Thiago Javaroni Prati

DESENVOLVIMENTO DE UMA METODOLOGIA PARA REALIZAÇÃO DE TESTES EM PROGRAMAS DE CLP NA INDÚSTRIA DE PETRÓLEO E GÁS

Florianópolis(SC)

2014

Thiago Javaroni Prati

DESENVOLVIMENTO DE UMA METODOLOGIA PARA REALIZAÇÃO DE TESTES EM PROGRAMAS DE CLP NA INDÚSTRIA DE PETRÓLEO E GÁS

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas para a obtenção do Grau de Mestre em Engenharia de Automação e Sistemas.
Orientador: Jean-Marie Farines, Dr.Eng.
Coorientador: Max Hering de Queiroz, Dr.Eng.

Florianópolis(SC)

2014

Thiago Javaroni Prati

DESENVOLVIMENTO DE UMA METODOLOGIA PARA REALIZAÇÃO DE TESTES EM PROGRAMAS DE CLP NA INDÚSTRIA DE PETRÓLEO E GÁS

Esta Dissertação foi julgada aprovada para a obtenção do Título de "Mestre em Engenharia de Automação e Sistemas", e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Florianópolis(SC), 04 de Dezembro 2014.

João Ninguém
Coordenador

Max Hering de Queiroz, Dr.Eng.
Coorientador

Banca Examinadora:

Presidente da banca
Presidente

Jean-Marie Farines, Dr.Eng.
Orientador

Segundo membro

Terceiro membro

Quarto membro

A minha família e amigos

AGRADECIMENTOS

Agradeço...

A minha família que sempre esteve presente garantindo apoio emocional, por serem compreensivos e atenciosos.

Aos meus amigos com os quais eu sempre pude contar, por garantirem ótimos momentos de conversa, reflexão e distração.

Àquele que, em seu olhar, nada além de lealdade foi possível encontrar, que nunca deixou de expressar seu amor inabalável, que sempre esteve lá, presente, mesmo que em seus momentos mais preguiçosos, que nunca recusou sequer um simples afago, que me acompanhou por momentos extremamente importantes durante um longo tempo na minha vida e que, mesmo sem perceber, acabou deixando um sentimento de saudade e boas recordações que serão eternas. Meu querido bebê.

Aos meus orientadores pelas ótimas ideias dadas e discussões essenciais para o desenvolvimento desse trabalho, pela paciência e compreensão que tiveram comigo.

Proponha-se a atingir o sol e você poderá não o alcançar, mas sua seta voará muito mais alto do que se fosse apontada para um objeto ao mesmo nível de você.

Joel Hawes

RESUMO

Os programas de automação na indústria de petróleo e gás, seja o programa voltado para sistemas de segurança ou de uso geral, são projetados com base em especificações de segurança, definidas em projeto, que devem ser validados antes da implantação. Este documento propõe um método para o teste automático das especificações encontradas na Matriz Causa e Efeito e implementadas em Controladores Lógicos Programáveis (CLPs). Para esses testes, as especificações são representadas como um conjunto de modelos de rede de Petri que observam o comportamento do sistema controlado. O uso de um modelo formal permite composição de diferentes modelos de forma sistemática e a tradução das Redes de Petri em um programa que comanda as entradas de um CLP e observa quando o comportamento do CLP segue ou falha em seguir as especificações de segurança. Uma ferramenta protótipo foi desenvolvida para executar automaticamente o teste da Matriz Causa e Efeito em um dado CLP. Um projeto de um forno foi utilizado para verificar se a metologia proposta é fácil de usar e viável.

Palavras-chave: Controladores Lógicos Programáveis, Testes Automatizados, Validação, redes de Petri, Sistemas Instrumentados de Segurança.

ABSTRACT

The automation programs in the oil and gas industry, be it in Safety Instrumented Systems (SIS) or in general purpose Programmable Logic Controllers (PLC), are designed based on safety specifications that must be validated prior to deployment. This paper proposes a method for the automatic test of the Cause and Efect Matrix specifications on PLC systems. For such testing, the specifications are represented as a set of Petri net models that observe the controlled system behavior. The use of a formal model allows to systematically compose and translate the Petri Nets into a program that commands the PLC inputs and observes when the PLC outputs fail the safety specifications. A prototype tool has been developed to automatically perform the test of the Cause and Efect Matrix on a given PLC. A furnace project has been used to ascertain that the proposed method is easy to use and viable.

Keywords: Programmable Logic Controllers, Automatic Testing, Validation, Petri Nets, Safety Instrumented Systems.

SUMÁRIO

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

A complexidade dos problemas de controle industrial pode chegar a grandes escalas. Na Indústria de Petróleo e Gás, controles específicos podem ser usados para executar tarefas de automação que requerem um elevado grau de confiabilidade, uma vez que as falhas podem levar à danos sérios e custosos de equipamentos, danos ambientais e até mesmo a perda de vidas humanas, como visto em (SKOGDALE; SMOGELI, 2011). O desenvolvimento destes sistemas de controle requer uma grande atenção, uma vez que lida com requisitos rigorosos, que podem incluir restrições temporais, de segurança e de confiabilidade. Para o desenvolvimento destes projetos, as empresas costumam utilizar normas que têm como um objetivo a normalização da construção de um software de automação que atenda aos requisitos do projeto.

No caso da indústria de petróleo e gás, a análise da confiabilidade do sistema resulta na decisão da utilização ou não de equipamentos especializados para a automação de sistemas considerados críticos. A automação pode ser, então, realizada através de lógica de relés, de dispositivos programáveis como Controladores Lógicos Programáveis (CLPs). Exemplos de sistemas que podem vir a ser automatizados através de CLPs são: sistema de parada de emergência (*ESD-Emergency shutdown*); Sistema de parada de segurança (*Safety shutdown*); Sistema de intertravamento de segurança; Sistema de fogo e gás. Os sistemas utilizados devem atender às especificações do projeto a fim de garantir a segurança de operação da planta.

A metodologia para o desenvolvimento de sistemas de automação atualmente adotada pela Petrobras consiste em um sequência de passos começando a partir de especificações básicas da planta a ser automatizada e termina com o desenvolvimento de um software utilizado em um CLP para automação. Esta metodologia utiliza um conjunto de normas gerais e principalmente internas, como visto em (COMISSãO DE NORMAS TéCNICAS, 2002) e (COMISSãO DE NORMAS TéCNICAS, 2012). De acordo com essas normas, uma série de documentos que contenham informações relevantes ao longo do projeto são criados. A Matriz de Causa e Efeito (matriz C&E), por exemplo, é um documento que define a relação entre sinais de campo que indicam situações críticas e ações de segurança adequadas. O documento de Teste de Aceitação de Fábrica (FAT) descreve como testar o programa de automação final, a

fim de validar o programa de CLP, como descrito nos documentos do projeto.

O desenvolvimento da aplicação presente em um sistema de automação, seja ele implementado em um CLP ou ainda um SIS, é constituído por sequenciamentos, intertravamentos e outras lógicas que podem ser facilmente representadas por Sistemas a Eventos Discretos (SED). Estes sistemas são caracterizados por um espaço de estados discreto de valores lógicos cuja dinâmica é dirigida pela ocorrência de eventos e podem ser representados por modelos formais. Em um projeto, estes SED poderiam ser desenvolvidos através de metodologias específicas e implementados nos sistemas de automação, utilizando-se de linguagens das normas IEC 61131-3 e IEC 61499. Embora a utilização destas representações formais já tenham sido abordadas em outros trabalhos como (PAKONEN et al., 2013), (FARINES; QUEIROZ; CARPES, 2011) e (JúNIOR, 2011) para demonstrar sua efetividade no cumprimento dos requisitos por parte do programa, existe, ainda, uma baixa aceitação na utilização destas representações no momento do projeto da aplicação de um sistema de automação ou ainda no momento da validação desta aplicação.

Apesar da complexidade envolvida e da confiabilidade exigida em sistemas de automação, a validação de programas de CLP no âmbito industrial é feita principalmente através de testes e simulações. Estas validações são realizadas a partir dos requisitos presentes nos documentos básicos de projeto e acabam por demandar um longo tempo para serem executadas, visto que são manuais, e mesmo com o tempo desprendido, não são exaustivas. Consequentemente, alguns erros podem ser negligenciados e, portanto, podem resultar em danos aos equipamentos e, às vezes, ameaça aos seres humanos.

Dada a atual abordagem na realização dos testes em âmbito industrial, se mostra importante a utilização de algum método que possibilite uma maior facilidade na realização dos testes de programa de CLP. A geração de testes automatizados, por exemplo, permite a execução dos mesmos sem a presença humana, economizando tempo de projeto ao poder ser realizado independentemente da supervisão de um engenheiro e pode ainda aumentar o leque de situações testadas devido à essa mesma economia de tempo.

1.2 OBJETIVOS

O objetivo desta pesquisa é o elaboração de uma metodologia que permita o desenvolvimento de sistemas mais seguros e que sigam as conformidades estabelecidas em projetos complexos, como aqueles encontrados na indústria de petróleo e gás. Nesse sentido, defini-se uma metodologia de geração de testes que permite ao engenheiro da indústria do petróleo e gás certificar programas a partir da conformidade destes com os documentos básicos existentes no projeto. Para apoiar a aplicação desta metodologia, desenvolve-se uma ferramenta que permite gerar e executar de forma automática as sequências de testes necessárias para a validação.

Para a validação da ferramenta, utiliza-se um cenário exemplo . Neste cenário, uma planta desenvolvida em ambiente de simulação e um sistema de controle criado baseando-se nos requisitos de projeto para esta planta exemplo são utilizados. A ligação entre a planta simulada e este sistema de controle se dá desta maneira: a planta simulada recebe comandos oriundos do sistema de controle e responde com valores de sensores gerados pela simulação. Isso deve ser tal que o sistema de controle não perceba a diferença entre um sistema dinâmico real e a simulação. Assim, condições de operação da planta que seriam difíceis de serem implementadas em uma planta real por questões de segurança, por exemplo, pudem ser realizadas pelo sistema. Durante esse processo de simulação da planta e atuação por parte do sistema de controle criado, a ferramenta obtém os dados pertinentes para a obtenção de resultados.

