivo principal que esta evolução possibilitou (PFLEEGER, 2004).

A qualidade de um software está relacionada com o cumprimento de requisitos funcionais e de desempenho explicitamente declarados, normas de desenvolvimento explicitamente documentadas e características implícitas que são esperadas de todo software desenvolvido profissionalmente (PRESSMAN, 2006).

Segundo a ABNT (2000), "qualidade é o grau no qual um conjunto de características inerentes satisfaz aos requisitos". Ou seja, pode-se afirmar que se algum produto ou serviço atende aos requisitos especificados, este possui a qualidade desejada.

Uma definição mais abrangente e completa para qualidade de software foi proposta por Bartié: "Qualidade de software é um processo sistemático que focaliza todas as etapas e artefatos produzidos com o objetivo de garantir a conformidade de processos e produtos, prevenindo e eliminando defeitos". (BARTIÉ, 2002).

O teste de software pode ser visto como uma parcela do processo de qualidade de software. A qualidade da aplicação pode, e normalmente, varia significativamente de sistema para sistema. Os atributos qualitativos previstos na norma ABNT (2003) são:

- funcionalidade: satisfação das necessidades;
- confiabilidade: imunidade a falhas;

- usabilidade: facilidade de uso;
- eficiência: rápido e "enxuto";
- manutenibilidade: facilidade de manutenção;
- portabilidade: uso em outros ambientes.

De forma geral, mensurar o bom funcionamento de um software envolve compará-lo com elementos como especificações, outros software da mesma linha, versões anteriores do mesmo produto, inferências pessoais, expectativas do cliente, normas relevantes, leis aplicáveis, entre outros (SOMMERVILLE, 2007).

Apesar do uso de novas tecnologias para melhorar a qualidade, os produtos de software ainda podem conter erros. Todas as fases do ciclo de vida de um software podem introduzir erros, mas Bartié (2002) afirma que é nas fases iniciais que ocorrem com maior frequência devido à má especificação e entendimento dos requisitos do sistema, conforme Figura 1 (BARTIÉ, 2002). Se o objetivo é reduzir o risco de erros, então é necessário um maior cuidado na execução das fases iniciais de desenvolvimento para que os erros não migrem para outras fases.

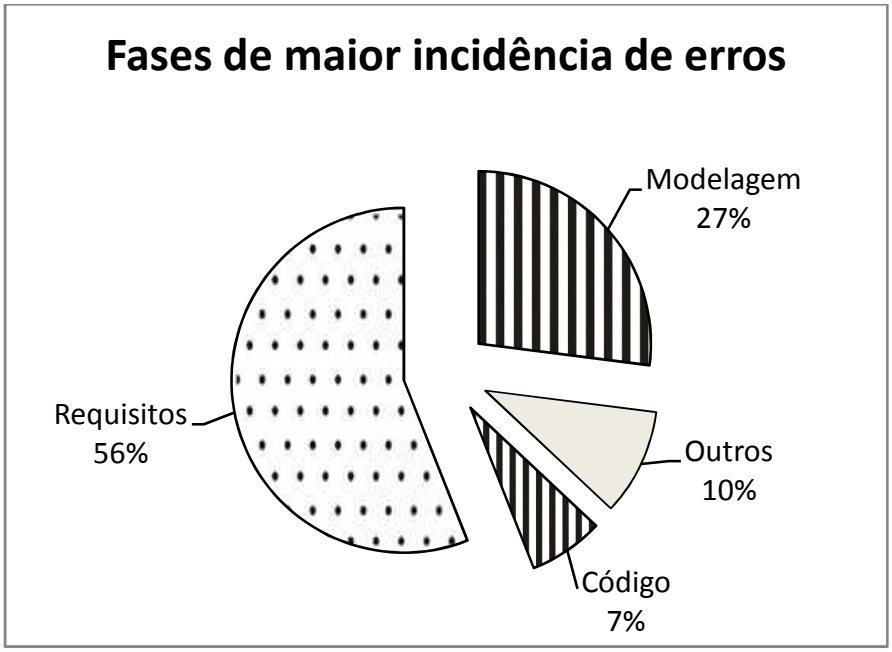

Figura 1: Fases de maior incidência de erros

Existem formas para diminuir a ocorrência de erros e riscos associados. A agregação de atividades conhecidas como Garantia de Qualidade de Software é necessária em todas as fases do processo de desenvolvimento de software, dentre estas atividades estão atividades de Veri-

ficação e Validação - V&V. Verificação se refere ao conjunto de atividades que garante que o software implementa corretamente uma função específica e validação se refere a um conjunto de atividades diferentes que garante que o software construído corresponde aos requisitos do cliente (PRESSMAN, 2006). Uma das mais utilizadas atividade de V&V é o teste, constituindo-se em um dos elementos para fornecer evidências da confiabilidade do software. Através dos testes são descobertos os erros que foram cometidos inadvertidamente durante o projeto e/ou construção do software.

A atividade de teste de software possui definições que merecem atenção na sua conceituação, a diferença entre erro, defeito e falha. Defeito (*Fault*) faz parte do universo físico, ou seja, sempre é cometido por um indivíduo e caracteriza-se por uso incorreto de processo ou sintaxe. Erro (*Error*) se dá no universo da informação e é conseqüência de um defeito em um artefato de software, é caracterizado pela diferença do valor processado e o valor esperado, portanto, todo valor que seja diferente do esperado é classificado como erro. Falha (*Failure*) só é possível perceber no universo do usuário quando o comportamento da aplicação é diferente dos requisitos relatados pelo cliente e é gerada a partir de um ou vários erros (GOETTENAUER, 2009).

Para cada paradigma de desenvolvimento de software existem técnicas de teste que abordam suas características específicas para que os erros sejam descobertos e não prejudiquem a qualidade final do software. Na abordagem de desenvolvimento de software orientada a objeto (OO), segundo Martins e Tchanner (2009) a etapa de testes é mais complexa, se comparada com softwares estruturados, devido aos tipos de relacionamento que existem, tais como: herança, encapsulamento, associação, criação dinâmica de objetos, polimorfismo, etc. Além disso, deve ser considerado que programas OO não têm uma sequência pré-definida para a execução de suas tarefas, e ainda podem existir múltiplas tarefas sendo executadas paralelamente na mesma aplicação, aumentando ainda mais a complexidade nos testes para estes sistemas (MARTINS e TCHANNER, 2009).

Para a realização de testes de sistemas OO são propostas diversas estratégias para cobrir as particularidades deste paradigma, por exemplo, hereditariedade, encapsulamento, abstração, polimorfismo, etc (PÈZZE, 2008).

A fim de verificar a eficácia da combinação de algumas técnicas de teste para software OO, é utilizado um sistema de software desenvolvido em linguagem orientada a objetos, C++, como estudo de caso.

O objetivo deste trabalho é mostrar a importância de criar os casos de teste e deixá-los documentados mesmo depois do sistema já ter sido utilizado pelo cliente, assim pode propiciar a descoberta de erros que serão tratados e devolvidos em forma de atualizações do sistema. A aplicação de testes em software implantado busca melhorar a qualidade do produto, uma vez que os defeitos descobertos são corrigidos e validados, causando o aumento do nível de credibilidade e segurança (Bartié, 2002).

Os próximos capítulos deste documento estão organizados como segue: no capítulo 2, é apresentada a revisão bibliográfica, ou seja, a comparação do que tem sido feito para realizar em testes de sistemas orientados a objetos e o que será proposto para realizar os testes neste trabalho; no capítulo 3, é definido o que é um teste para sistemas orientados a objetos, os tipos de teste que podem ser construídos neste paradigma e as estratégias que podem ser utilizadas para a realização destes testes.

O capítulo 3 contém o contexto em que a ferramenta, selecionada como estudo de caso, está inserida e como foi construída, as etapas desenvolvidas para obter o produto final e as funcionalidades que foram levadas em consideração para o planejamento dos testes.

No capítulo 4 é apresentado o planejamento dos casos de teste, ou seja, a elaboração, a maneira como foram executados e os resultados obtidos através da execução dos casos de teste. Para finalizar, as considerações finais oriundas dos resultados deste trabalho.

1 REVISÃO BIBLIOGRÁFICA

Alguns trabalhos relacionados foram encontrados em busca de uma solução para determinar qual a melhor estratégia de teste para aplicar em um software orientado a objeto. O objetivo dessas abordagens é encontrar falhas em um software como garantia de sua qualidade.

O trabalho de Bruneli (BRUNELI, 2006) destaca que em um processo de teste de software a atividade de teste deve começar tão logo a atividade de desenvolvimento se inicia e caminhar em paralelo com o ciclo tradicional de desenvolvimento. A abordagem por ele utilizada permite detectar e prevenir erros através do processo de desenvolvimento, conduzindo para uma maior confiança e qualidade do software. Concordando com a estratégia de Bruneli (2006), Marinho (2009) diz que a qualidade final do software está associada a um bom projeto de testes, este feito em paralelo com o projeto do sistema a ser desenvolvido. Marinho (2009) contribuiu com a qualidade final do sistema, pois as falhas encontradas eram mesmo oriundas do mau planejamento do projeto do sistema e não houve revisão quando ocorreram algumas mudanças de requisitos do projeto. Fernandes e Lopes (2007) também defendem a ideia de um planejamento integrado de codificação e testes, em que uma classe deve ser testada logo após sua criação, tal planejamento integrado ajuda a reduzir o tempo gasto para a realização dessas atividades e isso permitirá um melhor reaproveitamento de esforços na realização dos retestes.

Essas três abordagens, descritas acima, são utilizadas para softwares que estão sendo planejados, nesse trabalho não seria viável utilizar estas abordagens, pois o sistema utilizado como estudo de caso está implantado e não seria possível começar a testá-lo desde o levantamento de requisitos, primeira fase na confecção de software, onde são levantadas as funcionalidades que o software deve desempenhar.

Zimmermann (2006) destaca que à atividade de teste de software, por ser um processo complexo, deve ser destinado um bom tempo do projeto. Para minimizar esse tempo e conse-

quentemente os custos do software, surgem no mercado ferramentas de automação de testes. No presente trabalho, não foram utilizadas ferramentas para automatizar testes caixa-preta, mas essa estratégia de teste foi utilizada com o mesmo intuito de Zimmermann (2006), localizar erros funcionais utilizando a interface gráfica do usuário da mesma forma que um usuário final faria, por exemplo.

Molinari (2003) destaca que quando se realiza um teste funcional se está, na verdade, confrontando com o que se espera que o sistema vá fazer, ou seja, incluindo entrada de dados, processamento e resposta. Essa abordagem de Molinari (2003) é semelhante ao que foi realizado nesse trabalho para a análise após a execução dos casos de teste, o resultado esperado foi confrontado com o obtido e a partir disso foi verificado se houve ou não falha ao simular cada funcionalidade da ASE.

Para a produção de um software de alta qualidade é necessário a avaliação de um sistema e isso demanda um planejamento de testes, Primão et al. (2010) destaca que nessa fase é definido os tipos de testes que serão aplicados e a importância de desenvolver um bom planejamento dos testes garantindo que todos os seus requisitos foram devidamente testados levando em consideração tempo e custo para desenvolver esta atividade. Assim como Primão et al. (2010) na fase de planejamento de testes desse trabalho, foram definidos os tipos de testes a serem aplicados, testes funcionais e unitários, e também nessa fase foram criados os casos de teste para depois serem executados levando em consideração o tempo limite para findar esta atividade.

2 TESTES DE SOFTWARE ORIENTADO A OBJETO

Segundo Bartié (2002), os testes “têm por objetivo identificar o maior número possível de erros tanto nos componentes isolados quanto na solução tecnológica como um todo”.

Teste de software é o processo de execução de um produto para determinar se ele atingiu suas especificações e funcionou corretamente no ambiente para o qual foi projetado. O seu objetivo é revelar falhas em um produto, para que as causas dessas falhas sejam identificadas e possam ser corrigidas pela equipe de desenvolvimento.

De uma forma simples, testar um software significa verificar através de uma execução controlada se o seu comportamento está de acordo com o especificado. O objetivo principal desta tarefa é revelar o número máximo de falhas dispondo do mínimo de esforço, ou seja, mostrar a equipe de desenvolvimento se os resultados estão ou não de acordo com os padrões estabelecidos nos requisitos do sistema sob teste.

Os softwares orientados a objetos possuem algumas características particulares que impactam na elaboração dos testes. Pèzze (2008) destaca as seguintes:

- Comportamento dependente do estado: deve ser considerado o estado a partir do qual os métodos são ativados senão não há suficiência para revelar as falhas dependentes de estado;
- Encapsulamento: devido ao encapsulamento os oráculos de teste podem necessitar de acesso a essa informação encapsulada para distinguir entre o comportamento correto ou incorreto;
- Herança: deve ser considerado o efeito dos novos métodos e métodos sobrescritos sobre o comportamento dos métodos herdados e distinguir entre métodos que necessitam de novos casos de teste, métodos que não precisam ser testados nova-

mente e métodos ancestrais que podem ser testados novamente por outra execução dos casos de teste já existentes;

- Polimorfismo e ligação dinâmica: os testes devem exercitar diferentes ligações para revelar as falhas que dependem de ligações particulares ou de interações entre ligações de diferentes chamadas;
- Classes abstratas: esse tipo de classe não pode ser instanciada e testada diretamente, mesmo sendo componentes de interface importantes em bibliotecas e componentes. As classes abstratas são testadas sem o pleno conhecimento de como podem ser instanciadas;
- Tratamento de exceções: devido a distância textual de onde a exceção é lançada e onde é tratada, torna-se importante testar explicitamente as exceções e o fluxo de controle normal;
- Concorrência: introduz novos tipos de possibilidade de falhas, pois ao trabalhar com múltiplas *threads* de controle podem causar *deadlocks* e condições de corrida que tornam o comportamento do sistema dependente de decisões do escalonador que não está sob o controle do testador.

Existem diversas técnicas ou maneiras para testar software, todas estas têm um só objetivo: encontrar falhas no software; as mais conhecidas e utilizadas são as técnicas: testes caixa-preta, testes caixa-branca e testes de regressão. Os testes caixa-preta, embora não trabalhem diretamente com o código-fonte, são importantes, pois exercitam as funcionalidades do sistema.

