Universidade Tecnológica Federal do Paraná - UTFPR
Campus Ponta Grossa - Paraná - Brasil
ISSN 1808-0448 / v. 09, n. 02: p. 286-306, 2013
D.O.I: 10.3895/S1808-04482013000200001

Revista Gestão Industrial

MÉTODOS DE TESTES APLICADOS A SOFTWARES DE CONTROLE DO PROCESSO INDUSTRIAL DE EXTRAÇÃO DE ÓLEO DE SOJA POR SOLVENTE

METHODS OF APPLIED TESTS TO SOFTWARES OF CONTROL OF THE INDUSTRIAL PROCESS OF EXTRACTION OF OIL OF SOY FOR SOLVENT

Deise Regina Portal[1]; Nilo Otani[2]
[1]Escola Superior de Criciúma – ESUCRI – Criciúma – Brasil
deiseportal@hotmail.com
[2]Universidade Federal de Santa Catarina – EGC/UFSC – Florianópolis – Brasil
ni_otani@yahoo.com

Resumo

O objetivo desta pesquisa é caracterizar os métodos de teste que podem ser empregados nos testes de software de controle industrial para extração de óleo de soja por solvente. Para alcançar tal objetivo serve-se da fundamentação teórica no escopo do processo industrial de extração de soja por solvente, dos softwares de controle industrial empregados e dos métodos de teste de software. Quanto aos procedimentos metodológicos, trata-se de uma pesquisa exploratória e descritiva, em relação à abordagem do problema é qualitativa. A coleta de dados ocorre por meio de pesquisas de documentação indireta em fontes de dados secundários. Os resultados da pesquisa apontam que todos os métodos convencionais de teste de software são aplicados também aos softwares de controle industrial, o que difere é o modo como os testes devem ser conduzidos. Os testes de software de controle industrial são conduzidos com mais eficiência a partir da etapa de teste de sistema, devido à natureza destes softwares. Além dos testes convencionais a pesquisa também aponta que, devido à periculosidade do solvente empregado no processo de extração de óleo, os testes críticos são absolutamente necessários para a garantia de segurança, qualidade e confiabilidade dos softwares.

Palavras chave: processos industriais; teste de software; extração de óleo de soja.

1. Introdução

O método de extração por solvente, por sua eficiência, é amplamente adotado na indústria de extração do óleo de oleaginosas, como a soja. Oetterer, Regitano-d'Arce e Spoto (2006) argumentam que, a vantagem no emprego do solvente está em garantir o completo desengorduramento do grão, independente do teor de óleo inicial da massa de grãos.

O solvente atualmente utilizado é o hexano, um derivado do processo de refino de petróleo, segundo Mandarino e Roessing (2001). Este solvente é utilizado por satisfazer a alguns requisitos como "ser totalmente apolar e dissolver prontamente o óleo, ser miscível com água, ter baixo calor latente de ebulição, não atacar as canalizações e os aparelhos com os quais entra em contato" (OETTERER; REGITANO-D'ARCE; SPOTO, 2006, p. 319).

Segundo Feijó (2007), a partir da década de 1990, com os avanços tecnológicos e a disponibilização dos dados de produção, a automação industrial passou a integrar as várias etapas do negócio, desde o chão de fábrica aos sistemas corporativos. As plantas industriais de extração de soja por solvente fazem o uso de softwares de controle industrial.

O presente artigo tem como foco de pesquisa os testes de softwares de controle industrial de supervisão e os softwares dos CLPs, por terem funções vitais em um processo automatizado, segundo Andrade (2007).

A atividade de teste de software é crucial para a garantia de qualidade de software e seu objetivo é descobrir sistematicamente diferentes categorias de erros em menor tempo e esforço. Os erros podem surgir desde o inicio do projeto e quanto mais tardiamente forem detectados maior será o custo de correção. Segundo Pressman (1995), o custo por correção pode ser até 100 vezes menor se a correção for feita ainda na fase de desenvolvimento.

Com base nos aspectos descritos, o objetivo deste trabalho é apontar quais os métodos de teste de software podem ser aplicados aos softwares de controle industrial e de que modo os testes podem ser conduzidos com eficiência para a garantida de qualidade e confiabilidade dos mesmos.

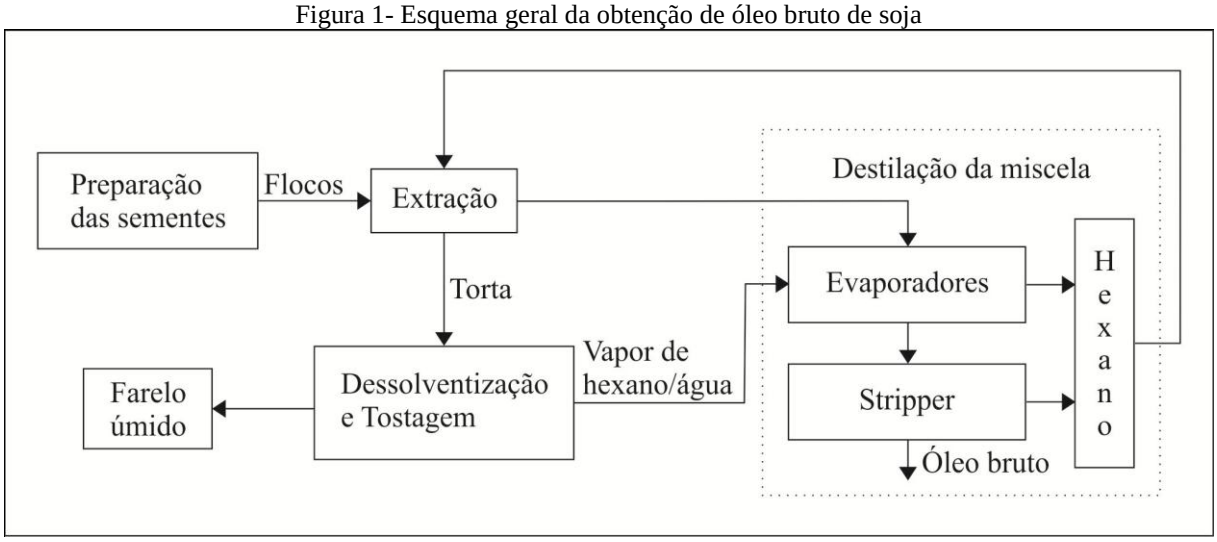
2. O processo de extração de óleo de soja por solvente

Este capítulo contextualiza o processo de extração de óleo de soja por solvente. O escopo teórico possui as bases de Carvalho, Kirschnik, Paiva e Aiura (2002), Custódio (2003), Mandarino e Roessing (2001), Mizubuti e Ida (1999), Oetterer, Regitano-d'Arce e Spoto (2006), Paraíso (2001), PETROBRAS (2012).

O método de extração por solvente, por sua eficiência, é amplamente adotado na indústria de extração do óleo de oleaginosas, como a soja. Oetterer, Regitano-d'Arce e Spoto (2006) argumentam que, a vantagem no emprego do solvente está em garantir o completo desengorduramento do grão, independente do teor de óleo inicial da massa de grãos.

Segundo Paraíso (2001), o processo de obtenção de óleo de soja bruto é uma parte importante na indústria de óleo de soja comestível e se divide em três grandes etapas: preparação das sementes, extração do óleo e recuperação do solvente. A Figura 1 demonstra o esquema geral do processo de extração do óleo de soja.