1.3 ESTRUTURA DA DISSERTAÇÃO

Na sequência deste capítulo introdutório, são apresentadas no capítulo 2 técnicas para verificação e técnicas de teste de software, assim como alguns trabalhos da área. No capítulo 3 é apresentada a metodologia atual de desenvolvimento de software utilizada pela Petrobras, juntamente com um exemplo para melhor entendimento. Em seguida, no capítulo 4, uma nova metodologia para a realização de testes dentro do desenvolvimento de um projeto, segundo o exposto no capítulo 3, é desenvolvida. No capítulo 5, é apresentado um estudo de caso onde são realizados testes utilizando um projeto da Petrobras como base para os mesmos. Finalmente, no capítulo 6, são apresentadas as conclusões deste trabalho e sugestões para trabalhos futuros.

2 MODELAGEM E VALIDAÇÃO DE PROGRAMAS DE CLP UTILIZANDO MÉTODOS FORMAIS

Diversas técnicas podem ser utilizadas para o desenvolvimento de programas de CLP. Da mesma maneira, todo desenvolvimento de programa conta com a realização de algum tipo de validação durante este processo ou ao final do mesmo. As técnicas para desenvolvimento podem se basear em boas práticas de programação (JACOBSON et al., 1999) ou ainda podem ser baseadas em métodos formais. As boas práticas de programação podem ser estipuladas tanto pela companhia que o desenvolve, quanto por pesquisadores da área, enquanto as técnicas baseadas em métodos formais baseiam-se em algum tipo de representação formal do sistema como: autômatos, redes de Petri, redes Bayesianas ou outras e têm, como objetivo, a geração de um programa correto por construção.

Já a validação de um programa de CLP pode se dar através de verificação formal ou através de testes. Assim como no desenvolvimento baseado em métodos formais, a verificação formal consiste em se utilizar uma representação formal para verificar se o programa apresenta o comportamento esperado. Para tal, o programa é transformado em um modelo formal equivalente e, através de fórmulas, tem suas propriedades verificadas. Já os testes se baseiam na definição de conjuntos de valores de entrada e de saída esperados e, através da inserção das entradas e subsequente comparação entre valores de saída esperados e os obtidos, se analisa o correto funcionamento do programa.

Este capítulo aborda conceitos necessários para o entendimento de futuros temas deste documento, além de trabalhos relacionados. São apresentadas redes de Petri, trabalhos relacionados que lidam tanto com a modelagem de sistemas e geração de código, como com a verificação de programas de CLP através de métodos formais e a realização de testes. São apresentados os pontos mais essenciais das técnicas pertinentes e também dos trabalhos que se utilizam dessas abordagens para validação de programas de automação.

2.1 MODELAGEM DE SISTEMAS ATRAVÉS DE REDES DE PETRI

Rede de Petri é uma técnica de modelagem que permite a representação de sistemas se utilizando de um formalismo matemático (MA-

CIEL; LINS; CUNHA, 1996), (ZURAWSKI; ZHOU, 2004), (ADAM; ATLURI; HUANG, 1998). Essa representação permite modelar sistemas paralelos, concorrentes, assíncronos e não-determinísticos (HAMADI; BENATALLAH, 2003), (MORALES; MELO; MIYAGI, 2007), (YOO; JEONG; CHO, 2010).

Três componentes definem a representação gráfica de uma rede de Petri: a transição (representado por uma barra), o lugar (círculo) e o arco(seta) . Os lugares são variáveis de estado e as transições correspondem às ações realizadas pelo sistema (MACIEL; LINS; CUNHA, 1996) enquanto o arco ligada os dois elementos anteriores entre si e podem ser únicos ou múltiplos.

Lugares podem ser marcados ou não. Um lugar marcado é representado por um ou mais pontos desenhados dentro do círculo que representa o lugar e são chamados de fichas. A quantidade de fichas que um lugar possui é o que define o seu estado e é chamada de marcação deste lugar(FRANCêS, 2003). O conjunto dos lugares da rede de Petri forma o conjunto das variáveis de estado que representam o sistema modelado. Assim, as marcações de todos os lugares em um dado momento representa o estado do sistema e é chamado de marcação da rede(MURATA, 1989), (ZURAWSKI; ZHOU, 2004), (XU et al., 2007).

A rede de Petri permite a modelagem de diferentes tipos de modelos, dentre eles: a sincronização de processos, de operações concorrentes e de conflitos, a ocorrência de eventos assíncronos ou ainda compartilhamento de recursos (ADAM; ATLURI; HUANG, 1998), (NASSAR et al., 2008), (ZURAWSKI; ZHOU, 2004). Desde sua apresentação por Carl Adam Petri, redes de Petri foram utilizadas na modelagem e análise de diferentes tipos de sistemas e aplicações como: diagnóstico e tratamento de falhas não críticas (MORALES; MELO; MIYAGI, 2007) (SILVA, 2002) , protocolos distribuídos (KANESHIRO et al., 2007), aplicações industriais (NASSAR et al., 2008), fluxo de processos (KIEPUSZEWSKI; HOFSTEDE; AALST, 2003), controle supervisório (LEE; ZHOU; HSU, 2005) e diagnóstico e tratamento de falhas não críticas (JúNIOR, 2011).

Além disso, várias ferramentas para simulação, edição e análise foram desenvolvidas desde a década de 90((ZURAWSKI; ZHOU, 1994) , (BERTHOMIEU, 2004)) e possibilitam a representação da dinâmica do sistema e sua estrutura em diversos níveis de abstração (de acordo com a complexidade do sistema (NASSAR et al., 2008)), a análise das modelagens e também sua simulação.

2.1.1 Fundamentos

A rede de Petri ordinária é uma tupla $N = (P, T, I, O)$ associada com uma descrição de estado indicada por M. $P = p1, p2, ..., pn$ é um conjunto finito não-vazio de lugares, $T = t1, t2, ..., tm$ é um conjunto finito não-vazio de transições. I:PxT$\rightarrow \mathbb{N}$ é a matriz de entrada que relaciona transições com lugares de entrada enquanto O:PxT$\rightarrow \mathbb{N}$ é a matriz de saída que relaciona transições com lugares de saída. As matrizes de entrada e de saída contêm, em cada elemento a_{ij}, o peso do arco que liga o lugar p_i à transição t_j se existir um arco, ou zero, caso não exista nenhum arco entre p_i e t_j (MURATA, 1989). Os arcos presentes na matriz de entrada (I) são aqueles que saem do lugar e apontam para a transição correspondente, ou seja, são aqueles que entram nas transições. Os arcos na matriz de saída (O) são aqueles que partem da transição e apontam para o lugar, ou seja, saem das transições. M é um conjunto de números naturais que representa a quantidade de fichas nos lugares, também conhecido como marcação dos lugares. M_0 indica uma marcação inicial, isto é, uma distribuição de fichas nos lugares que caracteriza o estado inicial da rede ((HAN et al., 2008), (MURATA, 1989),(LI; ZHOU, 2008),(XU et al., 2007),(YOO; JEONG; CHO, 2010),(ZURAWSKI; ZHOU, 1994)) enquanto M_p indica o número de fichas no lugar p. Uma transição t está habilitada se $\forall p \in P, M(p) \geq I(p,t)$ e, uma vez habilitada, pode ser disparada. O disparo de uma transição altera a marcação M da rede produzindo uma marcação M', tal que $M'_p = M_p - I(p,t) + O(p,t) \forall p \in P$.

A figura 1 apresenta um exemplo de rede de Petri exemplificando como uma comunicação simples entre 2 servidores poderia ser modelada.

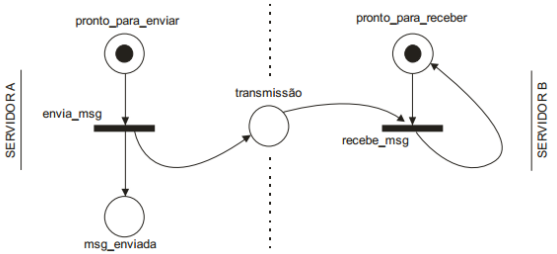


Figura 1: Exemplo de redes de Petri representando a comunicação entre dois servidores. Fonte:(OLIVEIRA, 2006)

Outro tipo de rede de Petri, o qual será abordado neste trabalho, é a rede de Petri temporizada. Uma rede de Petri temporizada é, estruturalmente, igual à rede de Petri ordinária, exceto que adiciona a noção de tempo associada às transições. Uma rede de Petri temporizada é uma tupla $R = (P, T, I, O, C)$ onde P, T, I, O possuem as mesma definições já estabelecidas para a rede de Petri ordinária, enquanto $C : T \rightarrow \mathbb{N}x\mathbb{N}$ é uma função que associa um intervalo de tempo (min,max), tal que min$\geq$ 0 e max $\geq$ min, a cada transição. Para essa rede, o disparo de uma transição só pode ocorrer após a passagem de min unidades de tempo após sua habilitação e deve ser disparada em até no máximo max unidades de tempo após sua habilitação. Outro aspecto importante para esta modalidade de rede de Petri é a memória de habilitação. Após o disparo de uma transição, todas as transições que estavam habilitadas na marcação anterior e que permanecem habilitadas na nova marcação mantém os valores de seus temporizadores. Apenas quando uma transição é desabilitada na nova marcação ou quando esta é disparada é que o seu tempo é perdido.(MACIEL; LINS; CUNHA, 1996)

2.1.2 Fusão de redes de Petri

Considerando um elemento de uma rede de Petri como um lugar ou uma transição, a fusão de elementos de diferentes redes de Petri permite, de forma sistemática e estruturada, a construção de redes de grandes dimensões a partir de módulos menores. Esta técnica permite a utilização de diferentes abordagens para a construção de sistemas representados em redes de Petri. Para a representação de um sistema qualquer em uma rede de Petri, redes menores que representam parte do fluxo de informação no sistema ou ainda que representam diferentes funcionalidades do sistema podem ser modeladas inicialmente e então fusionadas. Isto significa que, a partir de pequenas redes de Petri, as quais podem ser facilmente analisadas, pode-se construir uma rede mais complexa que oferece as características de cada uma das redes iniciais (GóIS, 2000).