No teste caixa-preta, conforme Figura 2 (DIAS-NETO, 2008), são fornecidos os dados de entrada, depois o teste é executado e o resultado obtido é comparado a um resultado esperado previamente conhecido. Haverá sucesso no teste se o resultado obtido for igual ao resultado esperado. O componente de software a ser testado pode ser: método, função interna, programa, componente, um conjunto de programas e/ou componentes ou mesmo uma funcionalidade. A técnica de teste funcional é aplicável a todos os níveis de teste (PRESSMAN, 2006).

Molinari (2008) destaca que os testes funcionais são os testes mais importantes, pois podem ser executado num ambiente corporativo no momento em que se resolve validar uma aplicação. Nos testes caixa-preta destaca-se: teste de aceitação.

Figura 2: Técnica de teste funcional

Os testes caixa-branca utilizam o código-fonte para avaliar os aspectos de comportamento interno do componente de software como: fluxo de dados, condição, ciclos e caminhos lógicos (PRESSMAN, 2006). Os testes caixa-branca reúnem: os testes de unidade, os testes de integração e os testes de sistema. A Figura 3, (DIAS-NETO, 2008), mostra como é a estrutura de um caso de teste caixa-branca.

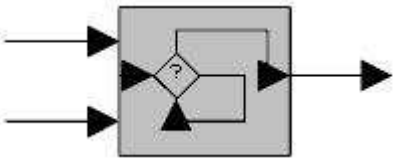

Figura 3: Técnica de teste estrutural

Teste de regressão consiste em aplicar, a cada nova versão do software ou a cada ciclo, todos os testes cujos resultados não resultaram em erro, que já foram aplicados nas versões ou ciclos de teste anteriores do sistema, isto é, os testes que exercitam as funcionalidades que não sofreram alterações. Pode ser aplicado em qualquer nível de teste (VIEIRA, 2004). O conjunto de testes que será reaplicado normalmente é definido pela equipe do projeto, já que muitas vezes não há tempo suficiente para refazer todos os testes de regressão.

2.1 Tipos de teste

Ricarte (2008) diz que os estudos de testes, em software orientado a objetos, devem ser divididos em quatro etapas: (1) testes de unidade, (2) testes de integração, (3) testes de sistema e (4) testes de aceitação. Os testes de unidade, integração e sistema são do tipo caixa-branca, já o teste de aceitação é do tipo caixa-preta.

Cada tipo de teste será explicado nas subseções seguintes.

2.1.1 Teste de unidade

Uma classe é construída com variáveis e métodos necessários para criar e também representar um objeto. O teste de unidade testa as operações no contexto da classe como unidades de software convencional, assim, a coleção de testes deve ser elaborada para que todos os

métodos, variáveis e constantes presentes numa classe sejam testados, para detectar os possíveis erros de semântica e sintaxe (MELNIKOFF, 2009).

O conjunto de testes de unidade deve ser bastante amplo a fim de poder testar o mais variado conjunto de entradas possíveis, e com isso assegurando que erros não ocorram devido a instanciação ou atribuição de valores não contextualizados num determinado método ou em uma variável presente na classe. Os testes devem ser planejados com a ideia de uma possível reutilização dos casos de teste num estado mais avançado da implementação do sistema. Quanto mais frequentes são os testes de unidade, mais confiável o software se torna, sendo recomendável o início dos testes no momento da codificação da classe (FERNANDES; LOPES, 2007).

Apesar da não complexidade deste conjunto de testes, deve se ter atenção nesta fase, pois os erros que não forem detectados em testes de unidade podem acarretar em um consumo elevado de tempo para a correção dos erros num estágio mais avançado do desenvolvimento do software e consequentemente, elevar o custo final.

2.1.2 Teste de integração

Segundo Binder (2006), os componentes dependem uns dos outros de diversas maneiras e são necessários para implementar colaborações e conseguir a separação de alguns interesses. Para testar se esses módulos dependentes conseguem desempenhar a função previamente conhecida, são utilizados os testes de integração. Esses testes representam a atividade de descobrir erros associados às interfaces entre os módulos quando esses são integrados. Os testes de integração possuem a finalidade de compor a estrutura do software que foi estabelecida na fase de projeto (ROCHA et al., 2001), isto é, é a maneira de se ter certeza que as unidades que compõem o software se comportam de maneira esperada conforme a descrição realizada na especificação dos requisitos (PFLEEGER, 2004).

Sommerville (2007) diz que a integração envolve identificar e unir os componentes que realizam uma determinada função no software e fazer com que eles trabalhem em conjunto. O principal problema do teste de integração é quando ocorre um erro e o sistema de integração de componentes é complexo, assim dificulta a identificação do local onde esse erro possa ter ocorrido. Uma forma de solucionar este problema é ir incrementando os componentes em composições mínimas e já os ir testando até chegar ao objetivo.

A discussão para definir qual a ordem de integração a ser testada se deve ao fato de que testes realizados não respeitando uma ordem natural podem resultar em dados equivocados como resultados dos testes, consequentemente o software fica mais propenso a erros, perdendo sua confiabilidade.

Com a ordem de prioridade definida uma relação entre as classes é estipulada, o processo de teste de integração é facilitado, pois já se tem uma possível relação adquirida entre as classes para uma fase de testes, simplificando a implementação das mesmas e proporcionando inclusive ganho em tempo e recursos. Os testes de integração de software são os mais complexos a serem desenvolvidos devido as propriedades do paradigma orientado a objetos ter mais impacto (MASIERO et al., 2006), - um estudo detalhado deve ser conduzido em cada sistema a fim de construir uma coleção de testes adequados que garantam a qualidade do sistema. É importante ressaltar que os testes de integração devem ser voltados principalmente às partes do sistema que mais são utilizadas e que a utilização de um registro deve ser mantido a fim de armazenar quais métodos de quais classes já foram exercitados nos testes (RICARTE, 2008).

Pesquisadores se dedicam a desenvolver estratégias e algoritmos para determinar a melhor ordem de integração das classes de forma que seja produzido um número menor de casos de teste, minimizando, assim, o esforço na hora de testar.

2.1.3 Teste de sistema

Teste de sistema é uma fase no processo de teste de software e hardware que utiliza o software completamente integrado para aplicação dos testes, isto é, seus requisitos são verificados num ambiente de produção com componentes devidamente preparados e configurados (PÈZZE, 2008).

O teste de sistema tem como finalidade principal exercitar o sistema por completo, são aplicados diferentes testes e todos trabalham para verificar se os elementos do sistema foram corretamente integrados e executam as funções a eles atribuídas (PRESMANN, 2006).

Esse tipo de teste não se detém a testar somente os requisitos funcionais. Os requisitos não-funcionais como expectativas do cliente, por exemplo, também devem ser testadas.

Sommerville (2007) utiliza uma abordagem diferente para testes de sistema, os divide em teste de integração e em teste de *release*. Testes de *release* são realizados devido ao versio-

namento do software que está sendo produzido. Neste tipo de teste é entregue ao cliente uma versão para que ele mesmo aceite ou reporte os erros para a equipe que está desenvolvendo.

Essa divisão de Sommerville é uma abordagem diferente na divisão dos tipos de testes, mas para manter a coerência será seguido o modelo proposto por Ricarte (2008).

2.1.4 Teste de aceitação

Testes de aceitação são testes do tipo caixa-preta e são realizados por um grupo específico de usuários finais do sistema os quais simulam operações de rotina, em cenário real, de modo a verificar se seu comportamento está de acordo com o solicitado. Este tipo de teste é conduzido para determinar se um sistema satisfaz ou não seus critérios de aceitação e para permitir ao cliente determinar se aceita ou não o sistema, é o último estágio de validação. Pode ainda ser incluídos testes de recuperação de falhas, de segurança, funcionais, de configuração e de desempenho (MYERS, 2004).

2.2 Estratégia de teste de software

Uma estratégia de teste, segundo Pressman (2006), fornece um roteiro que descreve o planejamento das etapas a serem seguidas para depois serem executadas: quanto tempo, esforço e recurso serão necessários para executar este conjunto de etapas. A finalidade de uma estratégia de teste é esclarecer as tarefas e os principais desafios do projeto do teste.

Para definir uma estratégia de teste para um software orientado a objeto, existem características próprias do paradigma, definidas no início deste capítulo, que merece uma atenção maior, pois, distintamente de softwares procedimentais, ao mesmo tempo em que elas tentam reduzir alguns tipos de erros, podem favorecer a adição e aparecimento de novos erros (MARTINS et al., 2009).

Uma estratégia utilizada para testar um sistema de software, segundo Pezzè (2008), deve começar com testes do tipo caixa-preta, que são os que simulam a parte funcional do software. Posteriormente, aplicar os testes do tipo caixa-branca que simulam a parte estrutural do código. Esta estratégia pode ser adotada independe do paradigma usado para desenvolver a aplicação.

Para tornar viável a aplicação dos testes funcionais antes dos testes de unidade, seguindo a estratégia adotada, o estudo de caso deve ser um software que esteja operando no ambiente de implantação dessa forma os testes buscam melhorar sua qualidade. Com a aplicação dos

testes funcionais, antes dos testes de unidade, a ferramenta sob teste primeiramente passa por validação, verificando se seu comportamento atende as necessidades do cliente, e somente depois se buscará descobrir onde que ocorreram os erros através dos testes de unidade.

No próximo capítulo, é apresentado o estudo de caso selecionado expondo os conceitos que envolvem a ferramenta na área da Engenharia Elétrica, as etapas seguidas para alcançar a finalidade que levou a sua construção e as funcionalidades testadas.

3 A FERRAMENTA DE ANÁLISE DE SISTEMAS ELÉTRICOS - ASE

3.1 ASE no conceito da Engenharia Elétrica

Uma rede de distribuição de energia elétrica é o meio utilizado para transportar a energia elétrica das unidades de geração até os grandes centros de consumo. Esta distribuição é realizada pela concessionária de energia através de equipamentos chamados de linhas de transmissão, que podem ser cabos aéreos sustentados através de pilares ou através de dutos subterrâneos.

Com a ocorrência de falha no fornecimento de energia elétrica em uma determinada região de uma rede, são causados muitos transtornos tanto para os consumidores quanto para a concessionária que é responsável por este fornecimento. Levando este problema em consideração foi lançado um desafio: como esta concessionária poderia sanar o transtorno para que suas equipes de manutenção não se deslocassem de forma errônea e para que os seus clientes não tivessem grandes perdas econômicas?

O grupo de pesquisa GESEP (Grupo de Energia e de Sistemas Elétricos de Potência) da Universidade Federal do Pampa desenvolveu uma ferramenta, nomeada de Análise de Sistemas Elétricos, ou simplesmente ASE, que analisa a rede de energia elétrica em operação e através de simulações de operação, faz um estudo de quais das chaves existentes na rede poderiam vir a ser substituídas por chave telecomandada por sistema computacional.

As chaves são equipamentos colocados em alguns dos pilares de sustentação dos cabos que transportam a energia elétrica com o objetivo de interromper, de modo visível, a continuidade de um dado circuito - rede energizada - permitindo assim a manutenção segura nesta parte do circuito, pois os cabos de transmissão estarão desenergizados.

Após a análise da simulação de operação da rede de distribuição de energia elétrica, é criado um ranking das melhores chaves utilizando multicritérios baseados em dados de criticidade, como clientes sensíveis, no caso os hospitais, que não podem ficar sem o fornecimento de energia e também baseados em dados de indicadores globais como DEC- Duração Equivalente por consumidor, FEC- Frequência Equivalente por Consumidor e ENS- Energia Não Suprida e dados de indicadores individuais como DIC- Duração Individual por Consumidor, FIC- Frequência Individual por Consumir e DMIC- Duração Máxima Individual por Consumidor. O ranking em questão é mostrado através de um gráfico de ganho para cada chave instalada. A chave que obtiver o maior ganho pode ser substituída por uma telecomandada (RECK; GIACOMELLI, 2009).

A vantagem deste tipo de substituição de chaves é que a concessionária consegue restabelecer o fornecimento de energia em um tempo reduzido, pois a operação de liga/desliga é feita a distância por comandos de sistemas computacionais, ou seja, não há a necessidade de deslocamento de equipes de manutenção para ligar ou desligar estas chaves.

Na topologia de rede, as chaves são inseridas sempre em pares, sendo que uma é Normalmente Fechada, ou NF, e a outra é Normalmente Aberta, ou NA. Esta última faz a divisa entre o seu alimentador e o que suporta a carga a jusante da NF (SPERANDIO, 2008), isto é, a NF está localizada entre uma NA e seu próprio alimentador.

Após ter inserido a ferramenta nos conceitos que a envolvem na área da Engenharia Elétrica, a seção 3.2 apresenta as funcionalidades desta ferramenta utilizadas para construir os casos de teste baseados no comportamento do sistema.

3.2 Funcionamento da ASE

A ferramenta ASE possui três estágios que realizam cálculos até encontrar a chave correta para ser substituída por uma telecomandada.

No primeiro estágio é calculado o impacto da inserção de chaves em uma rede de distribuição de energia elétrica que se resume ao cálculo dos indicadores, tanto globais quanto individuais. Após, calcula a composição do ranking das possíveis chaves que podem ser substituídas por chaves telecomandadas com base em indicadores globais e em fatores de criticidade (RECK; GIACOMELLI, 2009).

Para calcular os indicadores globais (DEC, FEC, ENS) e individuais (DIC, FIC, DMIC) é construída uma Matriz Lógico-Estrutural (MLE), onde os valores de tempo e taxa de falhas

são utilizados para a construção da matriz. Estas informações são extraídas de um banco de dados da concessionária, onde ficam armazenadas todas as ocorrências das falhas em um determinado período e também fica armazenado o tempo de manobra (isolamento e transferência) que são enviados como dados de entrada (RECK; GIACOMELLI, 2009).

A partir do tempo e taxa de falhas, de cada chave, é feita uma redistribuição das taxas de falhas das chaves de proteção sobre as demais localizadas nos ramos troncais (ramo principal da rede de distribuição de energia elétrica). Essa redistribuição é feita já que o defeito que era em uma chave qualquer fica contabilizado como falha em um equipamento de proteção. Desta forma é necessária a distribuição da taxa de falhas entre as chaves dos ramos troncais, pois, em alguns casos, um defeito em uma chave troncal (que não seja de proteção) se propaga até encontrar um equipamento de proteção, podendo assim ter passado por outras chaves que não apresentavam nenhum tipo de falha.