Figura 1- Esquema geral da obtenção de óleo bruto de soja



Fonte: Paraíso (2001)

2.1 A preparação das sementes de soja

Conforme Paraíso (2001), a preparação das sementes de soja consiste em um conjunto de etapas apropriadas para melhorar o rendimento na operação de extração. Algumas operações comuns do processo de preparação dos grãos de soja, segundo Custódio (2003), são:

– Pré-Limpeza: a eliminação de impurezas, como grãos quebrados, avariados e ardidos (grãos ou pedaços de grãos que se apresentam visivelmente fermentados em sua totalidade e com coloração marrom escura acentuada, afetando o cotilédone, segundo Instrução Normativa 37/2007/MAPA), é denominada pré-limpeza, que é realizada através de máquinas apropriadas dotadas de peneiras vibratórias ou de outros dispositivos, capazes de separar os grãos dos contaminantes maiores (MANDARINO; ROESSING, 2001);

– Secagem: para aumentar a eficiência no descasque, os grãos passam por secagem até atingirem um teor de 10% de umidade, e após são armazenados de 1 a 5 dias (CUSTÓDIO, 2003);

– Quebra: reduz as dimensões do material sólido e permite a separação da casca através de máquinas que possuem, geralmente, dois pares de rolos estriados e rotativos, em cada um dos pares, a velocidade de cada rolo é diferente para provocar uma ação cisalhante nos grãos (CUSTÓDIO, 2003);

– Descascamento: segundo Mandarino e Roessing (2001), nesta etapa os cotilédones (polpas) são separados dos tegumentos (cascas), por meio de máquinas que quebram as cascas por batedores ou facas giratórias e as separam por peneiras vibratórias e insuflação de ar;

– Condicionamento: os cotilédones, após o descascamento, sofrem um aquecimento entre 55°C e 60°C, segundo Mandarino e Roessing (2001), para facilitar a laminação na próxima etapa do processo de preparação;

– Laminação: para facilitar a extração do óleo, o material sólido passa entre rolos de aço

inoxidável, resultando em lâminas ou flocos com espessura de dois a quatro décimos de milímetro, com um a dois centímetros de superfície (MANDARINO; ROESSING, 2001);

- Cozimento: o objetivo da etapa de cozimento é o rompimento das paredes celulares para facilitar a saída do óleo. "Nesse processo, a temperatura e a umidade dos flocos são elevados de 70°C a 105°C e 20%, respectivamente" (MANDARINO; ROESSING, 2001, p. 13).

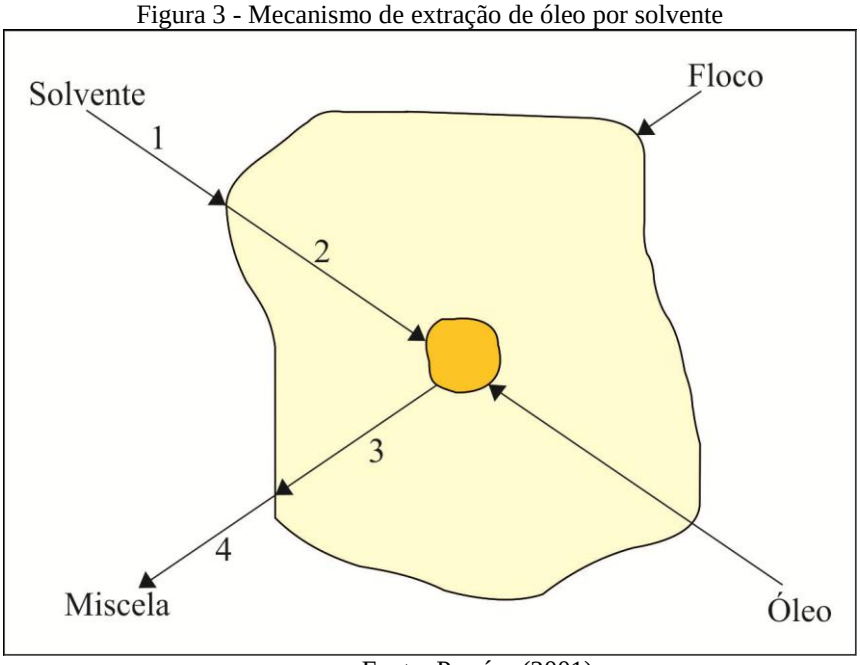
2.2 A extração do óleo bruto

O processo de extração por solvente é amplamente utilizado na indústria de alimentos. Neste processo, um solvente passa pelas sementes previamente preparadas em flocos, de forma a ocorrer a transferência do óleo da fase sólida para a fase líquida. O objetivo da extração é separar o máximo possível de óleo dos flocos com o mínimo de solvente. "A operação de extração é considerada a mais importante operação de todo processo de obtenção de óleo bruto por solvente. Nas indústrias típicas ela ocorre num único equipamento denominado de extrator" (PARAÍSO, 2001, p. 15).

Ainda conforme Paraíso (2001), dentro do extrator o óleo é obtido através de duas formas de extração, que ocorrem simultaneamente: extração por solução e extração por difusão. Na extração por solução a maior parte do óleo, disponível em células obstruídas durante o processo de preparação, é rapidamente dissolvido pelo solvente; já a extração por difusão é mais difícil e demorada, pois ocorre através da difusão do solvente até pequenas quantidades de óleo dentro de células ainda intactas. No final do processo de extração a solução do óleo no solvente é chamada de *miscela*. "Na prática, não ocorre extração completa. O menor conteúdo de óleo no farelo após a extração gira em torno de 0,5% a 0,6%." (MANDARINO; ROESSING, 2001, p. 15).

A Figura 3 demonstra, segundo Paraíso (2001), as etapas distintas que devem ocorrer durante a extração do óleo presente no floco: a etapa 1 refere-se ao contato do solvente com a superfície do floco; a etapa 2 refere-se à difusão do solvente da superfície do floco até o óleo do interior; a etapa 3 refere-se à difusão da mistura do solvente e óleo (miscela) através do floco até a superfície; a etapa 4 refere-se à drenagem da miscela para longe do floco extraído.

Figura 3 - Mecanismo de extração de óleo por solvente



Fonte: Paraíso (2001)

O solvente atualmente utilizado é o hexano, um derivado do processo de refino de petróleo, segundo Mandarino e Roessing (2001). Este solvente é utilizado por satisfazer a requisitos tais como "ser totalmente apolar e dissolver prontamente o óleo, ser miscível com água, ter baixo calor latente de ebulição, não atacar as canalizações e os aparelhos com os quais entra em contato" (OETTERER; REGITANO-D'ARCE; SPOTO, 2006, p. 319).

Por outro lado, o uso do hexano apresenta algumas desvantagens, pois segundo a sua FISPQ da Petrobras (2012), o hexano é classificado como um produto líquido inflamável e nocivo, seus vapores são prejudiciais ao meio ambiente. Segundo Shoemaker (1981), em campo aberto, a explosão do vapor de vinte galões de hexano equivale à força explosiva de uma libra de TNT. O hexano pode causar danos à saúde humana, "se a exposição for prolongada, pode provocar dor de cabeça, náuseas, tonteiras, perturbações visuais e auditivas, além de excitação" (PETROBRAS, 2011, p. 1).

2.3 A destilação da miscela

Segundo Paraíso (2001), o objetivo desta etapa do processo é a separação do solvente do óleo com a máxima recuperação possível do hexano. Ao sair do extrator a miscela passa usualmente por uma filtragem, para a remoção dos finos, e é transferida para um destilador contínuo, onde o óleo é separado do solvente por aquecimento sob vácuo, à temperatura de 70°C a 90°C. No destilador a redução de solvente no óleo pode chegar até cerca de 5% e o hexano residual é destilado em um evaporador de filme com insuflação de vapor direto (MANDARINO; ROESSING, 2001).