A figura 2 apresenta duas redes e a fusão dessas redes utilizando a técnica de fusão de lugar. Os lugares P são elementos do mesmo tipo pertencentes às redes iniciais R' e R" e estão envolvidos no processo de fusão. Nota-se que os elementos do mesmo tipo desaparecem na fusão, dando origem a um novo elemento na rede final R. A fusão de transição ocorre da mesma maneira, exceto que os elementos envolvidos na fusão

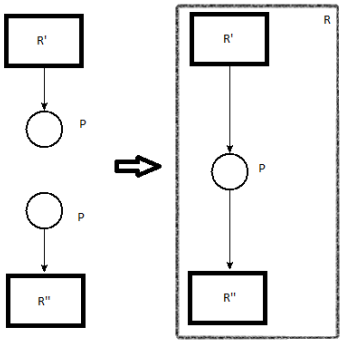


Figura 2: Exemplo representando a fusão entre duas redes de Petri Genéricas

são transições e não lugares.

2.1.3 Observadores baseados em Redes de Petri

Observadores são normalmente utilizados objetivando a verificação da alcançabilidade de um sistema e, quando utilizados, devem ser não intrusivos, ou seja, não devem alterar o sistema original inserindo novos dados ou influindo no comportamento do sistema sendo observado. Em sistemas modelados, os observadores são, normalmente, construídos com a mesma abstração utilizada para os sistemas modelados e são comuns na área de verificação formal de sistemas (ABID et al., 2011), sendo sua função observar transições ou estados de um sistema.

2.2 VERIFICAÇÃO FORMAL POR *MODEL CHECKING*

Model checking refere-se a um conjunto de técnicas para a análise automática de sistemas reativos. Pequenos erros no desenvolvimento de sistemas críticos que utilizam técnicas de simulação e testes convencionais podem ser (e têm sido) encontrados (MERZ, 2001) através da utilização de métodos formais. Como é relativamente simples de utilizar, a abordagem que envolve verificação vem sendo adotada como um procedimento padrão para a garantia da qualidade de sistemas reativos. (MERZ, 2001)

Uma estrutura de Kripke é uma variação de automato não-determinístico proposto por Saul Kripke, utilizado no model checking para representar o comportamento de um sistema. É um modelo matemático que captura a idéia de uma máquina de computação, sem adição de complexidades desnecessárias. É basicamente um grafo cujos nós representam os estados alcançáveis do sistema e cujas arestas representam as transições de estado. A função de marcação mapeia cada nó a um conjunto de propriedades que existem em um estado correspondente. Lógicas temporais são tradicionalmente interpretados em termos de estruturas de Kripke

As entradas para um verificador de modelos são modelos (geralmente finitos), na forma de estrutura de Kripke, do sistema a ser analisado e também as propriedades a serem verificadas, geralmente expressas como fórmulas de lógica temporal , que vão ser verificadas do modelo . O verificador de modelo ou confirma que as propriedades, de fato, existem no modelo ou afirma que essas são violadas. Neste último caso , ele fornece um contra-exemplo, na forma de uma execução de sistema que viola a propriedade. O usuário deve então decidir se o contra-exemplo é um condição do modelo (causada talvez por uma super simplificação do modelo fornecido) e, talvez, fornecer um modelo melhor para o verificador, ou se corresponde a um funcionamento do sistema real e rever o projeto do sistema para excluir o erro. Na prática um terceiro resultado possível de verificação do modelo é que a análise exige recursos superiores aos disponíveis(memória, por exemplo). Nesse caso , pode ser possível analisar uma representação mais superficial do modelo do sistema ou aplicar otimizações que fazem a análise viável .

Execuções bem sucedidas do verificador de modelos só podem estabelecer que certa propriedade é garantida no modelo abstrato e não no "mundo real"ou sistema de hardware "concreto"ou programa. A inserção de erros durante a construção do modelo abstrato é bem comum e procedimentos devem sempre ser adotados (análises de código por exemplo) para garantir que a representação abstrata representa fielmente o comportamento do sistema real na medida do necessário para a análise das propriedades consideradas. O verificador pode ser de alguma ajuda ao estabelecer que o modelo abstrato, pelo menos, não exclui certas execuções.

2.3 ABORDAGENS FORMAIS NO DESENVOLVIMENTO OU VERIFICAÇÃO DE PROGRAMAS DE CLP

Nesta seção serão abordados alguns trabalhos que se baseiam na utilização de representações formais para a construção de um programa de de CLP ou sua verificação. A utilização de representações formais confere ao desenvolvimento ou verificação de teste de programas de CLP uma base matemática para sua execução. Os trabalhos que se utilizam de tais representações têm como objetivo garantir certas propriedades aos programas sendo desenvolvidos ou validados. Cada trabalho apresentado possui suas próprias particularidades e serão explicados individualmente. Dentre as abordagens abordadas, serão utilizados os conceitos apresentados nesse capítulo.

2.3.1 Diagnóstico e tratamento de falhas em sistemas críticos utilizando métodos formais

Em (JúNIOR, 2011), uma abordagem para a geração de algoritmos de controle é proposta. O princípio do trabalho é a concepção de um conjunto de procedimentos sistematizados para desenvolvimento e validação de algoritmos de controle, a partir da modelagem de funções para diagnóstico e tratamento de falhas através da análise de documentos. A abordagem deste trabalho caracteriza-se pela construção e análise de modelos de diagnóstico e tratamento de falhas em sistemas críticos e é dividia em quatro etapas:

- modelagem.
- análise.
- geração dos programas de controle
- testes finais de aceitação.

A etapa de modelagem é subdividida em dois estágios complementares: a modelagem do diagnóstico de falhas e a modelagem do tratamento e das falhas. A sistemática proposta utiliza como fonte de dados de entrada o relatório de análise de riscos (HAZOP) e as informações presentes em um dos documentos de automação (matriz causa e efeito). A partir dos dados de entrada, no estágio da modelagem do diagnóstico de falhas, são geradas redes Bayesianas para o diagnóstico de falhas e, após sua obtenção, essas são traduzidas para

redes de Petri. Em outro estágio, a modelagem do tratamento e das falhas, baseando-se nas falhas possíveis para o sistema, são geradas funções instrumentadas de segurança, as quais modelam funções responsáveis por lidar com essas possíveis falhas(tratamento das falhas). Essas funções são modeladas utilizando-se redes de Petri. Neste momento, existem dois grupos de modelos de redes de Petri. O responsável pelo diagnóstico e o responsável pelo tratamento. De acordo com informações contidas nos documentos, as redes de Petri dos dois grupos são relacionadas e integradas.

A etapa de análise investiga a interação entre os modelos de diagnóstico e de tratamento de falhas críticas uma vez que integrados. Para essa atividade, ferramentas de simulação de redes de Petri são utilizadas. Na etapa de geração, tem-se a geração dos programas de controle baseados nos modelos de diagnóstico e de tratamento integrados na etapa prévia. Estes programas são gerados de acordo com a norma IEC 61131-3. Finalmente, na etapa de aceitação, os testes finais de validação são executados a fim de validar se as funções criadas atendem às especificações técnicas de acordo com a norma IEC 61511. Informações mais detalhadas sobre este trabalho podem ser encontradas em (JúNIOR, 2011) A figura 3 apresenta a sistemática proposta.

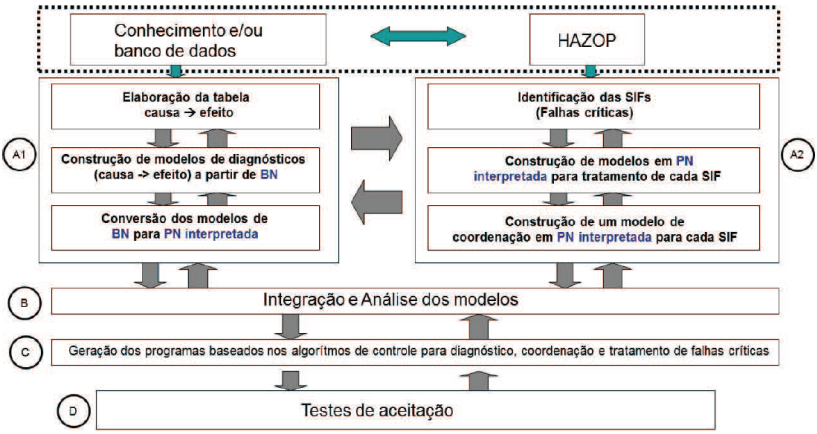


Figura 3: Sistemática apresentada em (JúNIOR, 2011)

2.3.2 Uma abordagem baseada em engenharia dirigida a modelos para a verificação de programas de CLP

Este trabalho(FARINES; QUEIROZ; CARPES, 2011) se insere no contexto do projeto TOPCASED (VERNADAT et al., 2006), que visa o desenvolvimento de um ambiente para sistemas embarcados , com base em engenharia dirigida a modelos (MDE) . A plataforma apresentada, a TOPCASED, integra diferentes ferramentas de verificação como por exemplo TINA (BERTHOMIEU, 2004) e permite a utilização de várias linguagens de modelagem de usuário. O FIACRE, a linguagem (BERTHOMIEU et al., 2008) concebida no projeto TOPCASED serve como um formato intermediário entre as linguagens de usuário e de verificação de modelos formais.

Primeiramente, um metamodelo para um diagrama *Ladder* é proposto. Regras de transformação entre elementos básicos e blocos de função *Ladder* para FIACRE foram definidos. O resultado foi então uma semântica Fiacre para programas escritos em *ladder* . O modelo *ladder* é traduzido para FIACRE a partir dessas regras, de tal forma que cada programa *ladder* concebido em conformidade com o metamodelo *ladder* pode ser automaticamente traduzido para uma representação FIACRE. A partir da linguagem FIACRE, a ferramenta de compilação existente FRAC pode gerar um modelo de Transição Temporário(TTS) usado pelo conjunto de ferramentas TINA para verificação. Em seguida, as propriedades referentes ao sistema , expressas em LTL , podem ser verificadas no modelo TTS obtido usando o *model checking* disponível no TINA. MDE garante que a verificação é realizada em um modelo que é construído automaticamente a partir do programa *ladder* e em conformidade com regras de transformação definidas, eliminando, assim, possíveis erros inseridos pela transformação manual dos modelos.