Para calcular a redistribuição das taxas de falhas das chaves de proteção dos ramos troncais é necessário calcular a distância entre as chaves em unidades de quilômetros. Para cada chave de proteção são percorridos trechos troncais protegidos por ela e para cada chave encontrada nesse trecho é calculada a taxa de falhas através da equação 1. Nesse processo as chaves de proteção também recebem o valor da equação 1.

$$TFc = \left(\frac{KMc}{KMp}\right) * TFp \quad (1)$$

onde *TFc* é a nova taxa de falhas, *KMc* é o comprimento dos trechos protegidos pela chave, *KMp* é o comprimento total coberto pela chave de proteção e *TFp* a taxa de falhas original da chave de proteção.

Logo após essa redistribuição, a matriz MLE é preenchida e o cálculo dos indicadores iniciais é realizado e ficam armazenados para futuras comparações. Uma observação importante é que somente as chaves consideradas pontos notáveis (PN) têm sua taxa de falhas alterada pela redistribuição. Os PN são chaves que a concessionária executa manobras (isolamento e transferência).

O segundo passo é fazer o cálculo que irá escolher a chave candidata para ser substituída por uma telecomandada, sabendo-se que uma chave candidata deve ser capaz de manter a rede estável sem que haja sobrecarga nos condutores ou queda de tensão. O disjuntor não é levado em consideração para ser candidato a chave telecomandada.

Para a escolha da chave, são separados todos os pares de chaves NA e, as NF pertencentes aos ramos troncais. Com todos estes pares separados, é efetuada a transferência iniciando pela chave mais a montante até encontrar uma chave que atenda aos requisitos de sobrecarga e queda de tensão. Essa chave e as a jusante dela são consideradas chaves candidatas (RECK e GIACOMELLI, 2009).

O terceiro e último estágio é quando o cálculo do impacto de alocação das chaves telecomandadas é realizado. Na Figura 4 (RECK; GIACOMELLI, 2009) é mostrada uma topologia de rede de distribuição de energia elétrica:

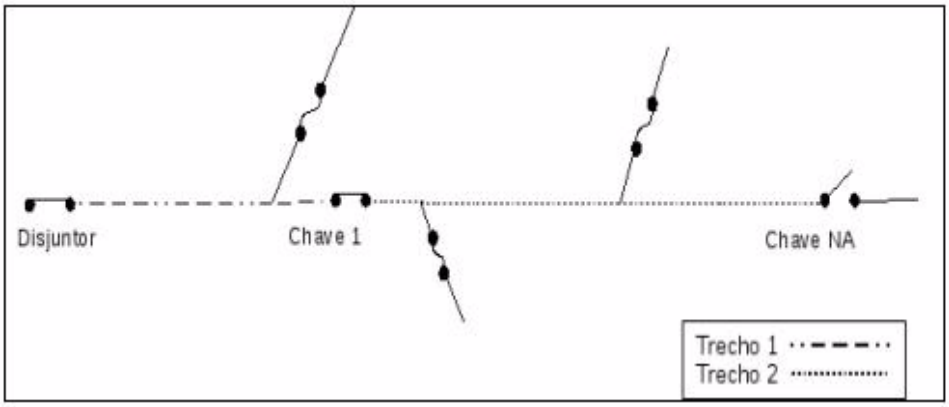

Figura 4: Topologia de rede de distribuição de energia elétrica

Instalar a chave telecomanda no par Chave 1-Chave NA, irá alterar a MLE de modo que: para caso de falhas a jusante da chave, todos consumidores a montante (trecho 1) sejam isolados em menor tempo, mantendo os mesmos tempos para os demais consumidores. Para falhas a montante, os clientes a jusante (trecho 2 e trecho posterior a Chave NA) são transferidos também em menor tempo (RECK; GIACOMELLI, 2009).

Para o cálculo do impacto do par Chave NA (segunda chave), primeiramente a MLE será retornada ao seu estado anterior e a Chave 2 terá que se tornar PN temporariamente, mudando a redistribuição das taxas de falhas. As alterações promovidas na MLE pela troca desse par de chaves será muito semelhante ao par testado anteriormente. A diferença agora é que já existe uma chave que pode ser usada para transferência de carga, assim, para defeitos a montante da Chave 1, será feita a transferência com a Chave NA (trecho posterior a Chave NA), uma vez que é mais rápida e depois da Chave 1 (trecho 2). Para defeitos a jusante da Chave 2, o tempo de isolamento será considerado da Chave 2 (RECK; GIACOMELLI, 2009).

Para saber a redução dos indicadores provocada pela inserção de um par de chaves é efetuado o seguinte procedimento:

a. Salvar a MLE atual: a matriz é salva para restauração futura;

b. Transformar o par de chaves em chaves telecomandadas: isso inclui dizer que a chave é também um PN, já que a concessionária passará a executar manobras com elas;

c. Redistribuir as taxas de falha;

d. Recalcular a MLE: recalcular o impacto que a inserção dessa chave provoca na MLE;

e. Calcula os indicadores globais e verifica a diferença deles para os indicadores originais;

f. Retornar aos tipos originais do par de chaves;

g. Restaurar a MLE para estado inicial.

Esse procedimento é executado para todas as chaves candidatas e as diferenças dos indicadores, juntamente com os dados de criticidade, são inseridas em um relatório para análise.

A ferramenta ASE foi construída na linguagem de programação C++, portanto, uma linguagem orientada a objetos, onde a estratégia de teste pode ser executada. Esta ferramenta passou por um pequeno número de testes e com cobertura superficial, antes de entrar em operação em uma companhia de distribuição de energia elétrica. Em se tratando de um software que está em uso, melhorar a sua qualidade através da realização de uma bateria de testes estrategicamente definidos é o objetivo deste trabalho.

3.3 As funcionalidades da ASE

A ferramenta ASE tem funcionalidades específicas para cálculos de alocação de chaves ótimas que podem vir a serem substituídas por chaves telecomandadas por sistema computacional, essas funcionalidades vão desde abertura de arquivos à alocação das chaves. A Figura 5 apresenta a interface gráfica do usuário, para tornar mais fácil o entendimento da descrição das funcionalidades da ASE apresentadas a seguir:

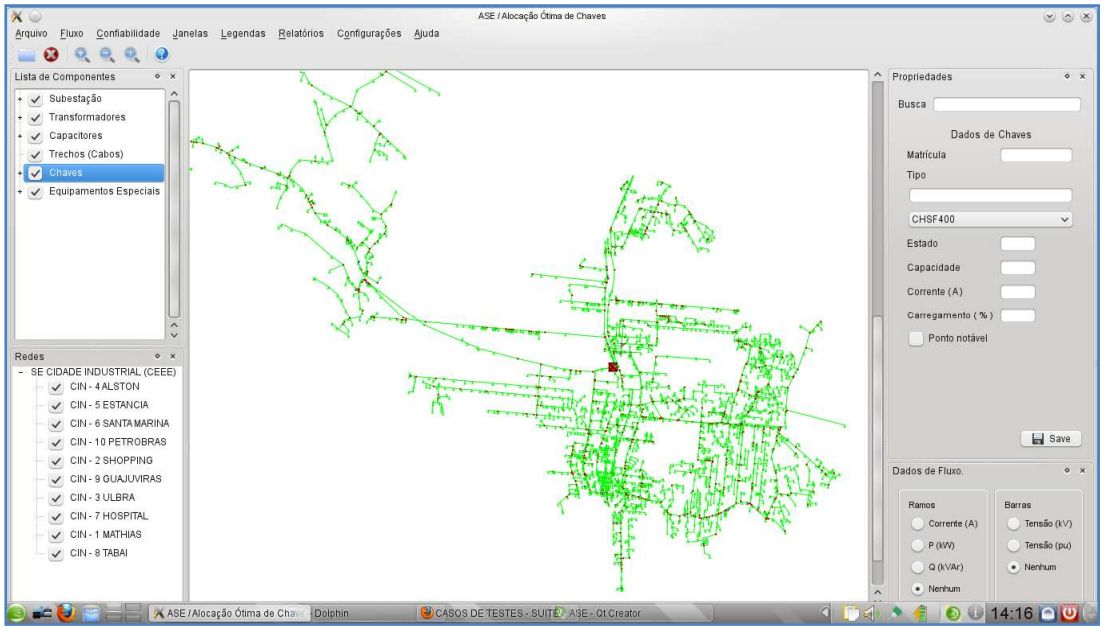

Figura 5: Interface gráfica com o usuário da ASE

a. Abrir arquivo, fechar arquivo e sair: através da barra de menu na opção arquivo é possível abrir as redes salvas em arquivos, fechá-las ou até mesmo sair da ASE. Essas funcionalidades estão disponíveis também na barra de ferramentas;

b. Ajuste de carga: para efetuar o ajuste para um alimentador basta alterar o valor de corrente, demanda ativa ou demanda reativa, de acordo com o método de correção desejado, e clicar em 'Altera'. Lembrando que o ajuste de carga é feito por alimentador tendo que selecioná-lo para executar a função de ajuste de carga;

c. Modo edição: é o modo normal do ASE. O modo só é trocado quando é calculado o fluxo de potência;

d. Cálculo de confiabilidade: são calculados os tempos de espera, deslocamento, serviço, taxa de falhas e tempo médio de desarme de todas as chaves da rede a partir de um banco de dados, já construído, contendo todos os dados de desarma de chaves. Os valores padrões para isolamento da chave anual e telecomandada e transferência de carga estão preenchidos nos campos correspondentes e podem ser alterados;

e. Mostra as chaves telecomandadas alocadas na rede: após fazer a alocação de chaves telecomandada e clicando em "Mostrar", as chaves telecomandadas são marcadas com um T (T maiúsculo dentro de um círculo) na área de vizualização, que contém a rede carregada, da ASE;

f. Exibe, salva, fecha e minimiza relatórios de indicadores globais (DEC, FEC, ENS) e individuais (DIC, FIC, DMIC): salva em formato .cvs (recomendado) para tornar possível visualizá-los em um documento de planilha eletrônica. Estes documentos são: relatórios de redução de indicadores, relatório de carregamento dos trafos, relatório de carregamento dos trechos, relatório de resumo por alimentador e relatório de tensão dos trafos;

g. Calcula o fluxo de potência: em uma rede de distribuição de energia elétrica é possível selecionar quais partes da rede deseja calcular o fluxo, e pode-se calcular o fluxo de toda a rede. Também há a possibilidade de alterar os valores de corrente normal, corrente emergencial, limite precário de tensão, limite crítico de tensão, todos em porcentagem. Após o cálculo do fluxo de potência o ASE passará para o 'Modo Fluxo', mostrando dados de corrente de condutores e tensão, colorindo a rede de acordo com os níveis calculados: vermelho para condutores com corrente acima da capacidade (carregamento maior que 95%) e transformadores com valores de tensão abaixo do normal (voltagem menor que limite crítico), amarelo para condutores e transformadores com níveis críticos de carregamento e tensão (carregamento entre 90% e 95%) e voltagem entre limite crítico e precário) e verde para níveis normais de carregamento e tensão;

h. Exibe os componentes elétricos desejados de uma rede: através de um menu de componentes há a possibilidade de mostrar apenas alguns componentes para serem exibidos na tela, são estes: subestação, transformadores, capacitores, trechos (cabos), equipamentos especiais e chaves;

i. Zoom: com a função de zoom mais (pressionando a tecla Ctrl e tecla + simultaneamente) e zoom menos (pressionando a tecla Ctrl e tecla - simultaneamente) pode-se aumentar ou diminuir, respectivamente, determinado trecho da rede de energia elétrica;

j. Exibe informações de um determinado componente da rede: ao clicar em um ícone de transformador, por exemplo, aparece os dados do transformador selecionado: matrícula, tipo, estado, capacidade, corrente (medida em Amperes), porcentagem de carregamento e pode marcar esta como ponto notável - PN. O ícone na tela aparece mais escuro para fins de identificação com maior celeridade;

k. Aloca as chaves telecomandadas em uma rede: tendo como base os dados fornecidos, o ASE faz o cálculo para descobrir as chaves candidatas a serem substituídas por chaves telecomandas e calcula o impacto positivo da alocação de cada uma dessas chaves.

Depois de conhecer a ferramenta usada como estudo de caso, o próximo capítulo contém o planejamento dos casos de teste. É mostrado como estão armazenados os casos de teste, a forma usada para elaborá-los, como foram executados e o resultado dos casos de teste com a classificação das falhas, se houverem, e como serão publicados estes resultados obtidos após a execução dos mesmos.

4 PLANEJAMENTO DOS CASOS DE TESTE

O planejamento dos testes é uma etapa caracterizada por definir uma proposta de testes baseada nas expectativas do cliente em relação a prazos, custos e qualidade esperada, possibilitando dimensionar a equipe e estabelecer um esforço de acordo com as necessidades apontadas pelo cliente. Esta é uma tarefa árdua, pois se não for bem planejada, o custo final do software pode aumentar exponencialmente (BARTIÉ, 2002).

No planejamento dos testes, é detalhado todo o processo de realização dos testes, para ilustrar são citados alguns detalhes como: os diferentes tipos de testes que serão aplicados; quem fará os casos de teste e os executará; como os casos de teste estarão organizados; como que o resultado será publicado.

Este trabalho foi dividido em duas etapas para a criação dos casos de teste visando dividir os testes funcionais dos testes unitários. Como resultado cada etapa conterá o conjunto documentado dos casos de teste, onde cada caso de teste está identificado individualmente e o resultado obtido após a sua execução estará anexado na linha de resultado.

O primeiro conjunto contém os casos de teste caixa-preta, exercitado sobre as funcionalidades da ferramenta ASE. São os testes realizados utilizando a interface gráfica da aplicação da mesma forma que os usuários finais da aplicação farão uso da ferramenta.

O segundo conjunto contém os casos de teste de unidade da classe MLE, esta classe é a principal da ferramenta, pois é através desta que a alocação de chaves telecomandadas em uma rede de distribuição de energia elétrica é realizada, que é o propósito da ferramenta.

A última fase de teste foi selecionada de acordo com o grau de importância que a classe exerce no projeto ASE. A classe MLE foi escolhida depois de uma análise, juntamente com o grupo de estudos GESEP, pois é através dela que a alocação de chaves ótimas é realizada.

O resultado das duas etapas de teste foi publicado parcialmente à medida que os testes foram realizados. As falhas encontradas foram anexadas no Mantis Bug Tracker, software utilizado para reportar falhas, através deste, a equipe de desenvolvimento terá acesso ao conteúdo e poderá trabalhar na correção dos problemas encontrados.