2.4 A dessolventização e tostagem do farelo

Segundo Mandarino e Roessing (2001), a torta proveniente da extração passa por um processamento térmico para a retirada do solvente residual e para inativar os fatores antinutricionais como inibidores de tripsina, as lectinas ou fitohemaglutininas e substâncias que causam sabor desagradável. Conforme Carvalho, Kirschnik, Paiva e Aiura (2002), a presença de inibidores de tripsina no trato intestinal inibe a ação da tripsina, que é responsável pela digestão das proteínas, levando a um aumento na produção enzimática pelo pâncreas e à hipertrofia deste órgão. "Lectinas ou Fitohemaglutininas são definidas como proteínas que possuem afinidade específica por certas moléculas de açucares e são capazes de aglutinar eritrócitos de sangue humano e animal" (MIZUBUTI; IDA, 1999, p. 108).

O equipamento mais usado atualmente neste processo é o dessolvetizador-tostador (DT), como afirmam Mandarino e Roessing (2001). O DT opera continuamente de cima para baixo através de compartimentos denominados de estágios e se divide em duas etapas: dessolventização que visa à retirada da maior parte do solvente retido na torta; tostagem onde o farelo de soja recebe um tratamento de calor e de umidade para evaporar mais alguma quantidade de hexano, que permaneceu após a etapa da dessolventização, e para destruir enzimas prejudiciais à sua digestibilidade. "O DT típico tem de 5 a 8 estágios sendo que, normalmente, nos dois primeiros ocorre a dessolventização" (PARAÍSO, 2001, p. 34).

2.5 A recuperação do solvente

Segundo Mandarino e Roessing (2001), praticamente todo o solvente usado na extração do óleo é removido na dessolventização da miscela e do farelo. Os mesmos autores também afirmam que a solubilidade do hexano na água, proveniente do vapor usado na dessolventização e tostagem, é mínima. Portanto, a principal causa de perda de solvente é a mistura incondensável formada entre seus vapores e o ar. Dadas às características do hexano, esta é uma etapa importante no processo de extração por evitar danos ambientais, humanos e perdas econômicas, já que o solvente é reaproveitado.

A recuperação do solvente desta mistura é efetuada com o emprego de compressores de frio ou, em instalações mais modernas, por colunas de absorção com óleo mineral. Isto é possível porque a solubilidade do hexano em óleo mineral é maior do que no ar. "Nessas instalações, os gases incondensáveis entram na parte inferior da coluna e o solvente é absorvido pelo óleo mineral em contracorrente, sendo esse contato aumentado por meio de anéis "Raschig" ou por atomização" (MANDARINO; ROESSING, 2001, p. 19).

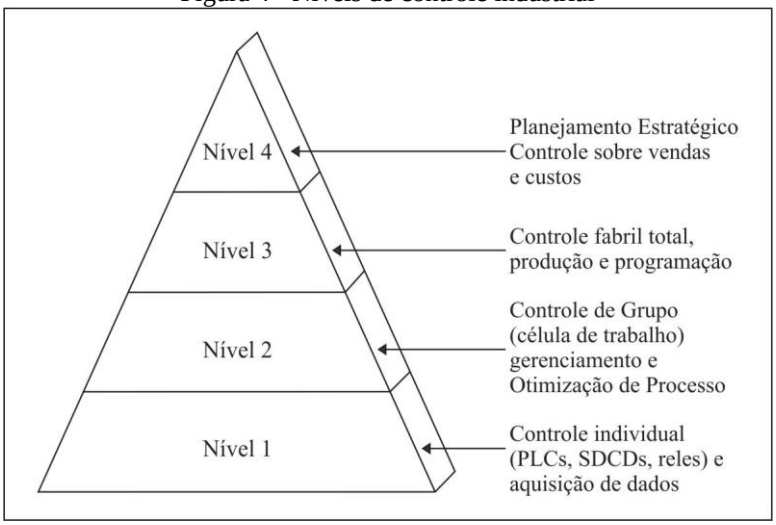
3. Testes de softwares de controle industrial

Este capítulo contextualiza os tipos de software de controle industrial e os principais testes que podem ser aplicados a estes softwares. O escopo teórico possui as bases de Andrade (2007), Elipse (2012), Feijó (2007), Pressman (1995), Leveson (2002), Sommerville (2005), Webb e Greshock (1992) e Wellenreuther e Zastrow (2005).

3.1 Sistemas de automação industrial

Segundo Feijó (2007), a partir da década de 1990, com os avanços tecnológicos e a disponibilização dos dados de produção, a automação industrial passou a integrar as várias etapas do negócio, desde o chão de fábrica aos sistemas corporativos. A pirâmide da automação industrial, demonstrada na Figura 4, é uma das formas de classificação do grau de automação de uma planta industrial, segundo Webb e Greshock (1992).

Figura 4 - Níveis de controle industrial



Fonte: Adaptado de Webb e Greshock (1992)

Exemplos de dispositivos constituintes dos níveis de controle industrial:

– Nível 1: botoeiras, chaves de emergência, medidores de vazão, sensores de temperatura, pressão, nível, umidade, opacidade, PH, movimento, CLPs (Controlador Lógico Programável), controladores digitais do tipo "*single-loop*", SDCDs (Sistema Digital de Controle Distribuído), inversores de freqüência e demais *drivers* de acionamento;

– Nível 2: SSs (Sistemas supervisórios), simuladores de processo, sistemas de operação por batelada e demais interfaces industriais homem-máquina;

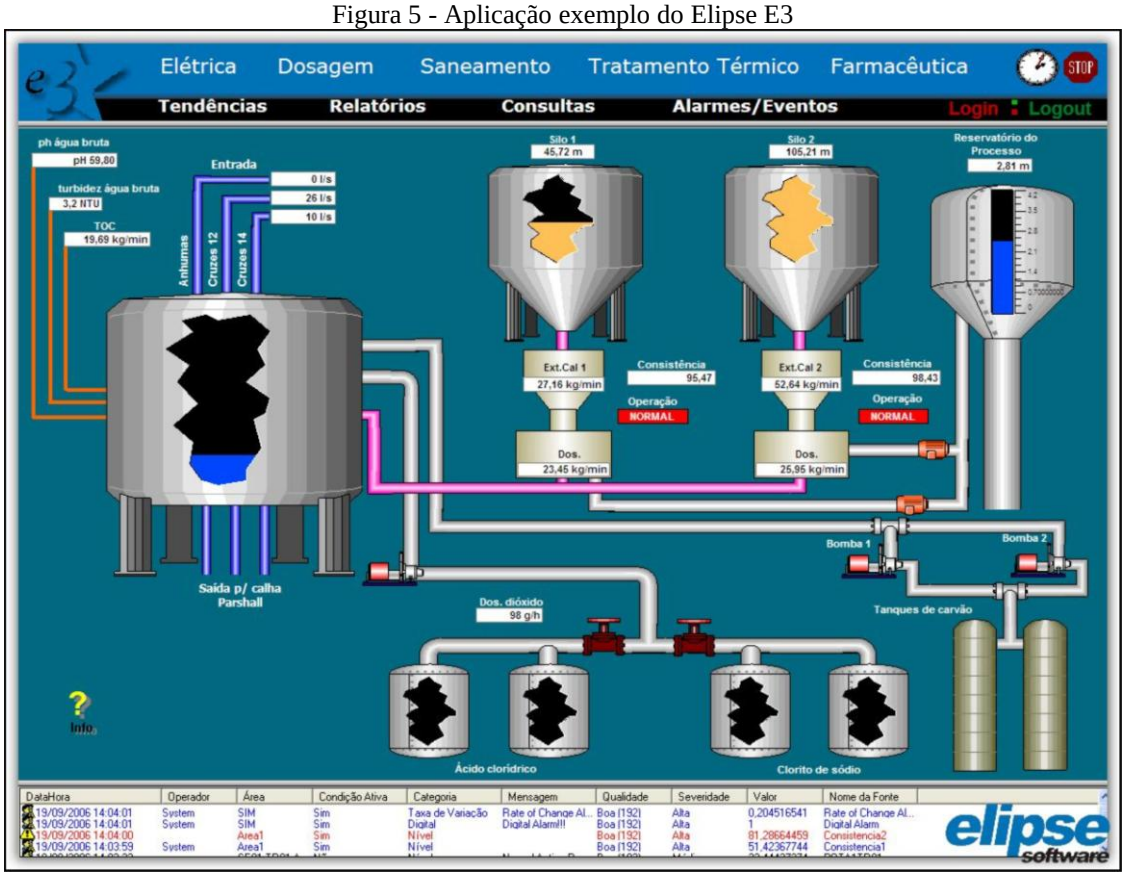
– Nível 3: é definido pela exploração dos sistemas MRP (Material Requirement Planning), *Just-in-time* e o MRP II (*Manufacturing Resource Planning*);

– Nível 4: sistemas ERP (*Enterprise Resource Planning*), gestão de recursos coporativos, etc.