Este trabalho encaixa-se no ambiente TOPCASED e utiliza uma metodologia similar e algumas de suas ferramentas para construir um conjunto de ferramentas de verificação para os programas de PLC escritos com linguagens definidas na IEC 61131-3. Devido à diversidade de linguagens existentes (como UML, SysML ou AADL) e de ferramentas de verificação disponíveis com seus formalismos próprios (como redes de Petri , autômatos ou álgebra de processos), o ambiente TOPCASED define o idioma FIACRE como um passo intermediário na transformação do modelo. O conjunto de ferramentas de verificação tem, portanto, como base, dois níveis de transformação: um nível de uma linguagem para o idioma intermediário FIACRE e, em seguida,

um outro de FIACRE às linguagens de verificação formal. Dentre os benefícios da utilização do FIACRE apresentados pelo trabalho, tem-se: a redução da diferença semântica entre linguagens de alto nível e formalismos de baixo nível para ferramentas de verificação e a definição de uma semântica exclusiva para diferentes conjuntos de ferramentas de verificação, o que torna mais fácil fácil a introdução de novas linguagens de alto nível e novas ferramentas de verificação. No caso do trabalho em questão, o FIACRE facilita o uso do conjunto de ferramentas de verificação TINA para a linguagem *Ladder* para CLP, sem necessidade de tradução direta de representações de alto formalismo como Redes de Petri em linguagens de baixo nível como *Ladder*.

A ferramenta desenvolvida nesse projeto é exemplificada na figura 4 e consiste do seguinte: um editor de PLC permite a criação de programas *Ladder* e os salva em um arquivo XML. Um tradutor - construído usando abordagem MDE e escrito em ATL (JOUAULT; KURTEV, 2006) transforma o XML resultante em FIACRE. Ao código FIACRE resultante, se adiciona o modelo da planta também em FIACRE e obtém-se assim o modelo FIACRE completo. Enfim, esse modelo completo é compilado para TTS utilizando o compilador FRAC (parte do TOPCASED) e pode, através do conjunto de ferramentas TINA, ser verificado.

2.3.3 Um conjunto de ferramentas para verificação de modelo de programa de CLP

A abordagem apresentada em (PAKONEN et al., 2013) consiste na especificação manual de um código verificador para cada bloco de função elementar utilizado em um determinado programa. A justificativa apresentada para a definição de cada bloco manualmente foi a seguinte:

- Em vez de usar linguagens padrão, conforme especificado pela IEC 61131-3, muitos dos principais fabricantes de CLP usam blocos de funções específicas a cada fornecedor.
- Muitas vezes, os fornecedores não estão dispostos a divulgar a algoritmos de implementação reais para o blocos de funções elementares (caixa preta) por os considerarem uma propriedade intelectual da companhia. Apenas a descrição funcional é fornecida aos clientes.
- Mesmo que a lógica interna dos blocos fossem reveladas (caixa

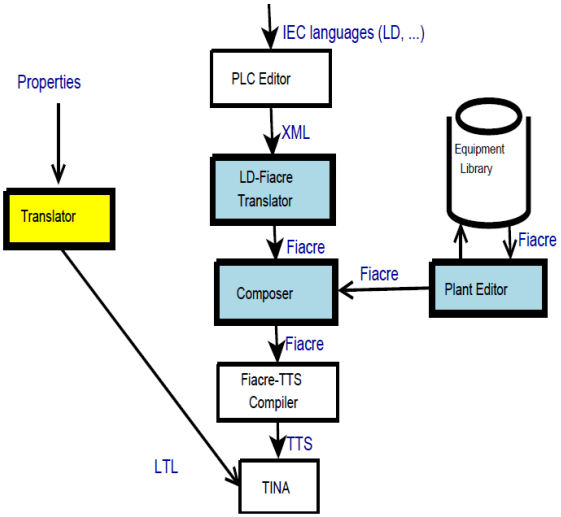


Figura 4: Ferramenta desenvolvida em (FARINES; QUEIROZ; CARPES, 2011)

branca), é provável que a implementação desses algoritmos se dê em linguagens como C ou Java, o que não costuma permitir representação direta em algumas ferramentas de verificação.

O ponto de partida na metodologia proposta neste trabalho é a descrição funcional dos blocos de função elementar. O código para *model checking* de cada bloco é então escrito e a verificação do modelo de cada bloco pode ser utilizada para assegurar que o código está correto. As propriedades a serem verificadas são obtidas a partir da descrição funcional de cada bloco.

O verificador utilizado neste trabalho foi o NuSMV, um verificador de modelo simbólico baseado em BDD que permite representar ambos os sistemas de estados finitos síncronos e assíncronos, com descrição de tempo discreto e as propriedades podem ser especificados usando LTL ou CTL (BAIER; KATOEN et al., 2008). A ferramenta permite apenas expressões muito básicas e tipos simples de variáveis (boleana, inteira, enumeração, matriz), o que significa que os blocos de funções de algoritmos complexos (por exemplo, PID) não podem ser modelados com precisão razoável. Lógicas complexas de controle devem ser, portanto, abstraídas, ou completamente omitidas da verificação.

Foi desenvolvida uma ferramenta de verificação de modelos para Simantics (KARHELA; VILLBERG; NIEMSTÖ, 2012) (uma ferramenta que oferece um ambiente de integração entre ferramentas de simulação e de engenharia). Através da ferramenta desenvolvida, pôde-se construir ou reproduzir um programa em FBD e transformar esse diagrama para um arquivo de entrada para o NuSMV. Através de uma entrada de texto, deve-se introduzir o código do bloco de função elementar além da interface para o mesmo(entradas e saídas). Através da interface gráfica da ferramenta, constrói-se uma representação FBD para o programa a ser verificado e, com esse programa FBD, gera-se um arquivo de entrada para NuSMV contendo todos os elementos necessários do modelo - código para os blocos elementares utilizadas , bem como as ligações entre blocos. As propriedades a serem verificadas são introduzidas no NuSMV através de texto.

2.4 VALIDAÇÃO DE PROGRAMAS ATRAVÉS DE TESTES

A atividade de testar consiste em uma análise dinâmica do programa produzido e é uma atividade relevante na identificação e eliminação de possíveis erros contidos no programa. A atividade de realizar testes em programas prontos é de grande utilização em sistemas onde a operação não é crítica, uma vez que seu mau funcionamento não resulta em grandes perdas ou desastres. Por ser a abordagem mais natural para se checar o correto funcionamento de um programa, é muito utilizada na indústria como uma das etapas finais antes da entrega de um programa de automação. Mesmo algumas plantas críticas têm seu programa testado baseando-se em especificações iniciais e conhecimento interno da equipe de engenharia responsável pelo projeto.

A realização de testes de programa envolve basicamente 4 etapas (BEIZER, 1990) (MYERS, 1979) (MALDONADO, 1991) (PRESSMAN, 1997):

- planejamento de testes
- projeto de casos de teste
- execução
- avaliação dos resultados dos testes

Cada uma dessas quatro etapas, que devem ser realizadas durante o desenvolvimento do programa, podem ser divididas em três

fases de testes: testes de unidade(teste de cada módulo do programa), testes de integração(testa o interfaceamento de cada módulo) e teste de sistema(testa se o sistema atende aos requisitos pré-estabelecidos) (BEIZER, 1990).

Na realização de testes, é importante a definição de um conjunto de casos de teste, quando for impraticável a realização de todos os testes possíveis. Como o teste tem por principal finalidade a identificação de erros(MYERS, 1979), um bom conjunto de casos de teste é o conjunto que consegue fazer o programa em teste falhar. Para a definição do conjunto de casos de teste, existem alguns critérios de teste. Os principais são:

- funcional: o requisitos do teste provêm da especificação do programa
- estrutural: os requisitos são derivados a partir de uma implementação específica
- baseado em erros: os requisitos são estabelecidos a partir de erros típicos realizados durante o processo de desenvolvimento do programa

Os testes podem ainda ser divididos em três tipos principais:

- Testes de caixa-preta ou teste baseado em especificação tem por objetivo inferir se o programa atinge os requisitos funcionais e não funcionais . Dentre os testes de caixa-preta, pode-se citar particionamento em classe de equivalência (BEIZER, 1990), análise do valor limite, grafo de causa-efeito (BEIZER, 1990) e teste baseado em estado (MCGREGOR, 1994) (HOFFMAN; STROOPER, 1993)
- Teste de caixa-branca ou teste baseado em programa trabalha com a inspeção do código fonte e a seleção de casos que executem determinada parte do código(PERRY; KAISER, 1990)
- Teste de caixa cinza implica ter conhecimento de estruturas de dados e algoritmos internos para fins de concepção de testes, enquanto executa desses testes no nível de caixa-preta. O testador não é obrigado a ter pleno acesso ao código-fonte do software(PATTON, 2001).

Ao realizar testes de programa, é importante destacar alguns pontos:

- A realização de testes pode desprender muito tempo de projeto

- Testes nunca podem ser considerados exaustivos
- É necessário a correta interpretação de resultados para se garantir que um erro não seja considerado como comportamento correto de programa
- O conjunto de testes realizados pode ser parcial e dependente daquele que escolhe a rotina de testes

Ao se realizar testes, deve-se levar em consideração suas limitações para evitar a ineficácia desses. Quanto ao tempo gasto para a execução de testes, existem algumas abordagens para se melhorar sua eficiência. A seguir serão apresentadas algumas abordagens utilizadas durante a fase de testes que podem reduzir o tempo desta fase de projeto ou ainda melhorar sua eficácia na busca por falhas.

Particionamento de equivalência (também chamado de classe de equivalência ou ECP (BURNSTEIN, 2003)) é uma técnica de teste de programa que divide os dados de entrada de um programa em partições de dados equivalentes a partir do qual os casos de teste podem ser derivados. Em princípio, os casos de teste são projetados para cobrir cada partição pelo menos uma vez. Esta técnica tenta definir casos de teste para descobrir classes de erros, reduzindo assim o número total de casos de teste que devem ser desenvolvidas. Uma vantagem desta abordagem é a redução no tempo necessário para o ensaio de um programa, devido ao menor número de casos de teste.

Análise do valor limite é uma técnica de teste de programa em que os testes são projetados para incluir representantes de valores de limite(MALDONADO et al., 2004). Dado que um conjunto de vetores de teste para testar o sistema, uma topologia pode ser definida nesse conjunto. Esses elementos que pertencem à mesma classe de equivalência, tal como definido pela teoria particionamento de equivalência constrói a base (topologia). Dado que os conjuntos de base são vizinhos, existe uma fronteira entre eles. Os elementos de teste em ambos os lados da fronteira são chamados valores limite. Na prática, isso exige que os elementos de teste possam ser ordenados, e que os parâmetros individuais seguem algum tipo de ordem (ou de ordem parcial ou total).