Devido ao prazo para conclusão deste trabalho, não foi possível criar casos de teste para as demais classes da ASE, ressaltando que são tão importantes quanto a classe MLE selecionada, sem este conjunto de classes a ASE não desempenharia suas funções corretamente.

4.1 Elaboração dos casos de teste

Um caso de teste é um conjunto de condições usadas para testar software, é elaborado com o intuito de identificar falhas na estrutura interna do software, através de situações que exercitem adequadamente todas as estruturas utilizadas na codificação; ou ainda, garantir que os requisitos do software que foi construído sejam plenamente atendidos (MYERS, 2004).

O caso de teste deve especificar a saída e os resultados esperados do processamento. Numa situação ideal, no desenvolvimento de casos de teste, se espera encontrar o subconjunto dos casos de teste possíveis com a maior probabilidade de encontrar a maioria dos erros (MYERS, 2004).

A estrutura para documentar um caso de teste é composta pelos seguintes elementos: identificação do caso de teste, descrição, as pré-condições para que seja válida a aplicação do teste, a entrada do caso de teste, o resultado esperado e o resultado obtido. Caso o resultado obtido não condiga com o esperado, então houve uma falha e é necessário fazer a sua classificação de acordo com o nível de severidade da falha. A Tabela 1, explica detalhadamente os elementos que compõe cada caso de teste.

A classificação da severidade e da prioridade de correção de uma falha é um ponto normalmente negligenciado e foco de interpretações incorretas (CAETANO, 2007). A classificação incorreta da severidade e da prioridade normalmente contribui para a ineficiência da priorização e agendamento da correção dos defeitos. As consequências desse problema é a má utilização dos recursos em função do enfoque na correção de defeitos menos prioritários.

A severidade de uma falha define o impacto desta no bom funcionamento da aplicação e a prioridade indica a ordem de correção da falha, as falhas com prioridade alta devem ser as primeiras a serem corrigidas, por exemplo.

Estrutura do Caso de Teste	Descrição
Identificação do caso de teste	Nomenclatura que identifica o caso de teste
Descrição	Descreve a finalidade ou o objetivo do teste e o escopo do caso de teste
Pré-condições	Para cada condição de execução, descreve o estado obrigatório do sistema antes do início do teste.
Entrada	Para cada condição de execução, enumera uma lista dos estímulos específicos a serem aplicados durante o teste. Em geral, eles são denominados entradas do teste e incluem os objetos ou os campos de interação e os valores de dados específicos inseridos durante a execução deste caso de teste.
Resultado esperado	É o estado resultante ou as condições observáveis esperadas como resultado da execução do teste. Observe que isso pode incluir respostas positivas e negativas (como condições de erro e falhas).
Resultado obtido	É a resposta obtida após o caso de teste ser executado (pode ser positiva ou negativa).
Classificação da falha	Ao encontrar uma falha, deve ser classificada para auxiliar na priorização de sua correção.

Tabela 1: Estrutura de um caso de teste

As falhas podem ser classificadas em até cinco níveis de severidade, mas para uma melhor distinção entre esses níveis foi escolhido utilizar apenas três, o nível de severidade baixa, média e alta tornando possível a classificação das falhas com uma precisão maior. A prioridade para correção das falhas é de acordo com o nível de severidade estabelecendo se a liberação para o uso da ferramenta pode ser dado (CAETANO, 2007). A Tabela 2 mostra a descrição dos três níveis de severidade e a prioridade de correção das falhas.

Os casos de teste estão armazenados e disponibilizados em formato de planilha eletrônica de acordo com o conjunto que o identifica. Essa documentação é de suma importância, pois ao acoplar novas funcionalidades na ASE não será necessário criar novamente todos os casos de teste, mas sim, somente testes para cobrir as novas funcionalidades. A economia de tempo

fará com que a aplicação dos testes possa ter um alcance mais abrangente e, portanto, a qualidade em termos de confiabilidade e segurança aumentará.

Classificação da falha	Nivel de severidade	Prioridade de Correção
Baixa	Solicitações de melhorias	Defeito de baixa prioridade. A aplicação pode ser liberada com esse defeito
Média	Bloqueia a utilização de uma funcionalidade. Mas, no entanto, a funcionalidade pode ser usada por meio da utilização de um contorno (*workaround*) conhecido	É altamente desejável que o defeito seja corrigido tão logo seja possível. A aplicação não pode ser liberada sem a correção deste defeito
Alta	Bloqueia completamente a utilização de uma funcionalidade ou da aplicação inteira	O defeito deve ser corrigido imediatamente. A aplicação não pode ser liberada sem a correção deste defeito

Tabela 2: Classificação das falhas em níveis de severidade

Após mostrar como foi selecionada a estrutura dos casos de teste a e classificação de falhas quando houver, a subseção 4.1.1 mostra como foram elaborados os casos de teste para o conjunto 1 e a subseção 4.2.2 para o conjunto 2, respecticvamente.

4.1.1 Casos de teste funcionais

A elaboração dos casos de teste funcionais foi realizada de acordo com as funcionalidades disponíveis na interface gráfica com o usuário da ferramenta ASE. Os casos de teste criados exercitam estas funcionalidades com entradas válidas, entradas inválidas, com carregamento de redes de energia elétrica sob uma rede já disponível para teste, com cálculos de confiabilidade sem carregar rede, com zoom em determinados ramos da rede, etc. Para cada funcionalidade disponível na ASE, foi criado, no mínimo, um caso de teste.

As funcionalidades usadas para criar os casos de teste estão disponíveis na seção 3.3, na Tabela 4, estas mesmas funcionalidades estão associadas aos casos de teste, do conjunto 1, criados para exercitá-las.

Funcionalidades	Casos de teste
a. Abrir arquivo, fechar arquivo e sair	01, 03, 04, 06, 09, 10, 11, 12, 64
b. Ajuste de carga	08, 67
c. Modo edição	18
d. Cálculo de confiabilidade	55, 60
e. Mostra chaves telecomandadas alocadas na rede	62, 65, 66
f. Exibe, salva, fecha e minimiza relatórios de indicadores globais e individuais	16, 17, 19,20, 21, 22, 23, 24, 38, 39, 44, 45, 46, 47, 48, 49, 50, 51, 56, 57, 58, 59
g. Calcula o fluxo de potência	07, 28, 29, 40, 41
h. Exibe compontes elétricos desejados de uma rede	31, 32, 33, 35, 36, 37
i. Zoom	05, 13, 14, 15, 30
j. Exibe informações de um determinado componente da rede	02, 25, 26, 27, 34, 42, 43, 52, 53, 54, 63
k. Aloca chaves telecomandadas	61, 65, 66

Tabela 3: Casos de teste que exercitam cada funcionalidade

4.1.2 Casos de teste unitários

Os 18 casos de teste unitários elaborados exercitam a maioria dos métodos da classe MLE, apresentada na Figura 6. A classe MLE é a classe principal do sistema na qual são calculados os indicadores globais (DEC, FEC e ENS) e indicadores individuais (DIC, FIC e DMIC) para cada chave de uma rede de energia elétrica. Depois de realizado o cálculo dos indicadores para cada chave é montado um ranking das chaves que têm um maior ganho e estas são candidatas a telecomendadas, isto é, podem vir a ser substituídas por chaves telecomandadas.

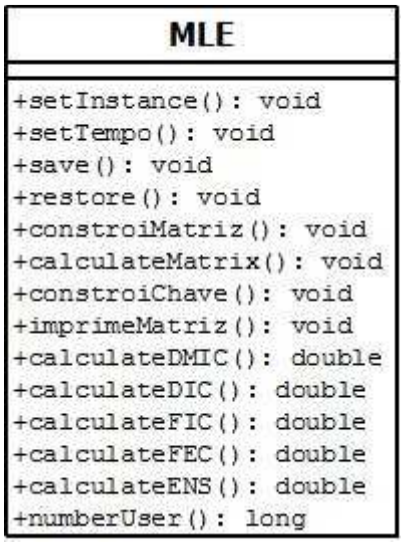

Figura 6: Classe MLE

No Apêndice A, estão as demais classes da ferramenta ASE ilustradas em um diagrama de classes com a classe MLE destacada.

4.2 Execução dos casos de teste

A etapa de execução dos casos de teste é caracterizada pela execução e conferência dos testes planejados, de forma a garantir que o comportamento do sistema permanece em "conformidade" com os requisitos contratados pelo cliente.

Na realização deste trabalho, à medida que foram sendo criados os casos de teste a execução foi sendo realizada em sequência e, portanto, a identificação dos casos de teste é numérica e em ordem crescente de acordo com o número que representa a ordem de criação no conjunto.

Os casos de teste seguem a modelagem exposta na seção 4.1 e foram divididos em subseções de acordo com o conjunto que os representa. Os casos de teste utilizados nestas subseções foram escolhidos para representar os possíveis resultados dos casos de teste, os que não resultaram em falhas e os outros que resultaram em falhas levando em consideração o nível de severidade da falha.

4.2.1 Casos de teste funcionais

Nessa seção, estão expostos alguns dos casos de teste funcionais pertencentes ao conjunto 1 para mostrar como foram executados os testes desse conjunto.

O caso de teste 63, Tabela 4, se refere à visualização das propriedades (matricula, tipo, estado, etc) de um determinado transformador (CAN-1839) da uma rede de energia elétrica devidamente carregada na tela da ASE. Para executar o teste, a entrada é através de um clique duplo sobre o transformador desejado. O resultado do caso de teste foi positivo, portanto, não houve falhas ao executar essa entrada de teste.

CASO DE TESTE 63	
Descrição:	Visualizar as propriedades de um transformador
Pré-condições:	Ter a rede de testes carregada na tela
Entrada:	Clique duplo em cima do transformador CAN-1839 da rede SE_OUTR-METR-CIN.TXT
Resultado esperado:	Frame propriedades com os campos preenchidos com informações (matrícula, tipo, estado, etc) do transformador
Resultado do teste:	**SUCESSO**

Tabela 4: Caso de teste 63 – Conjunto 1

O caso de teste 10, Tabela 5, se refere à configuração da tela inicial da ASE. A entrada é carregar uma rede de teste existente nos arquivos de teste e válida. O resultado obtido foi uma falha de severidade baixa, pois a tela incial desconfigurou, mudou de tamanho, mas isso não impossibilita a utilização do sistema.

CASO DE TESTE 10	
Descrição:	Configuração da janela ao carregar uma rede
Pré-condições:	Ter uma rede para carregar
Entrada:	Arquivo > Abrir Arquivo > buscar a rede TesteMLE.aes
Resultado esperado:	Ao abrir a rede de testes a janela principal não deve sofrer alterações
Resultado do teste:	Janela desconfigurada no lado inferior da tela – FALHA
Classificação da falha	BAIXA

Tabela 5: Caso de teste 10 – Conjunto 1

O caso de teste 64, Tabela 6, deve fechar uma rede existente na tela e abrir outra sem reiniciar o sistema. O resultado obtido foi uma falha de severidade média. Através de um mecanismo adicional – *backaround* (contorno que o usuário utiliza para fazer com que a funcionalidade se comporte conforme esperado) - é possível realizar esta atividade. Se atualizarmos o sistema a nova rede de teste aparece na tela, caso contrário, ficam vestígios da antiga.

CASO DE TESTE 64	
Descrição:	Fechar uma rede previamente carregada e abrir outra rede diferente
Pré-condições:	Ter duas redes de testes, uma carregada e outra disponível
Entrada:	Clique no no botão fechar da barra de ferramentas > Ctrl +O > SE_OUTR-METR-CIN.TXT > Abrir
Resultado esperado:	Fechar a rede RedeTesteMLE.aes a abrir a rede SE_OUTR-METR-CIN.TXT sem inicializar o sistema
Resultado do teste:	Fica partes da rede 1 e a segunda não aparece só com F5 – FALHA
Classificação da falha	MÉDIA

Tabela 6: Caso de teste 64 – Conjunto 1

O caso de teste 20, detalhado na Tabela 7, deve salvar um resumo de relatório do fluxo de potência por carregamento de trechos sem nomear o arquivo rede. O resultado obtido foi uma falha de severidade alta, pois pede para sobrescrever o arquivo no diretório que o usuário definir. É uma falha grave, pois deveria aparecer uma mensagem na tela que não há nome no arquivo, não pode simplesmente ir sobrescrevendo arquivos existentes em algum diretório. Pode comprometer o trabalho anterior do usuário, pela perca do arquivo anterior.

CASO DE TESTE 20	
Descrição:	Salvar o resumo do relatório do fluxo de potência por Carregamento dos Trechos - rede RedeTesteMLE.aes- sem nomear o arquivo
Pré-condições:	Ter o relatório do resumo por alimentador na tela
Entrada:	Relatórios > Relatórios do Fluxo de Potência > Relatório de Carregamento dos Trechos > Salvar
Resultado esperado:	Arquivo salvo no diretório que o usuário definir
Resultado do teste:	Mensagem pra sobrescrever o arquivo pois já existe neste diretório – FALHA
Classificação da Falha	ALTA

Tabela 7: Caso de teste 20 – Conjunto 1

Dado que a quantidade de casos de teste funcional é elevada, os demais casos de teste e seus resultados estão disponíveis no Apêndice B.

4.2.2 Casos de teste unitários da classe MLE

Nessa seção, estão expostos os casos de teste unitários, do conjunto 2.

O caso de teste 11, mostrado na Tabela 8, verifica se a ferramenta está operando conforme o especificado, ou seja, sem simulação de erros. O resultado desse caso de teste não resultou em falhas, se comportou conforme o esperado.