O presente artigo tem como foco de pesquisa os testes de softwares nos níveis 1 e 2 da pirâmide de controle industrial, mais especificamente os softwares de supervisão e os softwares dos CLPs, por terem funções vitais em um processo automatizado, segundo Andrade (2007).

CLP (Controlador Lógico Programável) tem a estrutura de um computador e sua função é de armazenar e executar o programa do usuário. É constituído basicamente de uma fonte de alimentação, uma CPU (Unidade Central de Processamento), módulos de entrada e saída digitais e um barramento interno. Suporta também módulos de leitura de sinais analógicos e funções específicas, como lógica, seqüenciamento, temporização, contagem e aritmética (WELLENREUTHER; ZASTROW, 2005). As principais linguagens usadas para a programação de CLPs são divididas em dois tipos: textuais e gráficas. Textuais: IL (*Instruction List*) e ST (Structured Text). Gráficas: SFC (*Sequential Function Chart*), LD (*Ladder Diagram*) e FBD (*Function Block Diagram*). Esta pesquisa se concentra especificamente nos CLPs S7-300 / S7-400 da Siemens.

Figura 5 - Aplicação exemplo do Elipse E3



Fonte: Elipse (2012)

Os sistemas de supervisão, também conhecidos como SCADA (*Supervisory Control and Data Aquisition*), são sistemas onde são monitoradas e rastreadas informações de um processo produtivo ou instalação física. As informações coletadas através de equipamentos de aquisição de dados são manipuladas, analisadas, armazenadas e exibidas de forma amigável (interfaces homem-máquina) ao operador. As variáveis de campo, numéricas ou alfanuméricas de uma aplicação SCADA, são representadas através de *Tags* que são usadas para executar funções computacionais (operações matemáticas, lógicas, etc.) e também podem representar condições de alarmes, quando o valor da *Tag* excede limites pré-determinados (ELIPSE, 2012). A Figura 5 ilustra uma tela da aplicação exemplo do Elipse E3, ferramenta de desenvolvimento de software SCADA utilizada nesta pesquisa.

3.2 Teste de software

A atividade de teste de software é crucial para a garantia de qualidade de software e seu objetivo é descobrir sistematicamente diferentes categorias de erros em menor tempo e esforço. Os erros podem surgir desde o inicio do projeto e quanto mais tardiamente forem detectados maior será o custo de correção. Segundo Pressman (1995), o custo por correção pode ser até 100 vezes menor se a correção for feita ainda na fase de desenvolvimento.

"A atividade de teste não pode mostrar a ausência de *bugs*; ela só pode mostrar se defeitos de software estão presentes" (PRESSMAN, 1995, p. 789). Uma atividade de teste bem conduzida não encontra apenas erros, mas também verifica se as funções de software estão de acordo com as especificações e se foram cumpridos os requisitos de desempenho. A partir da compilação e avaliação dos resultados de teste surge uma indicação de qualidade e confiabilidade do software.

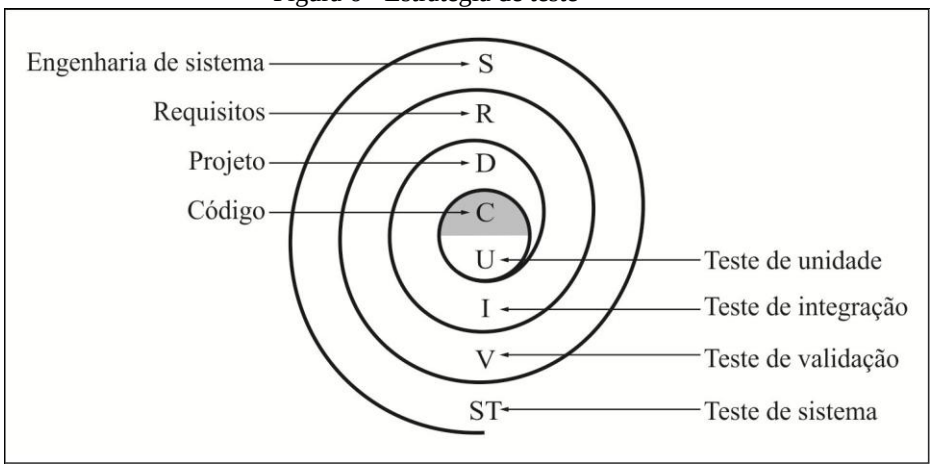
Segundo Pressman (1995), existem duas maneiras de testar software:

- Teste de caixa preta (*black box*): se concentra nos requisitos funcionais, sem levar muito em consideração a estrutura interna lógica do software; sua finalidade é detectar erros de interface, funções incorretas ou inexistentes, erros de estrutura de dados, de acesso a bancos de dados, problemas de desempenho, de inicialização e de finalização;
- Teste caixa branca (*White box*): se baseia na estrutura de controle procedimental, onde são testados os caminhos lógicos do software; a partir de aspectos de implementação são derivados casos de teste com definições de uso de variáveis, conjuntos específicos de condições e laços.

Pressman (1995) compara o processo de engenharia de software a uma espiral, ilustrada na Figura 6. Movimentando-se, de fora para dentro, ao longo da espiral, um projeto de software começa com a engenharia de sistema que leva à análise de requisitos, passa pela fase de projeto e no final à codificação. A cada volta na espiral o nível de abstração diminui. Do mesmo modo, fazendo o caminho inverso pela espiral, uma estratégia de teste pode ser pensada. A atividade de teste inicia

no vértice com o teste unitário e vai progredindo a cada volta, passando para o teste de integração, de validação e, finalmente, teste de sistema.

Figura 6 - Estratégia de teste



Fonte: Pressman (1995)

3.2.1 Teste de unidade

O teste de unidade, baseado na caixa branca, tem por objetivo fazer a verificação do módulo, a menor unidade desenvolvida no projeto. Importantes caminhos de controle são testados, através da descrição detalhada do projeto, visando descobrir a existência de falhas dentro das fronteiras do módulo. O campo de ação restrito para o teste unitário limita a complexidade dos testes e os erros que podem ser detectados por eles (PRESSMAN, 1995).

Estão entre os erros mais comuns de computação, segundo Pressman (1995): precedência aritmética incorreta ou mal compreendida; operações em modo misto; inicialização incorreta; erro de precisão; representação simbólica incorreta de uma expressão. O mesmo autor também afirma que os casos de teste devem encontrar erros como: comparação de tipos diferentes de dados; operadores lógicos ou precedência incorretos; expectativa de igualdade quando um erro de precisão torna a igualdade improvável; comparação ou variável incorreta; término de laço impróprio ou inexistente; falha para sair quando é encontrada iteração divergente; variáveis de laço modificadas de maneira imprópria.

O projeto e execução de casos de teste de unidade encontra-se normalmente associado à etapa de codificação. Por meio da revisão das informações do projeto, são estabelecidos os casos de teste com maior possibilidade de encontrar falhas em cada um dos tipos citados acima, deve haver também um conjunto de resultados esperados para cada caso de teste (PRESSMAN, 1995).