Testes *all-pairs* ou teste *pairwise* é um método combinatório de teste de programa que, para cada par de parâmetros de entrada para um sistema, testa todas as possíveis combinações distintas de tais parâmetros. Usando vetores de teste escolhidos dentre todos os parâmetros, isso pode ser feito muito mais rapidamente do que uma procura exaustiva de todas as combinações de todos os parâmetros, por uma "paralelização"dos testes de pares de parâmetros. O raciocínio por

trás dessa metodologia de testes é a seguinte: os erros mais simples em um programa são geralmente acionados por um único parâmetro de entrada. A próxima categoria mais simples de erros é constituído por aqueles dependentes de interações entre os pares de parâmetros, que podem ser capturados com teste *all-pairs* (BLACK, 2007). Problemas envolvendo interações entre três ou mais parâmetros são cada vez menos comuns(KUHN; WALLACE; GALLO, 2004),e, ao mesmo enquanto cada vez mais caro para se encontrar através de testes exaustivos, que tem como limite máximo, o teste exaustivo de todas as entradas possíveis (KUHN; KACKER; LEI, 2010).

A seguir, será apresentado um trabalho desenvolvido para a realização de testes em programas de sistemas instrumentados de segurança.

2.4.1 Testes em sistemas instrumentados de segurança através de equivalência de classes

A metodologia definida em (OLIVEIRA LEANDRO DIAS DA SILVA, 2012) consiste em quatro etapas ilustradas na figura 5 e pode ser explicada da seguinte maneira:

- A especificação do sistema é lida e traduzida em um modelo de autômato temporizado
- O modelo é usado para gerar automaticamente um conjunto de casos de teste
- Para cada caso de teste, os valores de entrada e de teste são enviados ao CLP através de OPC
- A rotina no CLP é executada gerando um conjunto de valores de saída
- A saída do CLP é então comparado com o resultado do modelo obtido e um veredito é obtido

Os casos de teste são gerados tendo em conta o ciclo de varredura do PLC. Cada caso de teste é gerado para ser executado em um ciclo de varredura. Um caso de teste é uma tupla C = I, O, T :

- I é um conjunto de valores booleanos às entradas correspondentes variável da especificação do sistema;

- O é um conjunto de valores booleanos correspondentes às variáveis de saída da especificação que são obtidos após o processamento de I;
- T é o traço, ou seja, as sequências de eventos de sincronização executados para obter O de I.

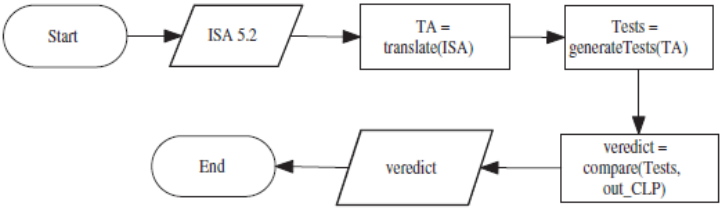


Figura 5: Metodologia de testes utilizando equivalência de classes

Ao gerar os casos de teste , algumas das possíveis combinações de valores das variáveis de entrada são selecionadas para serem utilizadas. Dois tipos de falhas foram consideradas ao selecionar os valores de entrada: falhas que afetam uma restrição de tempo e as falhas que afetam a saída do sistema. A seleção foi feita através de classes de equivalência,e foram removidas combinações que geram casos de teste redundantes. Dois casos de teste são redundantes se forem consideradas semelhantes e não agregam valor para o conjunto de teste . Eles não detectam novos erros e eles não fornecem uma melhor cobertura de teste do sistema em análise.A remoção de testes redundantes teve como objetivo a diminuição do tempo de realização de testes.

A seleção de combinações de entrada depende dos elementos da especificação do sistema . Para cada variável de saída, duas entradas são selecionadas, uma que a ativa e outra que a desativa . Este tipo de teste tem como objetivo detectar falhas que afetam a saída do sistema. Se o sistema tiver elementos temporizados, todas as transições de cada temporizador são executadas pelo menos uma vez, e as entradas adequadas devem ser selecionadas. A verificação dos temporizadores é feita com base na entrada de cada temporizador. Portanto, para cada temporizador, uma combinação de valores de entrada deve levar a entrada do temporizador para verdadeiro e outra para falso. Este tipo de teste objetiva a detecção de falhas que afetam restrições temporais.

Este trabalho propõe, então, a utilização de um número mínimo de entradas do CLP, o suficiente para ativar todas as saídas do programa de CLP e também todos os temporizadores do programa. Os

casos de teste são selecionados a partir dos modelos de autômatos. Essa abordagem pode trazer problemas uma vez que considerar uma única entrada de um grupo de entradas não garante o correto funcionamento de todas as entradas. Ainda, assim como explicado no trabalho, os testes devem poder ser realizados em um ciclo de CLP e, desta maneira, exclui qualquer teste que inclua uma possível sequência de entradas assim como uma entrada que resulta em um sequenciamento nas saídas.

2.5 COMENTÁRIOS FINAIS

Neste capítulo foi realizada uma revisão bibliográfica sobre redes de Petri, verificação formal através de *model-checking* e também sobre a realização de testes em programa. Foram apresentados trabalhos que utilizam abordagens formais para o desenvolvimento e validação de programas de automação industrial. No próximo capítulo, será apresentada a abordagem utilizada atualmente pela Petrobras para o desenvolvimento e validação dos programas de automação desenvolvidos em seus projetos.

3 METODOLOGIA DE DESENVOLVIMENTO DO SOFTWARE DE AUTOMAÇÃO NA INDÚSTRIA DE PETRÓLEO E GÁS

A metodologia de desenvolvimento de projetos de automação atualmente utilizada em empresas de petróleo e gás, como a Petrobras consiste em uma sequência de etapas que partem de definições básicas da planta a ser automatizada e culmina com o desenvolvimento do programa utilizado para automação de CLP. Durante o projeto, uma gama de documentos, cada qual contendo informações relevantes a uma ou várias etapas do projeto, é criada. Cada etapa dessa metodologia possui entradas e saídas na forma de documentos, sendo que os documentos na entrada de uma determinada etapa são aqueles que são necessários para a geração dos documentos de saída. Neste capítulo, a atual metodologia de desenvolvimento e os documentos serão explicados. Para a exemplificação da metodologia e dos documentos utilizados na metodologia atualmente encontrada, uma planta exemplo fornecida pela Petrobras sera utilizada. Trata-se de uma planta de um forno industrial e seu funcionamento é melhor explicado a seguir.

3.1 DESCRIÇÃO DE UM FORNO INDUSTRIAL PRESENTE EM UMA UNIDADE DE HIDROTATAMENTO DE DIESEL

Para exemplificação da metodologia atualmente utilizada por empresas como a Petrobras, será utilizada uma planta de um forno industrial. Esta planta será utilizada também, em capítulos posteriores para a validação da metodologia proposta neste documento. Esta seção não possui como objetivo explicar detalhadamente cada aspecto do forno, mas sim, esclarecer o funcionamento dessa planta para que possa ser utilizada como exemplo. Maiores detalhes podem ser encontrados em (SILVA, 2009).

O princípio de um forno é a troca de calor e isso é obtido pela queima de um combustível, o que libera gases quentes e, esses, em contato com uma serpentina em seu interior, fornecem calor a um fluído. Um forno possui três partes principais:

- câmara de radiação: região onde estão os queimadores (responsáveis pela queima do combustível) e uma parte da serpentina para troca de calor
- zona de convecção: região onde se encontra a outra parte da

serpentina

- chaminé: trecho por onde os gases da combustão são lançados à atmosfera

Figura 6: Forno em uma planta industrial. Fonte: (SILVA, 2009)

A figura 6 ilustra um forno industrial. Dado o funcionamento mais básico do forno, é necessário entender a composição sua básica, seus instrumentos e sua interação no processo de geração e troca de calor. Alguns dos instrumentos pertinentes a um forno são:

- queimadores: são os instrumentos responsáveis pela liberação de calor necessário através da queima de combustíveis

- piloto: responsáveis pela chama piloto de cada queimador. A figura 7 ilustra o conjunto queimador e piloto
- vaso de *knock-out* é um tanque localizados a montante dos queimadores responsáveis pela retirada de umidade presente no gás para a sua melhor queima
- válvulas de bloqueio: duas válvulas anteriores ao queimador e posteriores ao vaso de *knock-out* que possuem o objetivo de não permitir o vazamento de gases
- válvula de ventilação: presente entra as válvulas de bloqueio e responsável por eliminar qualquer pressão que possa existir antes da segunda válvula de bloqueio causada por qualquer vazamento na primeira válvula de bloqueio.
- vapor de abafamento: vapor utilizado na purga do sistema para expulsão de gases indesejados que poderiam causar explosões
- damper: registro instalado na chaminé para controle de vazão de gases ou tiragem dos gases de combustão

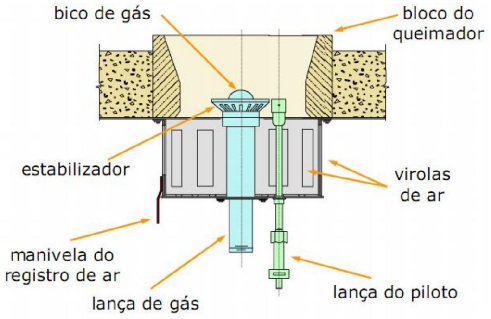


Figura 7: Esquema de um queimador e seu piloto.Fonte: (SILVA, 2009)

Os componentes básicos para o entendimento foram descritos, além disso, o forno possui um sistema para reaproveitamento de gás residual. Por esse sistema, gás de purga da unidade de geração de hidrogênio são reaproveitados como gás combustível no forno. Uma unidade de geração de hidrogênio recebe uma mistura de gás que possui H_2, CO_2 e CH_4 e separa o H_2 dos demais gases e os envia para a unidade de hidrotratamento. A figura 8 ilustra os equipamentos descritos anteriormente e sua interação juntamente com uma unidade de hidrotratamento.