CASO DE TESTE 11	
Descrição:	Calcular o número de consumidores da rede
Pré-condições:	Ter uma MLE com valores válidos
Entrada:	Soma o número de clientes de cada transformador da rede
Resultado esperado:	Número de consumidores= 800, pois cada um dos 8 transformadores tem 100 clientes
Resultado do teste:	SUCESSO

Tabela 8: Caso de teste 11 – Conjunto 2

O código em C++ do caso de teste 11 está disponível na Figura 7, através dele pode ser visto o que realmente foi testado. Nesse caso de teste é criada uma nova MLE com as chaves da rede disponível no arquivo de dados que contém a rede. Se o número de chaves é diferente de zero então obtém as chaves do arquivo, constrói a MLE e calcula o número de usuários da rede através do método numberUser da classe MLE.

```
Void BuildMLE::numberUser(){
   MLE * mle = new MLE(inst);
   Switch * s = ASEBaseData::getSwitch("TELE");
   QVERIFY(s != 0);
   this->inst->getSwitch(1)->getSwitchType()->setChave(s);
   mle->constroiMatriz()
   mle->getNumberUsers();
   delete mle;
}
```

Figura 7: Código do caso de teste 11 – Conjunto 2

O caso de teste 16, na Tabela 9, mostra que ao simular uma falha o ASE produz resultados compatíveis. Nesse caso de teste foi simulado o cálculo do DMIC para um transformador 9, vide Figura 8, e a rede construída para os testes tem somente 8 transformadores. O resultado do caso de teste 16 mostrou que ao simular o erro de calcular o DMIC para um transformador não exitente na rede o ASE apresentou a falha.

CASO DE TESTE 16	
Descrição:	Calcular o DMIC para trafo não exitente na rede
Pré-condições:	Ter uma MLE com valores válidos
Entrada:	Calcular o DMIC para o transformador 9 sendo que na matriz só existem 8
Resultado esperado:	MLE não se comporta como esperado por dados não compatíveis
Resultado do teste:	SUCESSO

Tabela 9: Caso de teste 16 – Conjunto 2

As linhas de código em C++ do caso de teste 16 estão disponíveis na Figura 8, onde pode ser visto o que realmente foi testado. Nesse caso de teste é criada uma nova MLE com as chaves da rede disponível no arquivo de dados que contém a rede. Se o número de chaves é dife-

rente de zero então obtém as chaves do arquivo, contrói a MLE e calcula o DMIC através do método calculateDMIC da classe MLE.

Esta simulação de erro é dada porque ao criar a matriz com oito posições (matrix[0][1] = 0.05 até a posição matrix[7][1] = 0.05), o argumento do método calculateDMIC está acima do limite superior (mle->calculeDMIC(9)) permitido que nesse caso é 7.

```
Void BuildMLE::dmicTrNoExistent (){
   MLE * mle = new MLE(inst);
   Switch * s = ASEBaseData::getSwitch("TELE");
   QVERIFY(s != 0);
   this->inst->getSwitch(1)->getSwitchType()->setChave(s);
   mle->constroiMatriz();
                                    matrix[0][1] =  0.05;
                                    matrix[1][1] =  0.05;
      matrix[2][0] = 0.0833;
      matrix[3][0] = 0.0833;
      matrix[4][0] = 0.0833;
      matrix[5][0] = 0.0833;
      matrix[6][0] = 0.0833;
                                   matrix[7][1] =  0.05; // 8 TRAFOS
   mle->calculeDMIC(9);
   delete mle;
}
```

Figura 8: Código do caso de teste 16 – Conjunto 2

Os demais casos de teste unitários do conjunto 2 estão disponíveis no Apêndice C.

4.3 Resultado dos casos de teste

No resultado dos testes o objetivo é criar um relatório que tem os resultados resumidos para prover avaliações baseadas nestes resultados. Através do relatório de resultado dos testes é possível avaliar a conformidade do software em relação aos requisitos do projeto de teste, especificados pelo cliente.

Cada conjunto de teste conterá:

- O número total de casos de teste: quantidade de testes escritos para executar as funcionalidades da ASE;

- A quantidade que resultou em sucesso: número de casos de teste que não resultaram em falha;
- A quantidade que resultou em falha: número total de casos de teste que resultaram em falha e o número de casos de teste de acordo com o tipo da falha;
- Dois gráficos: um que representa a porcentagem de testes que resultaram em sucesso versus falha e um segundo com a porcentagem de cada tipo de falha encontrado.

Nas subseções 4.3.1 estão armazenados os resultados de casos de teste funcionais, do conjunto 1, já a subseção 4.3.2 contém os resultados dos casos de teste unitários da classe MLE, do conjunto 2.

4.3.1 Resultado dos casos de teste funcionais

Nesse conjunto de testes foram escritos 67 casos de teste. Destes, 43 não falharam e 24 resultaram em falhas. Na classificação das falhas de severidade baixa, 2 casos de teste falharam, as de severidade média somaram 10 falhas e as severidade alta somaram 12 falhas.

Considerando as informações descritas acima, foi realizada uma comparação em porcentagem para dar uma noção de proporção nos resultados dos testes. A Figura 9 mostra que 64% dos testes aplicados não falharam e nos outros 36% foi encontrado algum tipo de falha.

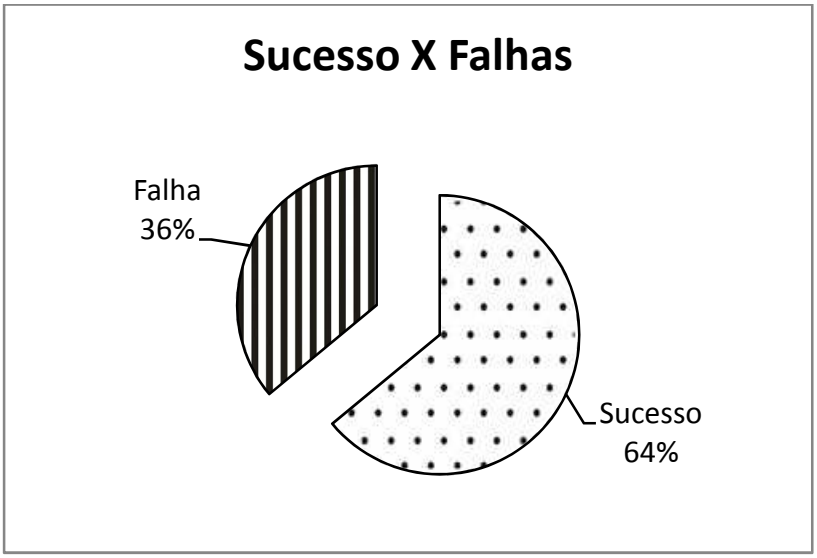

Figura 9: Resultado dos casos de teste - Conjunto 1

Dentre os testes que resulataram em falhas, 3% foram do nível de severidade baixa, 15% de nível médio e 18% do nível alta. De acordo com a classificação das falhas, a Figura 10

apresenta o resultado dos casos de teste para o número total de falhas, ou seja, dos testes que falharam 8% foi de severidade baixa, 42% de severidade média e 50% de severidade alta.

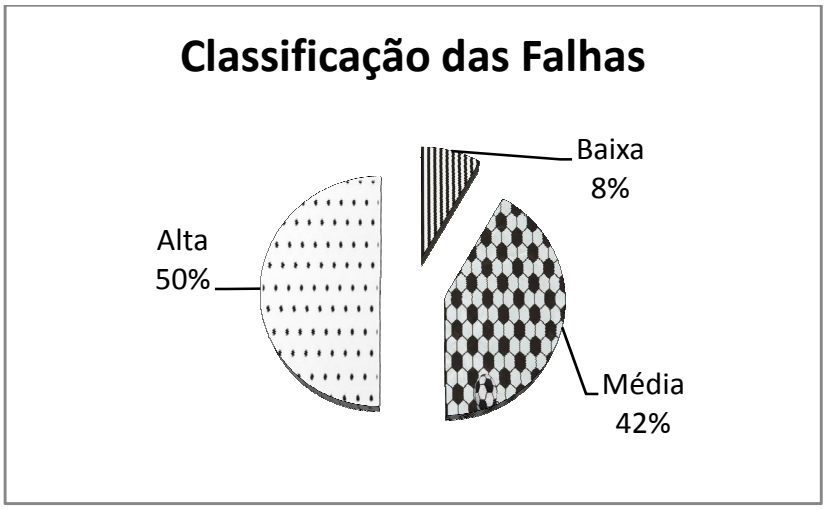

Figura 10: Classificação das falhas de acordo com o número total de falhas

4.3.2 Resultado dos casos de teste de unidade da classe MLE

Para verificar se todos os métodos da classe estavam corretos, foram simuladas algumas falhas nos dados de entrada para esses métodos, e a classe MLE respondeu conforme esperado. Para a realização dos testes unitários da classe MLE foram construídos 18 casos de teste, dentre estes nenhum resultou em falha.

Após a obtenção do resultado dos casos de teste, no próximo capítulo é o das considerações finais, estão as conclusões obtidas a partir dos resultados dos casos de teste, as dificuldades encontradas para a realização deste trabalho e sugestões para trabalhos futuros.

CONSIDERAÇÕES FINAIS

O principal objetivo da atividade de teste de software é diminuir o número de falhas a um nível mínimo possível. Entretanto, grande parte dos custos de um projeto de software é consumida pela atividade de testes, mesmo que não garanta que o software produzido está livre de falhas.

O presente trabalho buscou mostrar que testar software é uma das atividades que contribui com a melhoria da qualidade dos software produzidos. Considerando-se o paradigma OO, no qual a construção da ASE foi proposta, foi verificado que testar software não é uma tarefa das mais fáceis, exige conhecimentos técnicos sobre a aplicação por parte dos testadores.

Foram encontradas dificuldades para a criação dos casos de teste da ASE, pois a documentação carente das fases de desenvolvimento não propiciou a elaboração de um conjunto de testes com taxa de falhas elevada. Apesar deste fato não houve o impedimento da descoberta de falhas, já que ao executar os casos de teste do conjunto 1 foram encontradas falhas dos três níveis de severidade.

No conjunto 2 de casos de teste não foram encontradas falhas pois ao entrar com dados inválidos os métodos da classe testada se comportaram conforme o esperado reagindo às entradas incorretas.

Esse trabalho demonstrou, através da análise dos resultados, que é válida a aplicação de testes de software mesmo que o sistema esteja em uso pelo cliente, pois uma vez que as falhas encontradas sejam corrigidas a qualidade da ASE apresentará uma melhora da qualidade. A correção destas falhas poderá ser entregue ao cliente em forma de atualização de sistema.

Para trabalhos futuros, sugere-se a criação e execução de casos de testes unitários para as demais classes da ferramenta ASE, possibilitando assim a quantificação das falhas e o tipo de falha mais encontrada de acordo com a classificação das falhas. Sugere-se também documentar os casos de teste em ferramentas específicas como o Testlink, por exemplo.

REFERÊNCIAS BIBLIOGRÁFICAS

ABNT - ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBRISO/IEC9126-1 **Engenharia de software** - Qualidade de produto - Parte 1: Modelo de qualidade. Rio de Janeiro, 2003, 21 p.

ABNT - ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBRISO 9000:2000 **Sistema de Gestão de Qualidade** – Fundamentos e Vocabulário. Rio de Janeiro, 2000, 26 p.

BARTIÉ, A. **Garantia da Qualidade de Software**, São Paulo: Campus, 2002.

BINDER, R.V. **Testing object-oriented systems:** models, patterns, and tool; Addison-Wesley, 2006.

BRUNELLI, Marcos Valerio de Queiroz. **A utilização de uma metodologia de teste no processo de melhoria da qualidade de software**; Dissertação de Mestrado, Universidade Estadual de Campinas-UNICAMP, 2006.

CAETANO, Cristiano. **Severidade X Prioridade**: Aprenda a Classificar Corretamente um Defeito - Disponível em http://www.testexpert.com.br/?q=node/363. Acesso em: 14 set. 2010

DIAS-NETO, Arilo Claudio. **Introdução a Teste de Software.** Revista Engenharia de Software. Editora Devmedia, 2008. Disponível em http://www.devmedia.com.br/articles /viewcomp.asp?comp=8035. Acesso em: 15 ago. 2010.

GOETTENAUER, Daniel. **Defeito x Erro x Falha.** Qualidade Manaus: Teste de Software, Garantia da Qualidade e Engenharia de Software, 2009. Disponível em: http://qualidademanaus.wordpress.com/2009/08/14/defeito-x-erro-x-falha/. Acesso em: 10 dez. 2010

FERNANDES, C. T., LOPES, T. M. P. **Planejamento Integrado das Atividades de Codificação e Testes em Orientação a Objetos no Nível de Granularidade dos Métodos**, In: I-DEAS'2007 - X Workshop Iberoamericano de Ingenieria de Requisitos y Ambientes de Software, Isla de Margarita, 2007.

LIMA, G. M. P. S., TRAVASSOS, G. H. **Testes de Integração Aplicados a Software Orientado a Objetos,** Universidade Federal do Rio de Janeiro - UFRJ, 2004.

MARINHO, T. F**. Análise da Importância dos Testes de Software como Fator para Qualidade do Projeto**, Trabalho de Conclusão de Curso - Faculdade de Tecnologia da Zona Leste - SP, 2009.

MARTINS, J. C., TSCHANNER, H. L. **Testes de Software Aplicado à Orientação a Objetos**. Disponível em: http://www.batebyte.pr.gov.br/modules/conteudo/conteudo.php? conteudo=1100 >. Acesso em: 12 nov. 2009.

MASIERO, P. C., LEMOS, O. A. L., FERRARI F. C., MALDONADO J. C. **Teste de Software Orientado a Objetos e a Aspectos:** Teoria e Prática, 2006.

MELNIKOFF, S. S. S., **Teste de Software Orientado a Objetos.** EPUSP/PCS, 2009.

MOLINARI, Leonardo. **Teste de Software**: Produzindo Sistemas Melhores e Mais Confiáveis. 1. ed. São Paulo: Érica, 2003.

MOLINARI, Leonardo. **Testes Funcionais de Software**. 1.a ed: Visual Books, 2008.

MYERS, Glenford J., John Wiley & Sons, **The Art of Software Testing 2**. Nova Jersei, 2004

PEZZÈ, Mauro; YOUNG, Michal. **Teste e Análise de Softwàre**: processos, princípios e técnicas. Porto Alegre: Bookman, 2008.

PFLEEGER, S. L., **Engenharia de Software**: Teoria e Prática, Prentice Hall, 2a. Edição, 2004.

PRESSMAN, Roger S. **Engenharia de software**. 6.a ed. São Paulo: McGraw-Hill, 2006.

PRIMÃO, Aline Pacheco; RIBEIRO, Patric da Silva; KREUTZ, Diego Luís. **Estudo de caso:** Técnicas de Teste como Parte do Ciclo de Desenvolvimento, IX Simpósio de Informática da Região Centro/RS (SIRC/RS): Centro Universitário Franciscano – UNIFRA, 2010.