Para cada unidade de teste, devido ao módulo não ser um programa individual, deve ser desenvolvido um software adicional que não é entregue com o produto final. Na maioria das aplicações, os softwares de teste de unidade podem ser um *driver* (programa principal), que passa

dados para o módulo a ser testado e imprime os resultados, e/ou um *stub* (programa simulado), que substitui módulos subordinados ao módulo em teste, imprime a verificação de entrada e retorna. Quando um módulo é bem projetado, o número de casos de teste é reduzido e falhas são mais facilmente previstas e descobertas se apenas uma função é endereçada por módulo (PRESSMAN, 1995).

3.2.2 Teste de integração

O teste de integração, baseado também na caixa branca, tem por objetivo a construção da estrutura de programa a partir dos módulos no nível de unidade, e ao mesmo tempo, encontrar erros associados a interfaces. Mesmo que todos os módulos sejam testados no nível de unidade, não há garantias de que funcionarão, quando colocados juntos, de acordo com o especificado no projeto. Em razão disso, são feitos testes de integração, para encontrar problemas como: perda de dados; efeito inesperado de um módulo sobre outro; funções subordinadas combinadas podem não gerar a função principal esperada; uma imprecisão aceitável, quando individual, pode ser ampliada a níveis inaceitáveis; problemas nas estruturas de dados globais (PRESSMAN, 1995).

Existem dois modos para se testar a integração, segundo Pressman (1995): a integração não incremental e a integração incremental. Na primeira todos módulos são colocados juntos e o programa é testado como um todo, o que torna a correção mais difícil, pois um vasto conjunto de erros é encontrado, o que dificulta o isolamento das causas; a integração incremental é o oposto da abordagem anterior, nela o programa é construído e testado em pequenas porções, onde os erros são mais facilmente isolados e corrigidos e há maior possibilidade das interfaces serem completamente testadas.

3.2.3 Teste de validação

O teste de validação, baseado na caixa preta, tem por objetivo verificar se o software funciona de acordo com as expectativas do cliente. As expectativas do cliente são definidas no documento de Especificação de Requisitos de Software, na seção de Critérios de Validação, onde estão as informações que são a base para o teste de validação (PRESSMAN, 1995).

O software é avaliado através de testes que demonstram a conformidade com os requisitos. Os testes são projetados para garantir que todos os requisitos funcionais e de desempenho sejam satisfeitos, que a documentação esteja correta e que outros requisitos, como portabilidade e compatibilidade, sejam obedecidos. A realização de cada caso de teste de validação resultará em duas possibilidades: ou as características estão em conformidade com as especificações ou um desvio é descoberto e é criada uma lista de deficiências. No entanto, um erro ou desvio encontrado nesta fase do projeto representa um grande problema, pois dificilmente pode ser corrigido antes do término do software (PRESSMAN, 1995).

Nesta fase dos testes é feita a Revisão de Configuração, também chamada de auditoria. O objetivo desta revisão é assegurar que elementos de configuração do software, como código fonte e executável, documentação, e ferramentas de desenvolvimento estejam devidamente desenvolvidos e catalogados. Estas informações são usadas como apoio durante a fase de manutenção do software (PRESSMAN, 1995).

3.2.4 Teste de sistema

No teste de sistema o software, é integrado a outros elementos no sistema computacional, como hardware, outros softwares, informações e pessoas. O ambiente do teste de sistema deve ser similar ao ambiente onde o software final será utilizado. Segundo Pressman (1995), o teste de sistema é constituído de uma sucessão de testes, cada um com uma finalidade diferente, para verificar se todos os elementos foram integrados e realizam as suas funções de forma apropriada.

Testes que fazem parte da fase de teste de sistema, segundo Pressman (1995):

– Teste de recuperação: é um teste onde o software é forçado a falhar de várias formas para verificar se este é capaz de se recuperar ou tolerar a falhas de maneira apropriada e em tempo aceitável; se a recuperação for automática são verificados a recuperação dos dados, o reinício e mecanismos de *checkpointing*; se a recuperação for manual o tempo médio até o reparo é medido para verificar se está dentro dos limites aceitáveis;

– Teste de segurança: verifica a vulnerabilidade a acessos indevidos e deve ser conduzido com o intuito de burlar os mecanismos de proteção do sistema;

– Teste de estresse: verifica a resposta do sistema em condições anormais, como cargas de trabalho extremas, memória insuficiente, recursos limitados, busca excessiva por dados em disco e volume anormal de entrada de dados;

– Teste de desempenho: verifica o desempenho de *run-time* do software dentro do sistema integrado; às vezes é combinado ao teste de estresse e muitas vezes necessita instrumentação de hardware e de software.

3.2.5 Teste de sistemas críticos

Para Sommerville (2005), um sistema é considerado crítico quando uma falha em seu funcionamento pode resultar em perdas econômicas significativas, desastres ambientais, danos físicos ou ameaças à vida humana. Segundo Leveson (2002), o software crítico é um componente de um sistema crítico, que pode levar este sistema a um estado perigoso.

Os testes são essenciais em sistemas críticos, mas não é a solução para todos os problemas. Não é considerada uma falha em um componente se o comportamento do mesmo satisfaz à sua especificação de requisitos, mesmo que os requisitos incluam situações de risco no contexto do sistema. Os requisitos dos componentes do sistema podem estar incompletos ou errados, podem

surgir problemas de estados não tratados no sistema e também de condições ambientais. Basear-se apenas na implementação exata dos requisitos, na maioria dos casos, não garante a segurança do software (LEVESON, 2002).

Casos de teste devem ser escritos, com base na atividade de segurança de software, a partir da identificação e avaliação de casualidades que possam representar situações de perigo no sistema. A partir da identificação das situações de risco podem ser usadas técnicas de análise, como análise de árvore, lógica de tempo real ou modelos em rede de Petri, para prever cadeias de eventos que podem causar perigo e a probabilidade de que cada um ocorra (PRESSMAN, 1995).

4. Procedimentos metodológicos

A partir da necessidade da realização do artigo científico e das características do trabalho, em relação aos parâmetros de classificação, a técnica empregada é de documentação indireta em fontes de dados secundárias. Segundo Otani (2010), a documentação indireta em fontes secundárias caracteriza-se pela coleta de dados por meio de uma pesquisa bibliográfica. Para Rampazzo (2005), a pesquisa bibliográfica se dispõe a explanar um problema a partir de referências teóricas publicadas (livros, teses, dissertações, artigos científicos, etc.). O período de realização desta pesquisa se estendeu de maio a outubro de 2011.

Quanto aos objetivos, caracteriza-se como pesquisa: exploratória em seus primeiros estágios, quando foram investigados os testes de software crítico; descritiva nas demais etapas da pesquisa, onde foram descritos o processo de extração de óleo de soja, os métodos de teste de software e os resultados da pesquisa. Segundo Otani (2010), a pesquisa exploratória consiste em realizar um estudo preliminar de um fato ou fenômeno, buscando criar maior familiaridade com o problema pesquisado. Ainda segundo o mesmo autor, a pesquisa descritiva consiste na descrição de um fato ou fenômeno através da observação ou de levantamentos. Quanto à abordagem do problema é qualitativa, pois os dados coletados são analisados pelo método indutivo e não são empregados métodos e técnicas estatísticas (OTANI, 2010).

5. Resultados da pesquisa

Embora os métodos convencionais de teste de software iniciem pelo teste de unidade, devido à natureza dos softwares analisados nesta pesquisa, o recomendável é que os testes iniciem a partir da integração das aplicações CLP e SCADA, ou seja, a partir da etapa de teste de sistema. Deste modo, alguns dos testes podem ser conduzidos com maior eficiência em determinadas funcionalidades, que são interligadas nas duas aplicações.