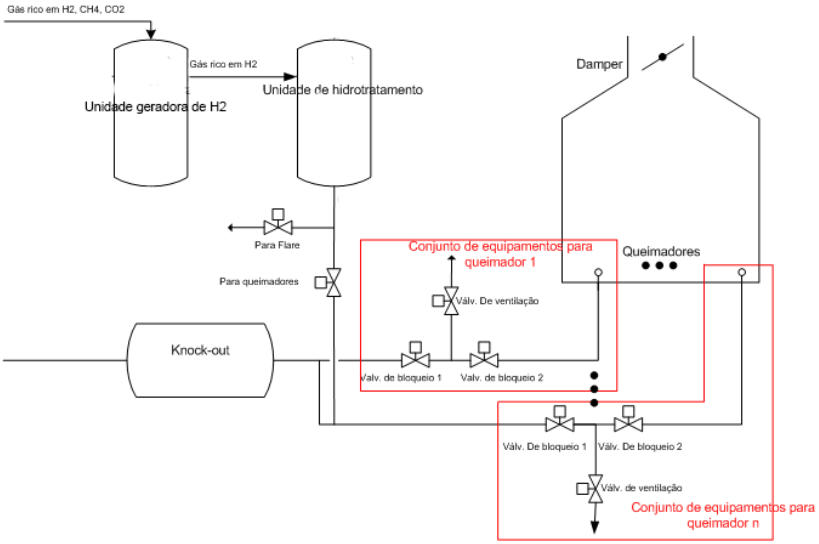


Figura 8: Esquema de um forno industrial com uma unidade de hidrotratamento

3.2 METODOLOGIA DE DESENVOLVIMENTO DE PROJETO DA INDÚSTRIA DE PETRÓLEO E GÁS

A metodologia implementada atualmente para o desenvolvimento de projeto da indústria de petróleo e gás inclui a criação de uma série de documentos numa determinada ordem e cada documento provê informações importantes para o desenvolvimento do projeto. Para melhor explicar esta metodologia, primeiramente, serão enumerados os tipos de documentos presentes em um projeto e as informações contidas em cada um, e, em seguida, será apresentada a metodologia de desenvolvimento de projeto, ressaltando os documentos diretamente relacionados com o programa de software desenvolvido para o CLP.

De acordo com normas internas à Petrobras (COMISSãO DE NORMAS TéCNICAS, 2002), alguns documentos são considerados essenciais para qualquer projeto. Os documentos que todo projeto deve possuir são:

- fluxograma de processo: o fluxograma de processo deve conter a representação simplificada das malhas de controle, identificando a variável, função, localização e conter, ainda, as válvulas de con-

trole, segurança e alívio. A figura 9 exemplifica um fluxograma de uma vaso de *knock-out* .

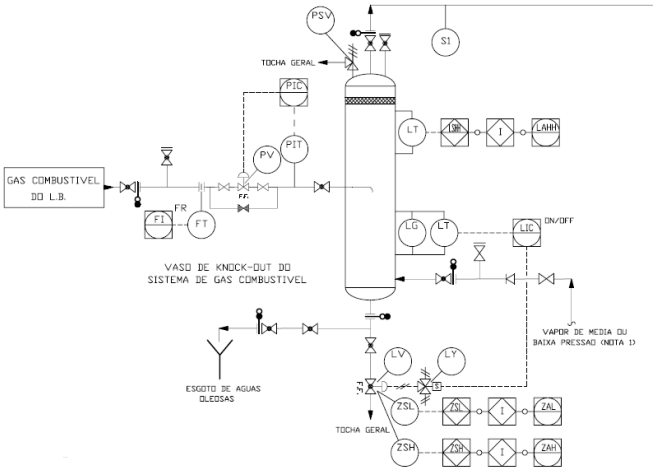


Figura 9: Exemplo de uma parte de um fluxograma de processo referente a um vaso de *knock-out*

- folhas de dados de processo: devem conter informações básicas de processo que permitam a correta seleção e dimensionamento dos instrumentos.
- matriz causa e efeito: mostra o inter-relacionamento entre os eventos anormais possíveis de ocorrer durante a operação normal da planta ou de um equipamento em particular e as ações que devem ser tomadas pelo sistema de segurança, como também manobras operacionais específicas. A figura 10 apresenta algumas relações referentes à planta já apresentada.
- lista de instrumentos preliminar: é um guia de acesso aos documentos de instrumentação do projeto básico. Deve conter todos os instrumentos da unidade, localizando-os por fluxogramas e dando informações básicas relativas aos instrumentos como: sua situação física (campo, painel, função em sistema digital, etc.). O documento que contém os dados para especificá-lo e seu tipo básico (placa, venturi, termopar, etc.)
- fluxograma de engenharia preliminar: Quanto à instrumentação, o Fluxograma de Engenharia deve conter as malhas de controle

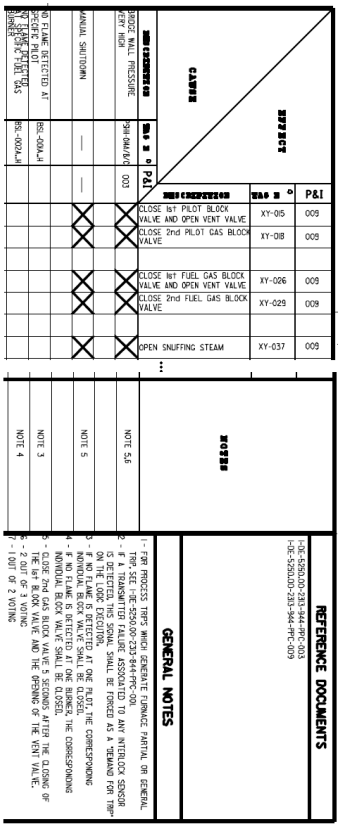


Figura 10: Exemplo de relação contida na matriz causa e efeito para a planta apresentada

explicitando as funções dos instrumentos, sua identificação preliminar usada como referência no projeto básico e a localização de cada instrumento. Deve conter também, notas explicativas que explicitem recomendações e exigências do projeto básico quanto à instalação, à locação detalhada ou outros requisitos ligados à instrumentação.

Alguns processos, por serem processos críticos necessitam de alguns documentos extras. Eles são:

- especificação para projeto de detalhamento: deve ser emitida na fase inicial do projeto e apresentar os fundamentos básicos de

instrumentação que devem orientar a execução das especificações e dos demais documentos do projeto de detalhamento.

- memorial descritivo: deve conter informações básicas que permitam a completa especificação de equipamentos e instrumentos para os diversos sistemas de instrumentação além de sequenciamentos presentes na planta. A figura 11 apresenta um exemplo de um sequenciamento, representado por um fluxograma, contido no memorial descritivo da planta utilizada como exemplo.

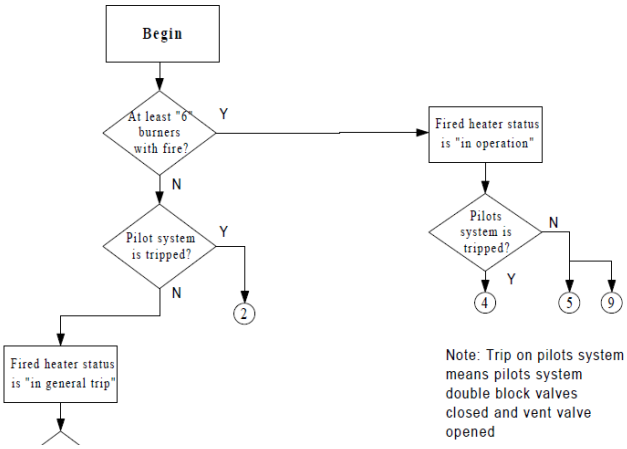


Figura 11: Exemplo de sequenciamento encontrado em memorial descritivo

- diagrama lógico: Deve ser baseado nos memoriais descritivos para os sistemas de proteção, intertravamento e sinalização-alarme e deve, ainda, ser construindo utilizando álgebra de Boole. Este documento tem por finalidade representar toda a lógica de intertravamento do projeto e pode ser pensado como uma prévia do programa final do CLP. A figura 12 apresenta um trecho um um diagrama lógico.

- desenhos de instalação para itens referentes a tecnologia própria indispensável ao funcionamento do processo: neste documento devem ser somente emitidos aqueles desenhos, os quais referem-se a instalação de instrumentos especiais, com tecnologia própria, e que necessitam de informações destes documentos para o seu correto funcionamento.

- arranjo preliminar do painel de controle: Deve mostrar a distribuição dos instrumentos, alarmes, chaves, lâmpadas, espaços reservados e demais dispositivos instalados na parte frontal do painel de controle com suas respectivas identificações.

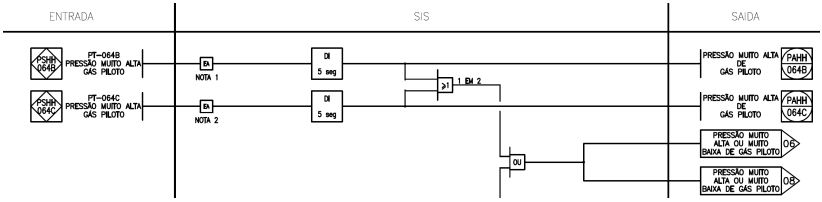


Figura 12: Exemplo de diagrama lógico

- manual de operação.
- arquitetura de instrumentação.
- lista de variáveis (entradas e saídas);
- folha de dados de instrumentação para instrumentos especiais: quando o projeto envolver instrumentos especiais como sistemas termométricos especiais, sistemas de nível de uso pouco comum (como sistema radioativos) ou instrumentos cuja especificação deve ser específica por exigências de processo, estes instrumentos devem ter seus dados definidos no Projeto Básico. As folhas de dados para este caso devem conter as informações necessárias ao desenvolvimento do Projeto Executivo e à compra de instrumento.
- especificação técnica para sistemas especiais: Quando o projeto envolve sistemas pouco comuns ou que incorporem exigências escritas do processo, estes devem ter sua especificação preliminar elaborada no projeto Básico. Tal especificação deve conter as exigências operacionais e construtivas fixadas pelo processo e todas as informações que se vincularem à exigência tecnológica do processo
- diagramas de controle avançado.
- descritivos das malhas de controle avançado: Deve conter explicações sobre o objetivo e forma de funcionamento das malhas de controle avançado, bem como explicitar as equações e

parâmetros a serem ajustados nas funções envolvidas nestas malhas.

- lista de variáveis calculadas.