RECK, Wagner; GIACOMELLI, Joseane. **Alocação de Chaves Telecomandadas em Redes de Distribuição com Base no Histórico de Falhas**. II Simpósio Interno de Pesquisa e Extensão (SIPE): Universidade Federal do Pampa - Campus Alegrete, 2009.

RICARTE, I. L. M. **Teste de Software Orientado a Objetos**, In: Engenharia de Software II, DCA- FEEC UNICAMP, 2008.

ROCHA, A. R. C., MALDONADO, J. C., WEBER, K. C., **Qualidade de Software:** Teoria e Prática, Prentice Hall, 2001.

RUMBAUGH, J. **OMT Insight**. SIGS Books, 1996.

SOMMERVILLE, Ian. **Engenharia de Software**. . 6.a ed. São Paulo: Addison Wesley, 2007.

SPERANDIO, Mauricio, **Planejamento da Automação de Sistemas de Manobra em Redes de Distribuição**, Tese de Doutorado, Universidade Federal de Santa Catarina - UFSC, 2008.

TOZELLI, Paulo. **Solução QAWEB para teste de Software 2** - Tutorial Grátis - Disponível em:http://www.tutorialgratis.com.br/web/75-solucao-qawebr-para-teste-de-software-2. Acesso em: 13 out. 2010

VIEIRA, Vanessa Gindri. **Uma Estratégia para Testes de Regressão Utilizando Classes Testáveis**. Dissertação de Mestrado, Universidade Estadual de Campinas-UNICAMP, 2004.

ZIMMERMANN, Ana Paula. **Ferramenta para Automação de Testes Caixa-Preta.** Trabalho de Conclusão de Curso, Universidade do Vale do Itajaí, Itajaí, 2006.

APÊNDICE A

Este apêndice contém o diagrama de classes da ferramenta ASE, Figura 11. Nesse diagrama está destacada com um círculo escuro a classe MLE, principal classe do sistema, utilizada para os testes unitários.

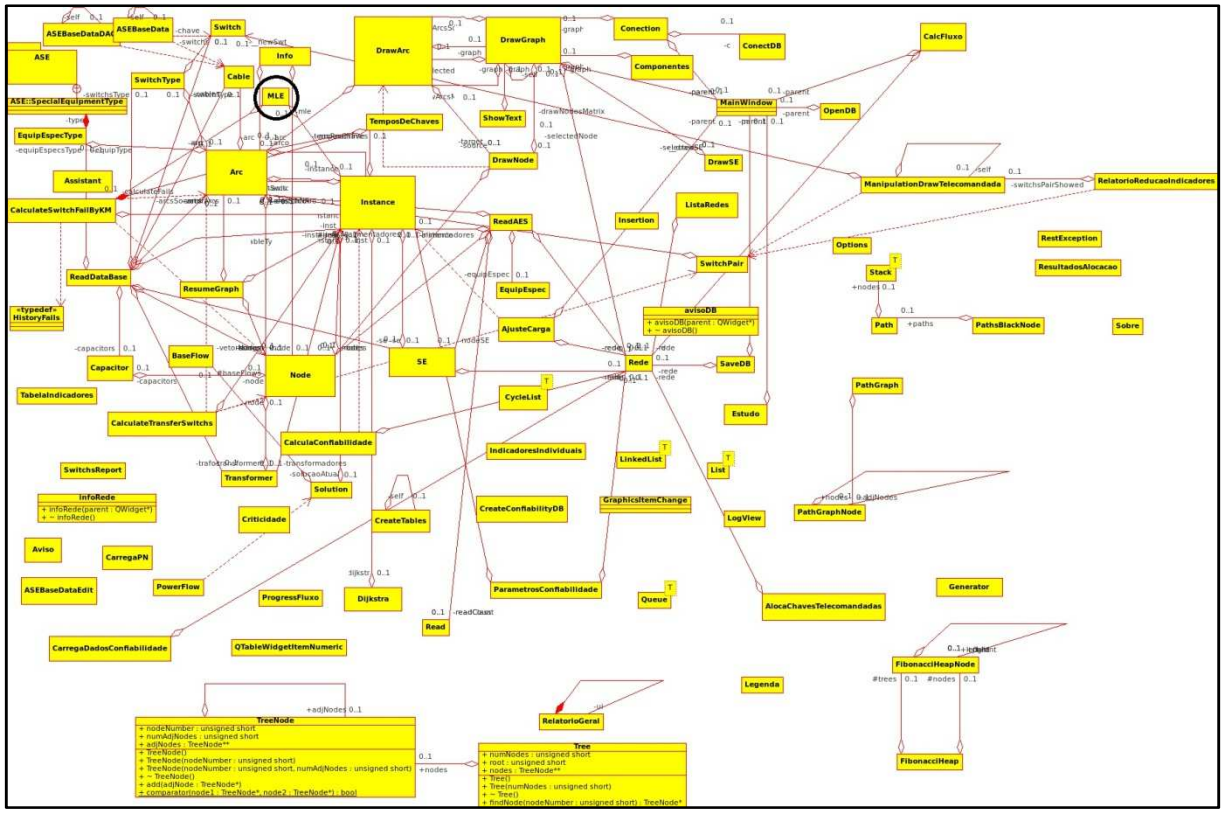

Figura 11: Diagrama de classes ASE

APÊNDICE B

Este apêndice contém os casos de teste do conjunto 1. Cada caso de teste contém o resultado obtido após a execução do teste e caso resultou em falha, a classificação da mesma foi realizada.

CASO DE TESTE 01	
Descrição:	Carregar uma rede para visualizar os equipamentos existentes
Pré-condições:	Ter uma rede redeTesteMLE.aes no memso diretório que o projeto ASE
Entrada:	File> Abrir Arquivo > redeTesteMLE.aes > Open
Resultado esperado:	Rede carregada em modo gráfico no centro da tela
Resultado do teste:	Rede carregada com SUCESSO

CASO DE TESTE 02	
Descrição:	Exibir a carga atual de um alimentador da rede
Pré-condições:	Ter a rede RedeTesteMLE.aes carregada na tela
Entrada:	Fluxo > Ajuste de carga > "Seleciona Alimentador" > Carrega alimentador
Resultado esperado:	Dados do alimentador completados na janela ajuste de carga
Resultado do teste:	Dados visualizados – SUCESSO

CASO DE TESTE 03	
Descrição:	Fecha a rede que está na tela
Pré-condições:	Ter uma rede carregada na tela pelo menu File > Abrir Arquivo
Entrada:	File > Close
Resultado esperado:	Tela central que continha a rede deve estar vazia
Resultado do teste:	Rede fechada com SUCESSO

CASO DE TESTE 04	
Descrição:	Sair do programa sem clicar no botão fechar no canto superior direito da tela
Pré-condições:	Estar com ASE aberta
Entrada:	File > Quit
Resultado esperado:	Programa fechado
Resultado do teste:	Programa fechado com SUCESSO

CASO DE TESTE 05	
Descrição:	Aumentar o zoom em uma rede
Pré-condições:	Estar com a rede redeTesteMLE.aes carregada
Entrada:	Três clicles no botão de zoom +
Resultado esperado:	Rede com uma visualização maior
Resultado do teste:	Desaparece alguns ramos da rede – FALHA
Classificação da falha:	ALTA

CASO DE TESTE 06	
Descrição:	Fechar uma rede aberta
Pré-condições:	Estar com a rede redeTesteMLE.aes carregada na tela
Entrada:	Arquivo > Fechar
Resultado esperado:	Frame Rede vazio e tela em branco
Resultado do teste:	Frame Rede vazio mas a rede ainda está visível na tela – FALHA
Classificação da falha:	MÉDIA

CASO DE TESTE 07	
Descrição:	Calcular fluxo de potência depois de fechar uma rede
Pré-condições:	Estar com a rede fechada pelo procedimento Arquivo > Fechar
Entrada:	Arquivo > Fechar
Resultado esperado:	Tela em branco ---- VER SE TEM MENSAGEM NO CÓDIGO
Resultado do teste:	Programa abortado – FALHA
Classificação da falha:	ALTA

CASO DE TESTE 08	
Descrição:	Ajustar a carga de um alimentador da rede de 112.26 Amperes para 112.9 Amperes através do método de correção Potência
Pré-condições:	Ter a rede RedeTesteMLE.aes carrregada na tela
Entrada:	Fluxo > Ajuste de carga > "Seleciona Alimentador" > Carrega alimentador > Altera
Resultado esperado:	Dados setados com sucesso
Resultado do teste:	Os campos ficam em branco mas refazendo a entrada o resultado está correto – FALHA
Classificação da falha:	MÉDIA

CASO DE TESTE 09	
Descrição:	Visualizar legenda das figuras encontradas na rede
Pré-condições:	Existir uma legenda com as figuras indicando o significado
Entrada:	da na tela
Resultado esperado:	Janela sobre o ASE com a legenda
Resultado do teste:	Legenda exibida na tela – SUCESSO

CASO DE TESTE 10	
Descrição:	Configuração da janela ao carregar uma rede
Pré-condições:	Ter uma rede para carregar
Entrada:	Arquivo > Abrir Arquivo > RedeTesteMLE.aes
Resultado esperado:	Janela configurada do tamanho da tela
Resultado do teste:	Janela desconfigurada no lado inferior da tela - FALHA
Classificação da falha:	BAIXA

CASO DE TESTE 11	
Descrição:	Fechar uma rede pelo botão FECHAR na barra de ferramentas
Pré-condições:	Ter uma rede carregada
Entrada:	Clique no botão FECHAR da barra de ferramentas
Resultado esperado:	Frame Rede vazio e tela em branco
Resultado do teste:	Frame Rede vazio e tela em branco – SUCESSO

CASO DE TESTE 12	
Descrição:	Abrir Arquivo com o comando Ctrl+O
Pré-condições:	Estar com o ASE aberto na tela
Entrada:	Ctrl+O
Resultado esperado:	Tela de Abrir Arquivo no centro da tela
Resultado do teste:	Tela de Abrir Arquivo no centro da tela – SUCESSO

CASO DE TESTE 13	
Descrição:	Ampliar modo de visualização da rede
Pré-condições:	Ter a rede redeTesteMLE.aes carregada
Entrada:	Ctrl++ - pelo menos uma vez
Resultado esperado:	Aumento de visualização da rede - Zoom +
Resultado do teste:	Aumento de visualização MAS perde partes dos ramos da rede- FALHA
Classificação da falha:	ALTA

CASO DE TESTE 14	
Descrição:	Reduzir modo de visualização da rede redeTesteMLE.aes
Pré-condições:	Ter com uma rede carregada
Entrada:	Ctrl+-
Resultado esperado:	Redução de visualização da rede - Zoom -
Resultado do teste:	Redução de visualização da rede – SUCESSO

CASO DE TESTE 15	
Descrição:	Rede em modo normal depois de zoom + ou zoom -
Pré-condições:	Ter uma rede no modo ampliado ou reduzido
Entrada:	F5
Resultado esperado:	Rede em tamanho normal no centro da tela
Resultado do teste:	Rede em tamanho normal no centro da tela – SUCESSO

CASO DE TESTE 16	
Descrição:	Exibe o resumo do relatório do fluxo de potência por alimentador - rede RedeTesteMLE.aes
Pré-condições:	Estar com o fluxo de potência da rede aberta calculado
Entrada:	Relatórios > Relatórios do Fluxo de Potência > Resumo por Alimenta-dor
Resultado esperado:	Janela RELATÓRIO GERAL carregada na tela com os dados nas co-lunas
Resultado do teste:	Relatório exibido na tela – SUCESSO

CASO DE TESTE 17	
Descrição:	Salvar o resumo do relatório do fluxo de potência por alimentador - rede RedeTesteMLE.aes - sem nomear o arquivo
Pré-condições:	Ter o relatório do resumo por alimentador na tela
Entrada:	Relatórios > Relatórios do Fluxo de Potência > Resumo por Alimentador > Salvar > Salvar
Resultado esperado:	Arquivo salvo no diretório que o usuário definir
Resultado do teste:	Mensagem pra sobrescrever o arquivo pois já existe neste diretório – FALHA
Classificação da falha:	ALTA

CASO DE TESTE 18	
Descrição:	Voltar ao modo edição depois de calcular o fluxo de potência
Pré-condições:	Ter a rede RedeTesteMLE.aes carregada na tela com o fluxo de potência calculado
Entrada:	Fluxo > Modo edição
Resultado esperado:	Rede no modo edição -estado anterior ao calcular o fluxo de potência
Resultado do teste:	Não é possível visualizar a frame com os dados de fluxo - SUCESSO

CASO DE TESTE 19	
Descrição:	Exibe o resumo do relatório do fluxo de potência por Carregamento dos trechos - rede RedeTesteMLE.aes
Pré-condições:	Estar com o fluxo de potência da rede aberta calculado
Entrada:	Relatórios > Relatórios do Fluxo de Potência > Relatório de Carregamento dos Trechos
Resultado esperado:	Janela RELATÓRIO GERAL carregada na tela com os dados nas colunas
Resultado do teste:	Relatório exibido na tela – SUCESSO

CASO DE TESTE 20	
Descrição:	Salvar o resumo do relatório do fluxo de potência por Carregamento dos Trechos - rede RedeTesteMLE.aes- sem nomear o arquivo
Pré-condições:	Ter o relatório do resumo por alimentador na tela
Entrada:	Relatórios > Relatórios do Fluxo de Potência > Relatório de Carregamento dos Trechos > Salvar > Salvar
Resultado esperado:	Arquivo salvo no diretório que o usuário definir
Resultado do teste:	Mensagem pra sobrescrever o arquivo pois já existe neste diretório – FALHA
Classificação da falha:	ALTA

CASO DE TESTE 21	
Descrição:	Exibe o resumo do relatório do fluxo de potência do Carregamento dos Trafos - rede RedeTesteMLE.aes
Pré-condições:	Estar com o fluxo de potência da rede aberta calculado
Entrada:	Relatórios > Relatórios do Fluxo de Potência > Relatório de Carregamento dos Trafos
Resultado esperado:	Janela RELATÓRIO GERAL carregada na tela com os dados nas colunas
Resultado do teste:	Relatório exibido na tela – SUCESSO

CASO DE TESTE 22	
Descrição:	Salvar o resumo do relatório do fluxo de potência do Carregamento dos Trafos - rede RedeTesteMLE.aes - sem nomear o arquivo
Pré-condições:	Ter o relatório do resumo por alimentador na tela
Entrada:	Relatórios > Relatórios do Fluxo de Potência > Relatório de Carregamento dos Trafos > Salvar > Salvar
Resultado esperado:	Arquivo salvo no diretório que o usuário definir
Resultado do teste:	Mensagem pra sobrescrever o arquivo pois já existe neste diretório – FALHA
Classificação da falha:	ALTA