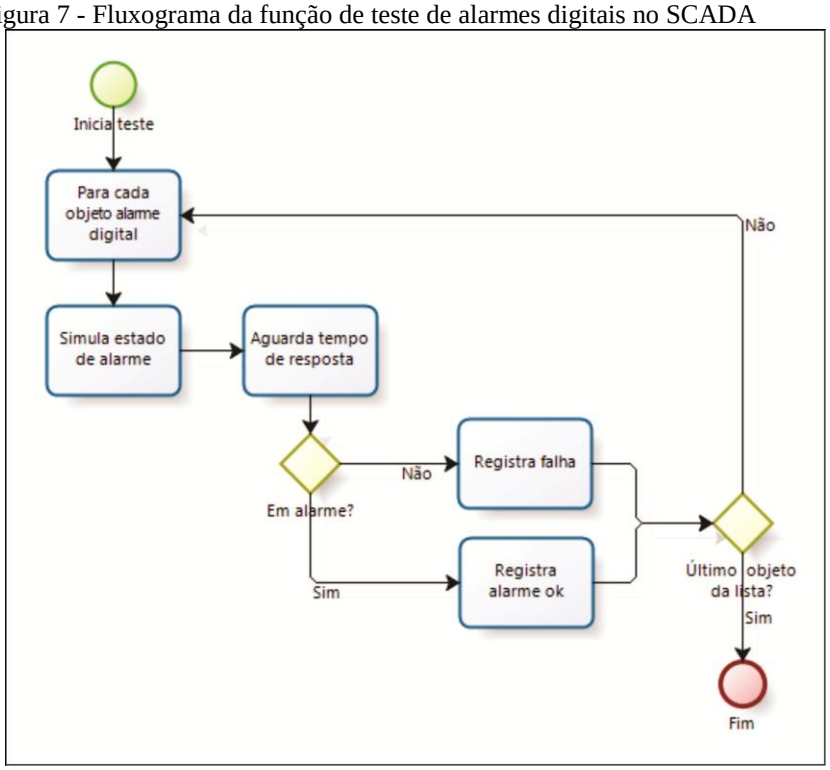
Por exemplo, um alarme digital, que tem sua função lógica dentro do CLP e a monitoração dos estados e mensagens de aviso dentro do SCADA. Neste caso o CLP vai gerar o alarme, mas é

no SCADA que o aviso do alarme vai aparecer e é onde o operador poderá reconhecer o mesmo. Se os testes ocorrerem com CLP e SCADA integrados serão verificadas as funções de alarme nos dois softwares ao mesmo tempo. Na espiral, proposta por Pressman (1995), na Figura 6, devemos fazer o caminho inverso dos testes de software, primeiro são os testes de sistema envolvendo a integração das aplicações, depois os testes de integração das unidades e assim por diante.

As ferramentas E3 e Step7, ao menos até a data de conclusão desta pesquisa, não disponibilizam ferramentas específicas para a automação de teste de software, como robôs de teste, por exemplo, mas disponibilizam ferramentas como *tag demo* e S7-PLCSIM. No E3 as *tags demo* são *tags* que simulam valores de acordo com o tipo de dado selecionado. O S7-PLCSIM é um software que permite a simulação do funcionamento de um CLP, onde é possível carregar e rodar a aplicação para testes. Apesar de não existirem ferramentas específicas, é possível o desenvolvimento de rotinas de teste, tanto no software do CLP quanto no do SCADA. A Figura 7 demonstra um exemplo de rotina para o teste dos alarmes digitais do sistema a partir do software SCADA, onde no final, ao menos todos os endereçamentos para este objeto serão verificados.

Algumas das ferramentas convensionais de teste de software, principlamente de automação de teste, pelo menos até a data de conclusão desta pesquisa, não se aplicam aos softwares de supervisão e de CLP. Além disso, existem outros fatores a serem considerados, antes de investir no uso de uma ferramenta de teste, como facilidade de uso, finalidade, eficiência e custo. Em todos os casos é preciso ter uma equipe de testes capacitada para o uso da ferramenta escolhida.

Figura 7 - Fluxograma da função de teste de alarmes digitais no SCADA



Fonte: Dados primários, 2012

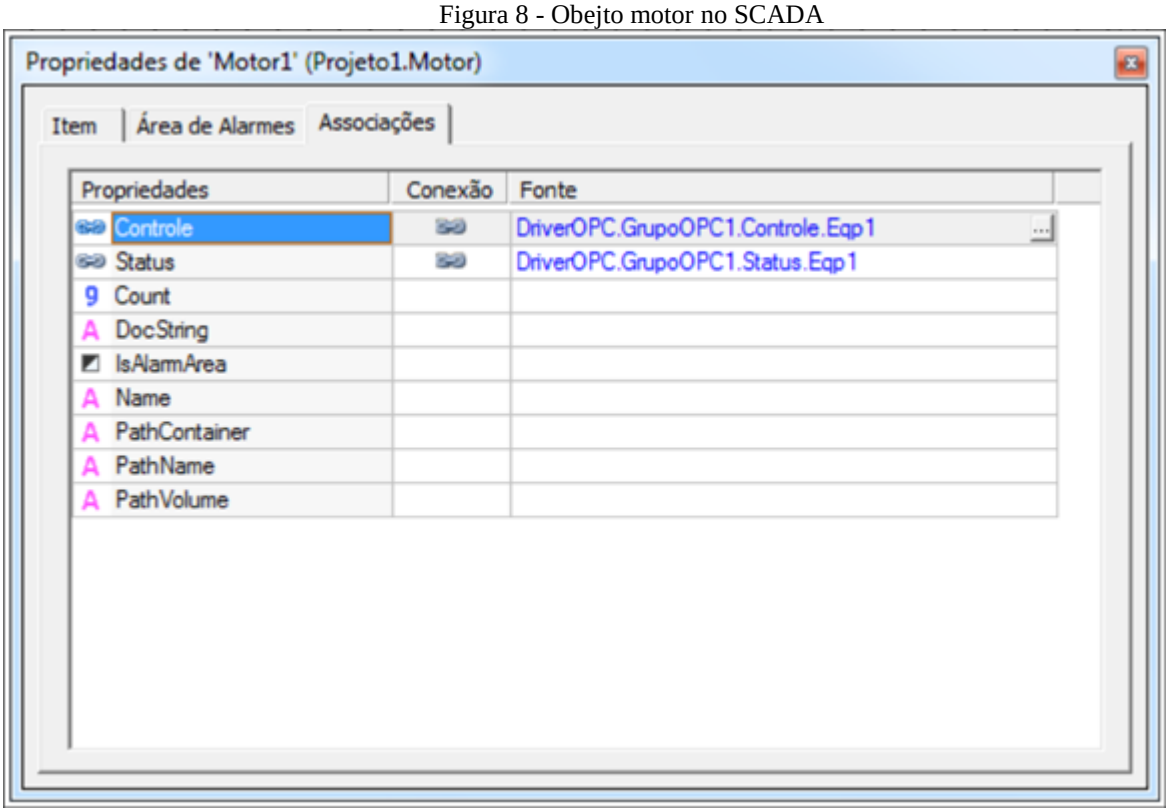
A simulação dos estados de variáveis de campo, através da implementação de rotinas de teste no software do CLP, é uma maneira muito eficiente de testar tanto o próprio software do CLP quando o SCADA. Neste caso, algumas das rotinas de teste precisam ser reescritas a cada novo projeto, devido às particularidades desse tipo de aplicação, mas podem ser matidas indefinidamente no software. As rotinas de teste podem ser habilitadas no ambiente de laboratório e desabilitadas na versão final de implantação, não afetando o desempenho do software final. Alguns dos tipos de simulações possíveis são: variáveis analógicas, variáveis digitais, funções de equipamentos, simulação de pesagens de balanças, etc. A maior vantagem é que os próprios desenvolvedores podem implementar as rotinas de teste durante o desenvolvimento do software sem a necessidade de capacitação específica.