3.3 METODOLOGIA PARA DESENVOLVIMENTO DO SOFTWARE DE CLP

Dentre os documentos descritos anteriormente, alguns estão mais ligados à criação do programa de intertravamento desenvolvido para o programa de CLP que outros. Dentro dos documentos disponíveis, pode-se citar:

- Fluxograma de processo
- Matriz Causa e Efeito
- Memorial descritivo
- Diagrama lógico

Ainda dentro dos documentos relacionados com o programa de CLP, existe uma documento criado exclusivamente para testar o programa de intertravamento e garantir que esse apresenta um comportamento esperado. Tal documento é o "Teste de aceitação de fábrica" e apresenta em forma textual como os testes devem ser realizados para que o programa de intertravamento seja aprovado.

É importante também ressaltar não somente quais os documentos utilizados num processo de automação ou quais são utilizados na geração de código para o CLP, mas também como se dá sua interação durante o processo de automação. Na metodologia de desenvolvimento da Petrobras, dentre os documentos citados na seção anterior, o primeiro documento a ser desenvolvido é o fluxograma de processo e, a partir das informações deste documento, a Matriz Causa e Efeito e o memorial descritivo são criados.

A geração das relações pertencentes à matriz causa e efeito é baseada em informações do fluxograma de processos (relação entre equipamentos destinados à segurança e sinais de sensores que devem ativar ou desativar tais equipamentos) em conjunto com os especialistas do projeto. Para o memorial descritivo, são utilizadas informações do fluxograma de processo e também conhecimento de especialistas do projeto para definição de especificações de equipamentos e sequenciamentos para a planta.

A informação conjunta dos dois últimos documentos (matriz causa e efeito e memorial descritivo) permite a criação do diagrama lógico e do teste de aceitação de fábrica. O diagrama lógico gera o programa de CLP e o TAF é utilizado durante a fase de testes do programa do CLP. O diagrama lógico contém as lógicas do futuro programa de CLP e, as informações que definem essas lógicas são obtidas das descrições encontradas no memorial descritivo assim como nas relações existentes na matriz causa e efeito. A figura 13 apresenta esquematicamente a geração dos documentos durante o processo de automação. Em geral, a programação propriamente dita do CLP é feita por uma empresa terceirizada e, após a entrega dos equipamentos utilizados na programação, a Petrobras realiza o teste de aceitação de fábrica para definir se o programa gravado no CLP é válido ou não. Caso seja encontrado algum erro, o CLP é retornado para reprogramação.

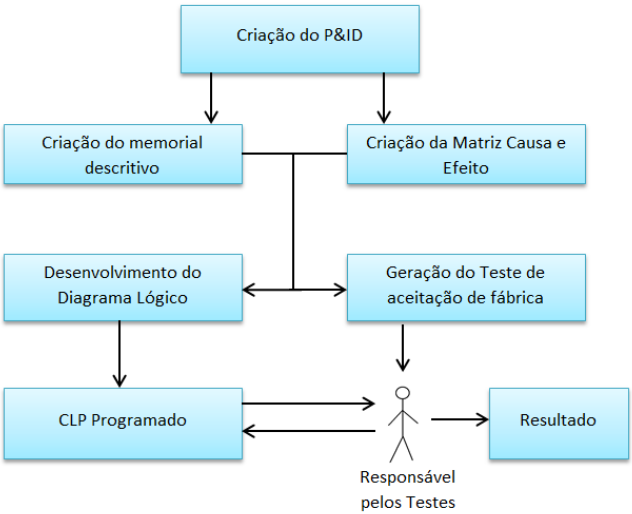


Figura 13: Metodologia de desenvolvimento de projetos na Petrobras

3.4 LIMITAÇÕES DA METODOLOGIA ATUAL

Na metodologia atual de desenvolvimento, o resultado final, isto é, o programa de CLP, é uma tradução direta das informações contidas no diagrama lógico e na matriz causa e efeito para uma lógica *Ladder*.

Os testes realizados em seguida são criados a partir de conhecimento prévio de engenheiros envolvidos no projeto além de informações dos documentos existentes.

O período reservado para a aprovação do CLP já programado é relativamente curto, o que resulta na necessidade de realização de vários testes em um curto espaço de tempo. Isso pode resultar em testes com resultados incorretos caso alguma especificação no documentos de testes não seja seguida corretamente. Ainda, os testes são realizados com o CLP já em mãos da Petrobras, e, caso algum erro seja encontrado, este é devolvido à terceirizada para correção. Esta abordagem pode provocar atrasos por exigir o translado do CLP entre terceirizada e Petrobras para cada teste, além de exigir o deslocamento de, pelo menos uma pessoa, para a realização dos testes durante as últimas etapas de um projeto de automação.

A criação de um teste de aceitação de fábrica deve conter, ao menos, uma quantidade mínima de testes para se validar um programa de CLP. Nem todos os casos de teste que um especialista pode conceber são inseridos no documento, afinal, o tempo para a realização dos testes é uma variável de grande importância e, cada teste extra, exige mais tempo para validação do programa de CLP. Esta abordagem portanto é um teste parcial, e os problemas de configuração do CLP vão aparecer durante a partida e a operação da planta, acarretando problemas de segurança ou paradas da produção, com altos custos.

3.5 COMENTÁRIOS FINAIS

Neste capítulo abordada-se a atual metodologia de desenvolvimento de projetos utilizada pela Petrobras. Apresenta-se os documentos utilizados nessa atual metodologia, sua interação para a geração do software de CLP e a realização de testes sobre tal CLP. Além disso, também apresenta-se limitações da metodologia como tempo desprendido e número de testes realizado No próximo capítulo, será apresentada uma nova metodologia baseada em redes de Petri para a geração e realização de testes baseada em um dos documentos do projeto.

4 PROPOSIÇÃO DE UMA NOVA METODOLOGIA PARA GERAÇÃO E REALIZAÇÃO DE TESTES

No capítulo anterior, foi mostrada a metodologia atual de desenvolvimento de programas em vigor na Petrobras. Neste capítulo, será abordada uma nova metodologia proposta para a realização de testes baseada em documentos de projeto e sua aplicação. Num primeiro momento, a metodologia será explicada a partir de uma visão geral e em seguida os passos envolvidos com essa metodologia serão detalhados.

4.1 VISÃO GERAL DA METODOLOGIA E SUA INSERÇÃO NO DESENVOLVIMENTO DO PROJETO

O propósito da metodologia a ser apresentada neste capítulo é ser integrada à metodologia de projeto já existente na Petrobras e, para tal, algumas características que tornam essa integração mais sutil são extremamente desejáveis pois aumentam o aceite por parte dos engenheiros. A introdução de uma nova fase em uma metodologia de desenvolvimento de projetos já consolidada se apresenta como um desafio por lidar com hábitos já antigos, bem estabelecidos e transmitidos entre gerações de funcionários. Para uma boa integração e uma boa aceitação por parte dos desenvolvedores, as seguintes características foram buscadas:

- A Metodologia deve trabalhar com informações já existentes
- A Metodologia deve evitar o aumento do tempo de projeto
- A Metodologia deve possuir uma fácil usabilidade
- A Metodologia deve apresentar resultados facilmente entendidos e aproveitáveis

Levando o citado em consideração, o primeiro passo para a especificação da metodologia é definir em que momento, durante o projeto, ela deve ser utilizada e, em consequência, quais informações serão utilizadas.

Como mostrado anteriormente, a metodologia atual segue a uma ordem de desenvolvimento como demonstrado na figura 13. A proposta é introduzir a nova metodologia ao fim do desenvolvimento do programa de CLP, para utilização do CLP já programado como uma das entradas para o teste. Outra entrada são as informações da matriz causa e

efeito e também uma simulação da planta. A saída após a realização de testes é um texto em formato de Log que apresenta os testes realizados (sequência de ações tomadas) e os resultados - sucessos ou falhas do programa. A figura14 apresenta a introdução da metodologia dentro do processo de desenvolvimento de um projeto. Outra característica importante ao se propor esta metodologia é que ela pode ser implementada em paralelo com a metodologia já existente. Esse paralelismo permite uma aderência melhor ao permitir comparação entre ambas metodologias.

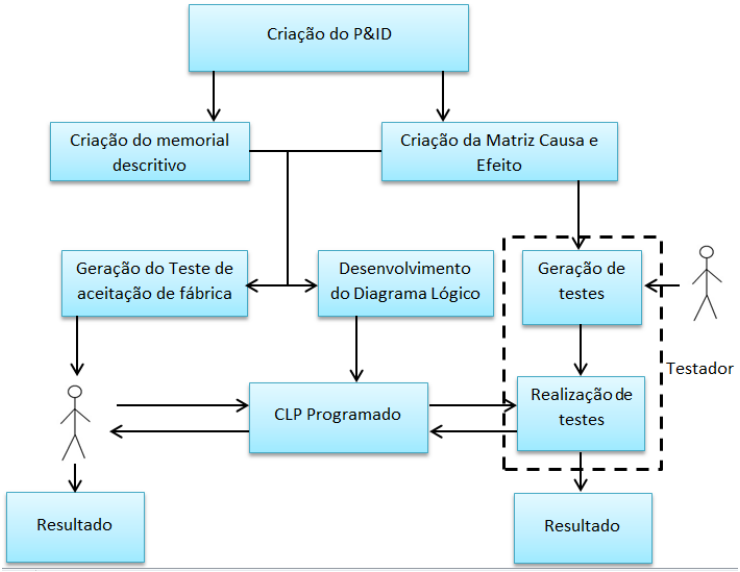


Figura 14: Inserção da metodologia no desenvolvimento do projeto.

Dentre os documentos descritos no capítulo 3, alguns possuem uma relação direta ou quase direta com o código Ladder gerado e dois devem ter sua importância ressaltada: a matriz causa e efeito e o diagrama lógico. Ambos são intimamente relacionados com o código desenvolvido para o CLP e possuem uma relação direta com o código gerado, isto é, a informação contida nesses documentos é traduzida diretamente para códigos Ladder sem maiores mudanças das lógicas presentes ou necessidade de grandes interpretações.