CASO DE TESTE 23	
Descrição:	Exibe o resumo do relatório do fluxo de potência da Tensão dos Trafos - rede RedeTesteMLE.aes
Pré-condições:	Estar com o fluxo de potência da rede aberta calculado
Entrada:	Relatórios > Relatórios do Fluxo de Potência > Relatório de Tensão dos Trafos
Resultado esperado:	Janela RELATÓRIO GERAL carregada na tela com os dados nas colunas
Resultado do teste:	Relatório exibido na tela – SUCESSO

CASO DE TESTE 24	
Descrição:	Salvar o resumo do relatório do fluxo de potência da Tensão dos Trafos - rede RedeTesteMLE.aes- sem nomear o arquivo
Pré-condições:	Ter o relatório do resumo por alimentador na tela
Entrada:	Relatórios > Relatórios do Fluxo de Potência > Relatório de Tensão dos Trafos > Salvar > Salvar
Resultado esperado:	Arquivo salvo no diretório que o usuário definir
Resultado do teste:	Mensagem pra sobrescrever o arquivo pois já existe neste diretório – FALHA
Classificação da falha:	ALTA

CASO DE TESTE 25	
Descrição:	Buscar trafos EXISTENTES pelo campo busca da frame propriedades
Pré-condições:	Estar com a rede RedeTesteMLE.aes carregada e fluxo de potência calculados
Entrada:	Busca > TRAFO-8 > Enter
Resultado esperado:	As informações do TRAFO-8 devem estar carregadas na frame propriedades
Resultado do teste:	Informações do TRAFO-8 carregadas e calculos realizados com sucesso – SUCESSO

CASO DE TESTE 26	
Descrição:	Buscar trafos INEXISTENTES pelo campo busca da frame propriedades
Pré-condições:	Estar com a rede RedeTesteMLE.aes carregada e fluxo de potência calculados
Entrada:	Busca > TRAFO-15 > Enter
Resultado esperado:	As informações do TRAFO-15 devem permanecer vazias na frame propriedades
Resultado do teste:	Informações do TRAFO-15 vazias ba frame Propriedades - SUCESSO

CASO DE TESTE 27	
Descrição:	Buscar trafos EXISTENTES pelo campo busca da frame propriedades sem ter uma rede carregada
Pré-condições:	Estar com a frame Redes e a frame central vazias
Entrada:	Busca > TRAFO-8 > Enter
Resultado esperado:	As informações do TRAFO-8 Não devem estar carregadas na frame Propriedades
Resultado do teste:	Informações do TRAFO-8 na frame Propriedades vazia - SUCESSO

CASO DE TESTE 28	
Descrição:	Alterar o valor de demanda ativa (P (KW)) na frame propriedades
Pré-condições:	Buscar por um transformador TRAFO-? presente na rede
Entrada:	Fase E coluna P(KW) altera de 100 para 200 > Salvar
Resultado esperado:	Total igual a 400
Resultado do teste:	Campo Total não é alterado automaticamente - FALHA
Classificação da falha:	MÉDIA

CASO DE TESTE 29	
Descrição:	Alterar o valor de demanda ativa (P (KW)) na frame propriedades
Pré-condições:	Buscar por um transformador TRAFO-? presente na rede
Entrada:	Fase E coluna P(KW) altera de 100 para 200 > Salvar > (Fazer nova pesquisa pelo mesmo trafo)
Resultado esperado:	Total igual a 400
Resultado do teste:	Campo Total alterado corretamente - SUCESSO

CASO DE TESTE 30	
Descrição:	Zoom com o botão direito do mouse
Pré-condições:	Ter a rede SE_OUTR-METR-CIN.TXT carregada na tela
Entrada:	Pressionar o botão esquerdo do mouse a arrastar na parte da rede que deseja uma visualização maior
Resultado esperado:	Parte da rede com uma visualização maior
Resultado do teste:	Parte da rede selecionada possui visualização maior - SUCESSO

CASO DE TESTE 31	
Descrição:	Selecionar os componentes de uma rede na Frame Lista de Compo-nentes
Pré-condições:	Estar com a rede SE_OUTR-METR-CIN.TXT carregada
Entrada:	Selecionar o componente SUBESTAÇÃO
Resultado esperado:	Ícone da SUBESTAÇÃO disponível sobre a rede na tela
Resultado do teste:	Ícone disponível – SUCESSO

CASO DE TESTE 32	
Descrição:	Selecionar os componentes TRANSFORMADORES de uma rede na Frame Lista de Componentes
Pré-condições:	Estar com a rede SE_OUTR-METR-CIN.TXT carregada
Entrada:	Selecionar o componente TRANSFORMADORES
Resultado esperado:	Ícones de TRANSFORMADORES disponíveis sobre a rede na tela
Resultado do teste:	Ícones disponíveis – SUCESSO

CASO DE TESTE 33	
Descrição:	Selecionar os componentes CAPACITORES de uma rede na Frame Lista de Componentes
Pré-condições:	Estar com a rede SE_OUTR-METR-CIN.TXT carregada
Entrada:	Selecionar o componente CAPACITORES
Resultado esperado:	Ícones de CAPACITORES disponíveis sobre a rede na tela
Resultado do teste:	Ícones disponíveis – SUCESSO

CASO DE TESTE 34	
Descrição:	Visualizar apenas bairros ou regiões de um rede
Pré-condições:	Estar com a rede SE_OUTR-METR-CIN.TXT carregada
Entrada:	Deselecionar o bairro na frame REDES
Resultado esperado:	Região nomeada de CIN - 8 TABAI sem visualização na tela.
Resultado do teste:	Região sem vizualização na tela - SUCESSO

CASO DE TESTE 35	
Descrição:	Deselecionar os componentes TRECHOS (Cabos) de uma rede na Frame Lista de Componentes
Pré-condições:	Estar com a rede SE_OUTR-METR-CIN.TXT carregada
Entrada:	Selecionar o componente TRECHOS (Cabos) - CARREGA COMO PADRÃO
Resultado esperado:	Ícones de TRECHOS (Cabos) disponível sobre a rede na tela
Resultado do teste:	Ícones não disponíveis – SUCESSO

CASO DE TESTE 36	
Descrição:	Selecionar os componentes EQUIPAMENTOS ESPECIAIS de uma rede na Frame Lista de Componentes
Pré-condições:	Estar com a rede SE_OUTR-METR-CIN.TXT carregada
Entrada:	Selecionar o componente EQUIPAMENTOS ESPECIAIS
Resultado esperado:	Ícones de EQUIPAMENTOS ESPECIAIS disponível sobre a rede na tela
Resultado do teste:	Ícones disponíveis – SUCESSO

CASO DE TESTE 37	
Descrição:	Selecionar os componentes CHAVES de uma rede na Frame Lista de Componentes
Pré-condições:	Estar com a rede SE_OUTR-METR-CIN.TXT carregada
Entrada:	Selecionar o componente CHAVES
Resultado esperado:	Ícones de CHAVES disponível sobre a rede na tela
Resultado do teste:	Ícones disponíveis – SUCESSO

CASO DE TESTE 38	
Descrição:	Minimizar a janela da Legenda
Pré-condições:	Ter a janela Legenda exibida na tela
Entrada:	Legenda > Ver Legendas > Clique no botão Minimizar no canto direito superior
Resultado esperado:	Janela Legenda Minimizada e o restante sendo exibido normalmente
Resultado do teste:	Todo o programa é minimizado - FALHA
Classificação da falha:	ALTA

CASO DE TESTE 39	
Descrição:	Salvar o resumo do relatório do fluxo de potência da Tensão dos Trafos - rede SE_OUTR-METR-CIN.TXT
Pré-condições:	Ter o relatório do resumo por alimentador na tela
Entrada:	Relatórios > Relatórios do Fluxo de Potência > Relatório de Tensão dos Trafos > Salvar
Resultado esperado:	Arquivo salvo no diretório que o usuário definir
Resultado do teste:	Arquivo salvo no diretório que o usuário definiu - SUCESSO

CASO DE TESTE 40	
Descrição:	Calcular o fluxo de potência setando o valor do Limite Crítico de Tensão para 97%
Pré-condições:	Ter a rede SE_OUTR-METR-CIN.TXT carregada
Entrada:	Fluxo > Calcula fluxo > (Limite Crítico de Tensão seta para 97%)
Resultado esperado:	Parte dos transformadores da rede ficarão em vermelho porque o limite crítico de tensão foi ultrapassado
Resultado do teste:	Transformadores do centro da cidade em vermelho - SUCESSO

CASO DE TESTE 41	
Descrição:	Calcular o fluxo de potência setando o valor do Limite Crítico de Tensão para 97% e Limite Precário de Tensão para 100%
Pré-condições:	Ter a rede SE_OUTR-METR-CIN.TXT carregada
Entrada:	Fluxo > Calcula fluxo > (Limite Crítico de Tensão seta para 97% e Limite Precário de Tensão para 100%)
Resultado esperado:	Todos os transformadores da rede ficarão em vermelho porque o Limite Crítico e o Precário de tensão foram ultrapassados
Resultado do teste:	Transformadores em vermelho na tela - SUCESSO

CASO DE TESTE 42	
Descrição:	Ocultar a visualização das frames na tela principal : Lista de Componentes, Redes, Dados do Fluxo, Propriedades, Arquivo
Pré-condições:	Ter o ASE carregado na tela
Entrada:	Clique com o botão direito do mouse na barra de menu e/ou de ferramentas e selecionar
Resultado esperado:	Frames ocultadas da tela principal
Resultado do teste:	Frames ocultadas da tela principal - SUCESSO

CASO DE TESTE 43	
Descrição:	Ocultar a visualização da frame Console
Pré-condições:	Ter o ASE carregado na tela
Entrada:	Clique com o botão direito do mouse na barra de menu e/ou de ferramentas e selecionar Console
Resultado esperado:	Frame Console ocultada da tela principal
Resultado do teste:	Não mostra nada – FALHA
Classificação da falha:	ALTA

CASO DE TESTE 44	
Descrição:	Exibe o resumo do relatório do fluxo de potência por alimentador - rede SE_OUTR-METR-CIN.TXT
Pré-condições:	Estar SEM O FLUXO DE POTÊNCIA da rede aberta calculado
Entrada:	Relatórios > Relatórios do Fluxo de Potência > Resumo por Alimentador
Resultado esperado:	Janela RELATÓRIO GERAL carregada na tela com algumas colunas completas
Resultado do teste:	Relatório exibido somente com as colunas Alimentador e Tensão no Final da Rede completos – SUCESSO

CASO DE TESTE 45	
Descrição:	Salvar o resumo do relatório do fluxo de potência por alimentador - SE_OUTR-METR-CIN.TXT
Pré-condições:	Ter o relatório do resumo por alimentador na tela
Entrada:	Relatórios > Relatórios do Fluxo de Potência > Resumo por Alimentador > Salvar > "nome" > Salvar
Resultado esperado:	Arquivo salvo no diretório que o usuário definir com extenção .csv
Resultado do teste:	Arquivo salvo em formato .txt – FALHA
Classificação da falha:	MÉDIA

CASO DE TESTE 46	
Descrição:	Exibe o resumo do relatório do fluxo de potência por Carregamento dos trechos - SE_OUTR-METR-CIN.TXT
Pré-condições:	Estar SEM o fluxo de potência da rede aberta calculado
Entrada:	Relatórios > Relatórios do Fluxo de Potência > Relatório de Carregamento dos Trechos
Resultado esperado:	Janela RELATÓRIO GERAL carregada na tela com algumas colunas completas
Resultado do teste:	Relatório exibido na tela – SUCESSO

CASO DE TESTE 47	
Descrição:	Salvar o resumo do relatório do fluxo de potência por Carregamento dos Trechos - rede SE_OUTR-METR-CIN.TXT
Pré-condições:	Ter o relatório do resumo por alimentador na tela
Entrada:	Relatórios > Relatórios do Fluxo de Potência > Relatório de Carregamento dos Trechos > Salvar > "nome" > Salvar
Resultado esperado:	Arquivo salvo no diretório que o usuário definir com extenção .csv
Resultado do teste:	Arquivo salvo em formato .txt – FALHA
Classificação da falha:	MÉDIA

CASO DE TESTE 48	
Descrição:	Exibe o resumo do relatório do fluxo de potência do Carregamento dos Trafos - rede SE_OUTR-METR-CIN.TXT
Pré-condições:	Estar SEM o fluxo de potência da rede aberta calculado
Entrada:	Relatórios > Relatórios do Fluxo de Potência > Relatório de Carregamento dos Trafos
Resultado esperado:	Janela RELATÓRIO GERAL carregada na tela com os dados nas colunas
Resultado do teste:	Relatório exibido na tela – SUCESSO

CASO DE TESTE 49	
Descrição:	Salvar o resumo do relatório do fluxo de potência do Carregamento dos Trafos - rede SE_OUTR-METR-CIN.TXT
Pré-condições:	Ter o relatório do resumo por alimentador na tela
Entrada:	Relatórios > Relatórios do Fluxo de Potência > Relatório de Carregamento dos Trafos > Salvar > "nome" > Salvar
Resultado esperado:	Arquivo salvo no diretório que o usuário definir com extenção .csv
Resultado do teste:	Arquivo salvo em formato .txt – FALHA
Classificação da falha:	MÉDIA

CASO DE TESTE 50	
Descrição:	Exibe o resumo do relatório do fluxo de potência da Tensão dos Trafos - rede SE_OUTR-METR-CIN.TXT
Pré-condições:	Estar SEM o fluxo de potência da rede aberta calculado
Entrada:	Relatórios > Relatórios do Fluxo de Potência > Relatório de Tensão dos Trafos
Resultado esperado:	Janela RELATÓRIO GERAL carregada na tela com os dados nas colunas
Resultado do teste:	Relatório exibido na tela – SUCESSO

CASO DE TESTE 51	
Descrição:	Salvar o resumo do relatório do fluxo de potência da Tensão dos Trafos - rede SE_OUTR-METR-CIN.TXT
Pré-condições:	Ter o relatório do resumo por alimentador na tela
Entrada:	Relatórios > Relatórios do Fluxo de Potência > Relatório de Tensão dos Trafos > Salvar > "Nome" > Salvar
Resultado esperado:	Arquivo salvo no diretório que o usuário definir com extenção .csv
Resultado do teste:	Arquivo salvo em formato .txt – FALHA
Classificação da falha:	MÉDIA