Quando existe uma padronização de bibliotecas de objetos e funções, tanto no SCADA quanto no CLP, as mesmas podem ser importadas de uma aplicação para outra, sem a necessidade de executar novamente todo o teste de unidade. Por exemplo, uma função motor no CLP, depois de escrita e validada, pode ser incluída em novos softwares sem a necessidade de testá-la novamente, a menos que haja alguma alteração. A programação orientada a objetos é uma das maiores vantagens do Elipse E3, em relação a outras ferramentas de desenvolvimento de sofwares de supervisão mais usadas no mercado mundial. O uso da programação orientada a objetos ou de bibliotecas de funções, dependendo do tamanho do software, pode reduzir a escala de tempo de desenvolvimento de meses para semanas.

O mesmo também pode ser aplicado às bibliotecas de funções de teste dentro do software de supervisao, como demonstrado na Figura 7, que não precisam ser reescritas para cada nova aplicação. As bibliotecas de testes, dentro da própria ferramenta de desenvolvimento do software SCADA, são uma boa solução para determinados tipos de testes, pois não requerem o uso de ferramentas adicionais, são reutilizáveis e a equipe de desenvolvimento também não precisará de capacitação adicional.

O uso de funções e bibliotecas de objetos homologados permite aos testadores concentrarem todos os esforços nos testes de sistema, de validação, de integração e de unidades mais específicas de cada projeto. Uma parte crítica em um sistema de controle industrial é o teste de sistema, se houver divergências entre endereços de varáveis no software do CLP e do SCADA, isto poderia levar todo o sistema a uma situação de perigo. Um sistema de controle industrial pode conter milhares de variáveis de campo e cada uma destas variáveis precisa estar endereçada corretamente nas duas aplicações. A Figura 8 demonstra um exemplo de um objeto motor, bem simplificado, no software de supervisão e os endereços de suas propriedades para comunicação com o software do CLP, através de um *driver* de comunicação OPC (OLE *for Process Control*).

Figura 8 - Obejto motor no SCADA



Fonte: Dados primários (2012)

Fabricantes de CLPs, como Siemens, disponibilizam softwares para a simulação do funcionamento do hardware do CLP para testes de aplicações, no entanto, nenhuma simulação substitui completamente os testes no hardware final. Alguns problemas podem não ser detectados nos testes em simuladores. Uma falha grave, em um projeto de sistema de controle com CLP, seria esperar para carregar o software apenas no *start-up* (a posta em marcha da planta industrial) e verificar que o tempo de varredura é alto demais, tornado o sistema inviável, ou que determindada função usada no software não é suportada pelo modelo de hardware usado. Pior que isto, seria descobrir apenas no final que a memória do equipamento não comporta todo o software. O recomendável é sempre executar os testes o mais cedo possível em um equipamento similar ao que receberá o software no final.

Os testes dos softwares em ambiente industrial são absolutamente indispensáveis. Durante o *start-up* tudo precisa ser testado novamente, pois não há garantias de que o software homologado em laboratório vai se comportar extamente do modo esperado no hambiente industrial. Algumas falhas não podem ser detectadas ou simuladas em laboratório, como falhas nas interações entre os equipamentos de campo, oscilações inesperadas de valores de variáveis, interferêcnias no sinal da rede industrial.

Embora tudo tenha que ser testado novamente no ambiente industrial real, os testes em laboratório também são indispensáveis, pois durante o *start-up* não é a ocasião, ou não deveria ser, para testar as funções lógicas do software. Um dos motivos é que os *start-ups* tem a data de início e

tempo de duração definidos com antecedência. Algumas empresas até cobram multas diárias por atraso, por isto não há tempo para testar funções básicas ou fazer grandes correções nos softwares durante este período.

O processo industrial de extração de óleo por hexano é considerado crítico, devido à periculosidade do solvente empregado. Para Sommerville (2005), um sistema é considerado crítico quando uma falha em seu funcionamento pode resultar em perdas econômicas significativas, desastres ambientais, danos físicos ou ameaças à vida humana. Segundo Leveson (2002), o software crítico é um componente de um sistema crítico, que pode levar este sistema a um estado perigoso. As empresas de desenvolvimento de softwares de controle industrial que não possuem um processo de teste de software expõem os seus clientes à situções de risco.

As indústrias de extração por solvente fornecem a documentação de segurança de software à empresa contratada para o desenvolvimento do software, contendo as situações de perigo bem identificadas. Além das situações de risco, a documentação também contém as respectivas medidas cabíveis. A partir desta documentação os casos de teste críticos devem ser escritos. São também críticos os testes de recuperação, de estresse e de desempenho, pois em um processo contínuo e crítico, como o abordado nesta pesquisa, os softwares devem funcionar ininterruptamente e sem falhas.

Devido à periculosidade do processo industrial em questão, é exigido que a empresa desenvolvedora do software comprove experiência na condução de projetos desta natureza, seja por meio de projetos anteriores ou por meio da contratação de profissionais especializados. Uma empresa que não tenha nenhuma experiência em sistemas críticos é automaticamente desconsiderada.

Os testes finais do sistema são realizados na planta industrial. Considerando os testes de continuidade elétrica já concluídos, inicialmente são realizados os testes de I/O (*Input e Output*), onde todos os pontos de ligações de sinais do campo e CCM (Centro de Comando de Motores) para o CLP são verificados. Após isto são realizados os testes operacionais com os equipamentos ligados, mas sem carga, também conhecidos como testes a vazio. Nesta fase são feitos ajustes e aferições, são testados os sensores de proteção dos equipamentos e os intertravamentos. Após a finalização dos testes sem carga iniciam os testes dos equipamentos com carga nas condições nominais dos mesmos.

Na última fase a operação normal do sistema é acompanhada pelos operadores e integradores, também são feitos ajustes dos parâmetros do processo e verificação das faixas de alarmes e malhas de controle. O acompanhamento é a última fase dos testes de sistema e seu tempo de duração pode variar de acordo com planejamento pré-definido. Ao final desta fase o sistema será considerado implantado e em pleno funcionamento após a avaliação conjunta do integrador e do

profissional responsável no lado do cliente. A assinatura do termo de entrega oficializa o término do projeto e aceite por parte do cliente.

6. Conclusões

Todos os métodos convencionais de teste de software são aplicados também aos softwares de controle industrial, o que difere é o modo como os testes devem ser conduzidos. Embora os métodos de teste convencionais iniciem pelo teste de unidade, os testes de softwares de controle industrial são conduzidos com mais eficiência a partir da etapa de teste de sistema, devido à natureza destes softwares. Na espiral, proposta por Pressman (1995), na Figura 6, devemos fazer o caminho inverso dos testes de software, primeiro são feitos os testes de sistema envolvendo a integração das aplicações, depois os testes de integração das unidades e assim por diante.

Algumas das ferramentas convensionais de teste de software, principlamente de automação de teste, pelo menos até a data de conclusão desta pesquisa, não se aplicam aos softwares de supervisão e de CLP. Além disso, existem outros fatores a serem considerados, antes de investir no uso de uma ferramenta de teste, como facilidade de uso, finalidade, eficiência e custo. Em todos os casos é preciso ter uma equipe de testes capacitada para o uso da ferramenta escolhida.

Na maioria das situações, se as aplicações forem bem estruturadas, o desenvolvimento das funções de teste, nas próprias aplicações, é um artifício eficaz no teste de softwares de controle industrial. A simulação dos estados de variáveis de campo, através da implementação de rotinas de teste no software do CLP, é uma maneira muito eficiente de testar tanto o próprio software do CLP quando o SCADA. As rotinas de teste podem ser habilitadas no ambiente de laboratório e desabilitadas na versão final de implantação, não afetando o desempenho do software final. A maior vantagem é que os próprios desenvolvedores podem implementar as rotinas de teste durante o desenvolvimento do software sem a necessidade de capacitação específica. As bibliotecas de testes, como demonstra a Figura 7, dentro da própria ferramenta de desenvolvimento do software SCADA, também são uma boa solução para determinados tipos de testes.