CASO DE TESTE 52	
Descrição:	Desmarcar os componentes de uma rede ma frame Lista de Componentes
Pré-condições:	Ter uma rede SE_OUTR-METR-CIN.TXT carregada na tela
Entrada:	Frame Lista de Componentes - desmarcar com um clique todos os componentes
Resultado esperado:	Tela em branco com rede oculta
Resultado do teste:	Rede oculta na tela – SUCESSO

CASO DE TESTE 53	
Descrição:	Mudar o tamanho de visualização do ícone de Subestação
Pré-condições:	Ter uma subestação na rede carregada na tela
Entrada:	Frame Lista de Componentes - clique no + > marca GRANDE com um clique
Resultado esperado:	Ícone maior que o do estado anterior e a rede no mesmo estado - sem mudar visualização na rede
Resultado do teste:	Ícone não mudou em tempo real de tamanho - FALHA
Classificação da falha:	BAIXA

CASO DE TESTE 54	
Descrição:	Mudar o tamanho de visualização do ícone de Subestação
Pré-condições:	Ter uma subestação na rede carregada na tela
Entrada:	Frame Lista de Componentes - clique no + > marca GRANDE com um clique > F5
Resultado esperado:	Ícone maior que o do estado anterior e a rede no mesmo estado
Resultado do teste:	Mudou de tamanho mas interferindo na rede - SUCESSO

CASO DE TESTE 55	
Descrição:	Calcular a confiabilidade da rede SE_OUTR-METR-CIN.TXT
Pré-condições:	Ter o fluxo de potência calculado
Entrada:	Confiabilidade > Calcular Confiabilidade > OK
Resultado esperado:	Tabela - Indicadores Coletivos Devido aos Desarmes na Média Tensão preenchida e carregada na tela
Resultado do teste:	SUCESSO

CASO DE TESTE 56	
Descrição:	Salvar o relatório Indicadores Coletivos Devido aos Desarmes na Média Tensão
Pré-condições:	Ter o relatório calculado na tela
Entrada:	Indicadores Coletivos Devido aos Desarmes na Média Tensão > "Nome" > Salvar
Resultado esperado:	Relatório salvo no local que o usuário definir com extensão.csv
Resultado do teste:	Relatório salvo com extensão.txt - FALHA
Classificação da falha:	BAIXA

CASO DE TESTE 57	
Descrição:	Fechar o relatório Indicadores Coletivos Devido aos Desarmes na Média Tensão
Pré-condições:	Ter o relatório calculado na tela
Entrada:	Indicadores Coletivos Devido aos Desarmes na Média Tensão > Fechar ou botão direito no canto superior da janela
Resultado esperado:	Relatório fechado
Resultado do teste:	SUCESSO

CASO DE TESTE 58	
Descrição:	Minimizar o relatório Indicadores Coletivos Devido aos Desarmes na Média Tensão
Pré-condições:	COM o relatório na tela
Entrada:	Indicadores Coletivos Devido aos Desarmes na Média Tensão > Clique no botão MINIMIZAR no canto superior direito da janela
Resultado esperado:	Relatório minimizado dentro da área de visualização da ASE
Resultado do teste:	Todo o programa é minimizado - FALHA
Classificação da falha:	ALTA

CASO DE TESTE 59	
Descrição:	Minimizar o relatório do fluxo de potência da Tensão dos Trafos - rede SE_OUTR-METR-CIN.TXT
Pré-condições:	Ter o relatório do resumo por alimentador na tela
Entrada:	Relatório de Tensão dos Trafos > Clique no botão MINIMIZAR no canto superior direito da janela
Resultado esperado:	Relatório minimizado dentro da área de visualização da ASE
Resultado do teste:	Todo o programa é minimizado - FALHA
Classificação da falha:	ALTA

CASO DE TESTE 60	
Descrição:	Calcular a confiabilidade da rede SE_OUTR-METR-CIN.TXT
Pré-condições:	SEM o fluxo de potência calculado
Entrada:	Confiabilidade > Calcular Confiabilidade > OK
Resultado esperado:	Tabela - Indicadores Coletivos Devido aos Desarmes na Média Tensão preenchida e carregada na tela
Resultado do teste:	SUCESSO

CASO DE TESTE 61	
Descrição:	Alocar chaves telecomandadas da rede SE_OUTR-METR-CIN.TXT
Pré-condições:	Ter o fluxo de potência e a confiabilidade calculados
Entrada:	Confiabilidade > Alocação Ótima de Chaves > Alocar
Resultado esperado:	Ranking com as chaves candidatas sendo exibido em forma de tabela na ordem de maior ganho
Resultado do teste:	Ranking exibido – SUCESSO

CASO DE TESTE 62	
Descrição:	Mostrar chaves candidatas a telecomandadas na rede SE_OUTR-METR-CIN.TXT
Pré-condições:	Ter alocado as chaves telecomandadas
Entrada:	Ranking com as chaves ótimas > Mostrar
Resultado esperado:	Rede de testes com as chaves telecomandadas assinaladas com o ícone correspondente
Resultado do teste:	Rede de testes com as chaves telecomandadas carregadas - SUCESSO

CASO DE TESTE 63	
Descrição:	Visualizar as propriedades de um transformador CAN-1839 da rede SE_OUTR-METR-CIN.TXT
Pré-condições:	Ter a rede de testes carregada na tela
Entrada:	Clique duplo em cima do transformador
Resultado esperado:	Frame propriedades com os campos preenchidos com informações do transformador
Resultado do teste:	SUCESSO

CASO DE TESTE 64	
Descrição:	Fechar a rede RedeTesteMLE.aes a abrir a rede SE_OUTR-METR-CIN.TXT sem inicializar o sistema
Pré-condições:	Ter duas redes de testes, uma carregada e outra disponível
Entrada:	Clique no no botão fechar da barra de ferramentas > Ctrl +O > SE_OUTR-METR-CIN.TXT > Abrir
Resultado esperado:	Rede RedeTesteMLE.aes apagada e a rede SE_OUTR-METR-CIN.TXT carregada na tela
Resultado do teste:	Fica partes da rede 1 e a segunda não aparece só com F5 – FALHA
Classificação da falha:	MÉDIA

CASO DE TESTE 65	
Descrição:	Cancelar o processo de alocação chaves telecomandadas da rede SE_OUTR-METR-CIN.TXT
Pré-condições:	Ter o fluxo de potência e a confiabilidade calculados
Entrada:	Confiabilidade > Alocação Ótima de Chaves > Alocar > Cancelar
Resultado esperado:	Alocação abortada
Resultado do teste:	Continua processando – FALHA
Classificação da falha:	ALTA

CASO DE TESTE 66	
Descrição:	Alocar chaves telecomandadas sem calcular o fluxo de potência da rede SE_OUTR-METR-CIN.TXT
Pré-condições:	Ter uma rede carregada sem calcular o fluxo de potência
Entrada:	Confiabilidade > Alocação Ótima de Chaves > Alocar
Resultado esperado:	Chaves alocadas com sucesso
Resultado do teste:	SUCESSO

CASO DE TESTE 67	
Descrição:	Ajustar a carga de um alimentador da rede de 112.26 Amperes para 112.9 Amperes através do método de correção Corrente (A)
Pré-condições:	Ter a rede RedeTesteMLE.aes carrregada na tela
Entrada:	Fluxo > Ajuste de carga > "Seleciona Alimentador" > Carrega alimentador > Altera
Resultado esperado:	Dados setados com sucesso
Resultado do teste:	Os campos ficam em branco mas refazendo a entrada o resultado está correto – FALHA
Classificação da falha:	MÉDIA

APÊNDICE C

Este apêndice contém os casos de teste do conjunto 2. Cada caso de teste contém o resultado obtido após a execução do teste unitário e caso resultou em falha, a classificação da mesma foi realizada.

CASO DE TESTE 01	
Descrição:	Construção da matriz MLE inicial
Pré-condições:	Ter a rede de teste RedeTesteMLE.aes carregada
Entrada:	Construção da matriz MLE com os tempos diferentes de 0.0 (tempo de isolamento, transferencia)
Resultado esperado:	Dados da matriz inicial setado com os valores diferentes de 0.0
Resultado do teste:	Dados da matriz inicial setados – SUCESSO

CASO DE TESTE 02	
Descrição:	Criar a matriz MLE
Pré-condições:	Ter uma MLE com os valores setados (diferentes de 0.0)
Entrada:	Compara os tempos da nova MLE com os da matriz original (tempos 0.0), ou seja, os tempos devem ser diferentes de 0.0
Resultado esperado:	MLE completada com os valores diferentes de 0.0
Resultado do teste:	SUCESSO

CASO DE TESTE 03	
Descrição:	Salva a matriz MLE
Pré-condições:	Ter uma MLE original
Entrada:	Salvar MLE original e verificar se não sofreu alterações
Resultado esperado:	MLE não alterada
Resultado do teste:	SUCESSO

CASO DE TESTE 04	
Descrição:	CASO DE TESTE DO POP
Pré-condições:	Ter uma matriz MLE original
Entrada:	Salva uma vez a MLE e restaura, depois compara com a matriz original
Resultado esperado:	Matriz MLE não sofreu alterações
Resultado do teste:	SUCESSO

CASO DE TESTE 05	
Descrição:	Restaura matriz MLE
Pré-condições:	Ter uma MLE original
Entrada:	Salvar a matriz original 3 vezes, restaurar e verificar se não sofreu alterações
Resultado esperado:	MLE não alterada
Resultado do teste:	SUCESSO

CASO DE TESTE 06	
Descrição:	Alterar a matriz MLE com tempos(isolamento, transferência ou reparo) válidos
Pré-condições:	Ter uma nova MLE com os tempos setados
Entrada:	Inserir novos tempos na MLE e verificar se foram alterados com sucesso
Resultado esperado:	Tempos alterados
Resultado do teste:	SUCESSO

CASO DE TESTE 07	
Descrição:	Comparar a MLE original com MLE com tempos setados
Pré-condições:	Ter a MLE original e outra setada
Entrada:	Comparar se a precisão da matriz criada com a original, se é maior que 0.001 ultrapassou o limite
Resultado esperado:	Precisão menor que 0.001
Resultado do teste:	SUCESSO

CASO DE TESTE 08	
Descrição:	Imprimir matriz MLE na tela
Pré-condições:	Ter uma MLE com tempos válidos
Entrada:	Verificar se existem chaves na rede, construir a matriz e imprimir os valores encontrados
Resultado esperado:	Tempos da MLE impressos na tela
Resultado do teste:	SUCESSO

CASO DE TESTE 09	
Descrição:	Calcular o DMIC da matriz MLE
Pré-condições:	Ter uma MLE com valores válidos
Entrada:	Taxa de falhas multiplicada pelos tempos de cada transformador, o maior valor é o DMIC ou é o maior DIC da MLE
Resultado esperado:	DMIC=2.8326
Resultado do teste:	SUCESSO

CASO DE TESTE 10	
Descrição:	Calcular o DIC de uma rede
Pré-condições:	Ter uma MLE com valores válidos
Entrada:	Taxa de falhas multiplicada pelos tempos de cada transformador
Resultado esperado:	[0]= 2.9326 [1]= 2.9326 [2]= 2.9992 [3]= 2.9992 [4]= 2.9992 [5]= 5.8318 [6]= 5.8318 [7]= 5.7652
Resultado do teste:	SUCESSO

CASO DE TESTE 11	
Descrição:	Calcular o número de consumidores da rede
Pré-condições:	Ter uma MLE com valores válidos
Entrada:	Soma o número de clientes de cada transformador da rede
Resultado esperado:	Número de consumidores= 800 pois cada um dos 8 transformadorores tem 100 clientes
Resultado do teste:	SUCESSO

CASO DE TESTE 12	
Descrição:	Calcular o DEC de uma rede
Pré-condições:	Ter uma MLE com valores válidos
Entrada:	Calcula o DIC multiplicado pelo número de consumidor por transformador e divide pelo total de consumidores
Resultado esperado:	DEC= 4.03645
Resultado do teste:	SUCESSO

CASO DE TESTE 13	
Descrição:	Calcular o ENS de uma rede
Pré-condições:	Ter uma MLE com valores válidos
Entrada:	Calcular o DIC, multiplicar pela potência (as 3 fases= 100+100+100 = 300 consumidores) de cada transformador e divide por 1000
Resultado esperado:	ENS=9.68748
Resultado do teste:	SUCESSO

CASO DE TESTE 14	
Descrição:	Calcular o FEC de uma rede
Pré-condições:	Ter uma MLE com valores válidos
Entrada:	Calcular a taxa de falhas (taxa=2) multiplicado pelo número de consumidores
Resultado esperado:	FEC=4.75
Resultado do teste:	SUCESSO

CASO DE TESTE 15	
Descrição:	Produzir uma falha na MLE, setando um tempo pra não válido
Pré-condições:	Ter uma MLE com valores válidos para setar somente um valor pra não valido
Entrada:	Setando o tempo válido matrix[0][1] = 0.05 para o não válido matrix[0][1] = 0.55
Resultado esperado:	MLE não se comporta como esperado por dados não compatíveis
Resultado do teste:	SUCESSO

CASO DE TESTE 16	
Descrição:	Calcular o DMIC para trafo não exitente na rede
Pré-condições:	Ter uma MLE com valores válidos
Entrada:	Calcular o DMIC para o transformador 9 sendo que na matriz só existem 8
Resultado esperado:	MLE não se comporta como esperado por dados não compatíveis
Resultado do teste:	SUCESSO

CASO DE TESTE 17	
Descrição:	Calcular o FIC
Pré-condições:	Ter uma MLE com valores válidos
Entrada:	Calcular a soma dos tempo de falha da rede
Resultado esperado:	[0]=4 [1]=4 [2]=4 [3]=4 [4]=4 [5]=6 [6]=6 [7]=6
Resultado do teste:	SUCESSO

CASO DE TESTE 18	
Descrição:	Limpar os casos de teste da memória
Pré-condições:	Ter alguma matriz não apagada
Entrada:	Apagar a matriz dos casos de teste para não ficarem na memória
Resultado esperado:	Memória livre
Resultado do teste:	SUCESSO