Quando existe uma padronização de bibliotecas de objetos e funções, tanto no SCADA quanto no CLP, as mesmas podem ser importadas de uma aplicação para outra, sem a necessidade de executar novamente todo o teste de unidade e de integração. O uso da programação orientada a objetos e de bibliotecas de funções, dependendo do tamanho do software, pode reduzir a escala de tempo de desenvolvimento de meses para semanas.

Os testes críticos são absolutamente indispensáveis para a segurança do software crítico, pois uma falha em seu funcionamento pode resultar em perdas econômicas, desastres ambientais, danos físicos ou ameaças à vida humana. São também indispensáveis os testes de recuperação, de estresse e de desempenho, pois em um processo contínuo e crítico, como o abordado nesta pesquisa,

os softwares devem funcionar ininterruptamente e sem falhas.

Abstract

The objective of this research is to characterize the test methods that can be used in the tests of software of industrial control for extraction of soy oil for solvent. To reach such an objective it is served as the theoretical founds in the mark of the industrial process of soy extraction for solvent, of the softwares of control industrial employees and of the methods of software test. As for the methodological procedures, it is treated of an exploratory and descriptive research; in relation to the approach of the problem it is qualitative. The collection of data happens through researches of indirect documentation in sources of secondary data. The results of the research point that all of the conventional methods of software test are applied also to the softwares of industrial control, the one that differs is the way as the tests should be driven. The tests of software of industrial control are driven with more efficiency starting from the stage of system test, due to the nature of these softwares. Besides the conventional tests the research also appears that, due to danger of the employed solvent in the process of oil extraction, the healthy critical tests quite necessary for safety's warranty, quality and reliability of the softwares.

Key-words: industrial processes; test of software; extraction of oil of soy.

Referências

ANDRADE, Alexandre Acácio. **Desenvolvimento de sistema especialista com operacionalidade de aprendizado para operar em tempo real com sistemas industriais automatizados**. 2008. 154 f. Tese (Doutorado). Departamento de Energia e Automação Elétricas, Escola Politécnica da Universidade de São Paulo, São Paulo, SP, 2008.

BRASIL, Instrução Normativa N 37, de 27 de julho de 2007. Altera o inciso IV, do art. 2 o, do Capítulo I, do Anexo da Instrução Normativa N 11, de 15 de maio de 2007. **Diário Oficial da República Federativa do Brasil**. Brasília, DF, 30 jul. 2007. Disponível em: <http://www.in.gov.br/imprensa/visualiza/index.jsp?jornal=1&pagina=9&data=30/07/2007>. Acesso em: 14 maio 2012.

CARVALHO, M. R. B.; KIRSCHNIK, P. G.; PAIVA, K. C.; AIURA, F. S. Avaliação da atividade dos inibidores de tripsina após digestão enzimática em grãos de soja tratados termicamente. **Revista de Nutrição**. Campinas, n. 3, v. 15, 2002. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1415-52732002000300002>. Acesso em: 20 mar. 2012.

CUSTÓDIO, A. F. **Modelagem e simulação do processo de separação óleo de soja-hexano por evaporação**. 2003. 247 f. Dissertação (Mestrado). Faculdade de Engenharia Química, Universidade Estadual de Campinas, Campinas, SP, 2003.

ELIPSE. **Elipse knowledgebase**. Disponível em: < http://kb.elipse.com.br/pt-br/questions/62/O+que+s%C3%A3o+sistemas+supervis%C3%B3rios%3F>. Acesso em: 12 abr. 2012.

FEIJÓ, R. H. B. **Uma arquitetura de software baseada em componentes para visualização de informações industriais**. 2007. 88 f. Dissertação (Mestrado). Centro de Tecnologia, Universidade Federal do Rio Grande do Norte, Natal, RN, 2007.

LEVESON, N. **A new approach to system safety engineering**. Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, Estados Unidos, 2002.

MANDARINO, J. M. G.; ROESSING, A. C. **Tecnologia para produção do óleo de soja:** descrição das etapas, equipamentos, produtos e subprodutos. Londrina: Embrapa Soja, 2001.

MIZUBUTI, I. Y.; IDA, E. I. Constituintes antinutricionais e seus efeitos indesejáveis na alimentação. **Semina: Ciências Agrárias**. Londrina, n. 1, v. 20, p. 107-112, mar. 1999. Disponível em: <http://www.uel.br/revistas/uel/index.php/semagrarias/article/view/5045/4543>. Acesso em: 22 mar. 2012.

OETTERER, M.; REGITANO-D'ARCE, M. A. B.; SPOTO, M. H. F. **Fundamentos de ciência e tecnologia de alimentos**. Barueri, SP: Manole, 2006.

OTANI, N. **Metodologia do trabalho científico**. Apostila. 2010. 63 f. Instituto de Pós Graduação, Escola Superior de Criciúma, Criciúma, SC, 2010.

PARAÍSO, P. R. **Modelagem e análise do processo de obtenção do óleo de soja**. Tese (Doutorado). 2001. 220 f. Faculdade de Engenharia Química, Universidade Estadual de Campinas, Campinas, SP, 2001.

PETROBRAS. **Ficha de Informação de Segurança de Produto Químico – FISPQ**. Disponível em: <http://www.br.com.br/wps/wcm/connect/5a60a1004c4aa93190ddd20 869efed74/fispq-quim-alif-hexanobr.pdf?MOD=AJPERES&CACHEID=5a60a1004c4 aa93190ddd20869efed74>. Acesso em: 10 fev. 2012.

PRESSMAN, R. S. **Engenharia de software**. São Paulo: Editora Pearson Makron Books, 1995.

RAMPAZZO, L. **Metodologia científica**. 3. ed. São Paulo: Loyola, 2005.

SHOEMAKER, L. W. Solvent safety. **Journal of The American Oil Chemist's Society**. Piqua, Ohio, United States of America, n. 3, v. 58, p.197-198, 1981.

SOMMERVILLE, I. **Ingeniería del software**. 7. ed. Madrid, Espanha. Editora Pearson Educación, 2005.

WEBB, J.; GRESHOCK, K. **Industrial control eletronics**. Maxwell Macmillan International Editions, 1992.

WELLENREUTHER, G.; ZASTROW, D. **Automatisieren mit SPS Theorie und Praxis**. Wiesbaden, Hessen, Deutschland. Editora Friedr. Vieweg & Sohn, 2005.

Dados dos autores:

Nome completo: **Deise Regina Portal**

Filiação institucional: Escola Superior de Criciúma - ESUCRI

Departamento: Pós-graduação

Função ou cargo ocupado: Gerente de Desenvolvimento de Software

Endereço completo para correspondência (bairro, cidade, estado, país e CEP): Rua Joaquim Nabuco, 235 – apto. 603 (Centro) – Criciúma/SC. CEP: 88.802-200.

Telefones para contato: (48) 9911-4668 (48) 3411-5046

e-mail: deiseportal@hotmail.com

Nome completo: **Nilo Otani**

Filiação institucional: Universidade Federal de Santa Catarina - UFSC

Departamento: INPEAU/UFSC

Função ou cargo ocupado: Pesquisador

Endereço completo para correspondência (bairro, cidade, estado, país e CEP): Rua Afonso Pena, 494 – apto. 603 (Estreito) – Florianópolis/SC. CEP: 88.070-650.

Telefones para contato: (48) 9167-9077

e-mail: ni_otani@yahoo.com.br

Enviado em: 27/08/2012

Aprovado em: 08/05/2013