O diagrama lógico, assim como exposto anteriormente, é gerado a partir de documentos iniciais de descrição do sistema (memorial descritivo, matriz causa e efeito e fluxograma de processo). Tal aborda-

gem envolve a geração de uma lógica em um documento através de interações, sem teste ou verificações realizadas sobre essa mesma lógica e pode resultar em uma lógica com falhas. O resultado final da criação do diagrama lógico, pode, dessa maneira, possuir erros de lógica ou ainda estar incompleto. Não se pode então, utilizar o diagrama lógico como especificação para a geração de testes exatamente porque essa especificação utilizada na criação do programa Ladder pode conter erros e, mesmo uma verificação baseada em *model checking*, que utilize essa lógica como base, não garante o correto funcionamento do programa Ladder.

Então, para a realização dos testes, escolheu-se a matriz causa e efeito como ponto de partida por 2 motivos principais: esse documento está presente durante todo o projeto, acompanha todo o processo de desenvolvimento e apresenta, ainda, requisitos funcionais do projeto. Por esse documento exibir os dados nele contidos, a relação entre sinais de campo e sinais para atuadores no campo, de maneira constante e baseada em regras, ele possui certa formalidade. Ele não está sujeito a representações puramente textuais, as quais, normalmente, tem a problemática de necessitar de interpretação humana, estando sujeitas a ambiguidade, e podem apresentar diversas variações dependendo do engenheiro responsável por sua criação.

A partir das informações contidas na matriz são gerados testes para o programa e são também criados observadores baseados em redes de Petri para representar o comportamento esperado para o software de CLP. O processo de geração de testes resulta em uma sequência de entradas para o CLP que é utilizada então para a realização dos testes. Através de um protocolo industrial, o CLP tem seu programa executado com suas entradas forçadas, de acordo com aquilo estipulado na sequência de entradas, e uma planta simulada é utilizada como substituta para a planta real. Os dados de saída do CLP, são armazenados para posterior comparação com os observadores gerados anteriormente.

Os observadores criados anteriormente são então utilizados como base para a validação do teste. Eles contém o comportamento esperado para o sistema e são executados utilizando os valores armazenados durante o teste. A partir do estado final alcançado por esses observadores, é possível concluir se houve um erro ou não. Os resultados finais, ou seja, a presença ou não de erros, o estado final alcançado pelo observador e os sinais envolvidos são então apresentados para o usuário. A figura 15 apresenta a metodologia explicada anteriormente numa visão geral.

Figura 15: Visão geral da metodologia proposta

4.2 GERAÇÃO DE OBSERVADORES

Um dos passos da metodologia deste documento é a geração de redes de Petri a partir das informações contidas na matriz causa e efeito. Escolheu-se representar as informações através de redes de Petri por permitirem uma fácil modelagem de eventos paralelos e, por conseguir representar um grande número de diferentes estados utilizando um número reduzido de lugares. A utilização de uma abordagem formal bem estabelecida como as redes de Petri para a representação das informações é importante por ser bem aceita pela comunidade de pesquisa, por facilitar a implementação de ferramentas que utilizem essa abordagem e ainda por permitir que os modelos criados sofram operações de maneira a se obter novos modelos mais completos.

Dentre aquilo que se deseja observar do sistema programado, existem o caso de uma saída de CLP que deveria ser ativada mas não foi e o caso de uma saída de CLP que foi ativada sem a necessidade de ser. Esses casos são respectivamente os falsos negativos e os falsos positivos. Os observadores aqui tratados são construídos para tratar

dos casos de falsos negativos, ou seja, eles são utilizados para observar se o sistema deixou de ativar alguma saída quando necessário.

Seguindo a visão geral apresentada na seção 4.1, a partir do documento inicial, seguem duas etapas que utilizam as informações desse documento. A criação de redes de Petri que representam as informações contidas nesse documento e a definição dos testes que serão gerados.

A matriz causa e efeito apresenta as informações de sensores de campo em linha e de acionamentos em coluna. O objetivo das redes de Petri geradas é apresentar um sistema reativo que possa ser utilizado mais tarde para comparação dos resultados obtidos em teste. Para a definição dessas redes de Petri, foram analisadas os diferentes tipos de sinais vindos do campo, sua relação com os acionamentos e suas características que deveriam estar presente no modelo criado. As entradas foram dividas em grupos que possuem características semelhantes, os grupos são:

- Entradas boleanas simples
- Grupo de entradas boleanas
- Votação de sinais de campo
- Entradas boleanas com múltiplos efeitos temporizados

Para cada um dos tipos de entradas foram definidas redes que apresentam algumas características básicas:

- Um sinal vindo do campo, se não tratado, deve caracterizar mais tarde uma falha no software
- Se todos os passos necessários para o correto tratamento do sinal forem realizados, a execução com sucesso deve ser identificada

A seguir as redes definidas serão apresentadas. Redes com maiores detalhes desenvolvidas a partir de uma matriz causa e efeito real juntamente com uma planta exemplo selecionada para melhor compreensão serão abordadas no capítulo 5.

Dado que o sistema(CLP programado) a ser testado é um sistema do tipo caixa-preta, ou seja, somente se tem acesso às entradas e saídas do sistema, escolheu-se trabalhar com observadores para o diagnóstico de possíveis erros. A opção por se utilizar observadores para o diagnóstico de erros no programa se dá pelo fato de que o sistema (CLP programado) é do tipo caixa-preta, e, os observadores, como explicado, não devem ser intrusivos, ou seja, não devem alterar o comportamento

do sistema sendo observado. Essa característica mandatória do observador está em concordância com o necessário para o diagnóstico em um sistema caixa-preta, uma vez que o programa do CLP não pode ser alterado.

Dentre as informações contidas na matriz causa e efeito, se deseja poder observar se a relação estipulada entre causa e efeito foi cumprida. Cada causa possui um ou mais efeitos correspondentes e são esses efeitos que se deseja observar. As redes de Petri aqui geradas são então observadores do sistema que representam as propriedades a serem observadas no programa de CLP e, assim como um observador, não introduzem novas informações no sistema sendo utilizadas somente para se observar a presença de falha ou não. Elas são utilizadas para determinar se os valores obtidos pelo teste são aqueles que deveriam ser obtidos por um programa que segue as especificações encontradas na matriz causa e efeito.

No decorrer desta seção, serão apresentados os observadores desenvolvidos para o caso de estudo.

4.2.1 Observadores básicos

4.2.1.1 Entradas boleanas simples

O caso mais simples existente no projeto é a relação direta entre uma entrada e uma saída na matriz causa e efeito. O modelo que representa esse caso é apresentado na figura 16 onde a entrada é "Desligamento Manual" e a saída é "Vapor de abafamento". Os lugares "Desligamento manual"e "Abertura do vapor de abafamento"indicam o recebimento de um sinal de campo para desligamento manual e o envio de um sinal para a abertura de uma válvula de abafamento respectivamente.

Para esse modelo, o comportamento esperado pode ser analisado da seguinte maneira: quando o comando para desligamento manual é enviado ao CLP, o tempo da transição t0 passa a decorrer e, se até o tempo limite t1 o lugar "Vapor de abafamento"não houver recebido uma ficha, indicando que a válvula de abafamento não foi aberta, a transição *time out* dispara sinalizando que o sistema atingiu um estado indesejável. Nesse caso, isso significa que o sistema não executou a ação de abrir o vapor de abafamento dentro de um tempo limite t1. Caso a ficha seja adicionada dentro do tempo limite, a rede alcança um estado que representa então o funcionamento desejado para essa

relação específica.

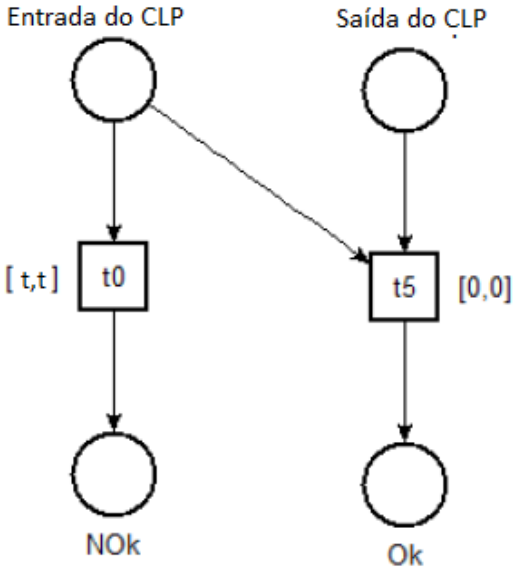


Figura 16: Observador em redes de Petri para uma entrada boleana simples

4.2.1.2 Grupos de entradas boleanas

Um tipo de entrada presente no documento é a existência de um número mínimo de sinais dentro de um determinado conjunto. Como exemplo temos "três ou mais queimadores sem chama", ou seja, para todos os queimadores da planta, se no mínimo três deles ficarem sem chama, o código referente a essa linha deve ser executado no programa de CLP. Para representar essa relação, a rede de Petri representada na figura 17 foi desenvolvida. Os sinais utilizados serão, a título de exemplificação somente, "Três ou mais queimadores sem chama"e "Vapor de Abafamento"como no exemplo anterior.

Ela apresenta um comportamento parecido com o já descrito para a entrada boleana simples. A diferença se dá que aqui, o arco para o disparo da transição tem peso 3, indicando a necessidade da existência de três fichas (três queimadores sem chama) para a execução

da rede.

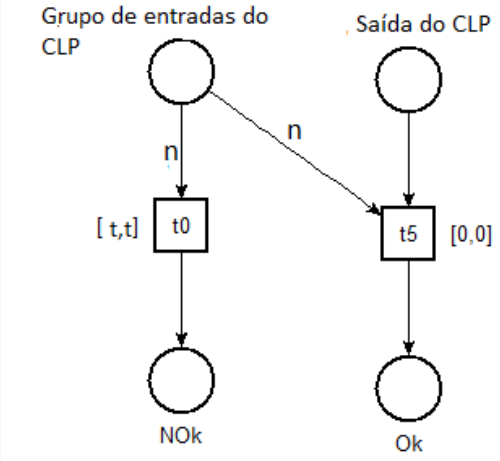


Figura 17: Modelo para um grupo de entradas boleanas

4.2.1.3 Votação de sinais de campo

Outro caso presente na matriz causa e efeito é a votação um de dois(1oo2) e a votação dois de três(2oo3). Essas votações são utilizadas quando um sinal lógico possui mais de um sinal físico no campo, isto é, quando dois transdutores diferentes oferecem ao sistema de controle um valor redundante e os dois sinais, um de cada transdutor, que chegam ao CLP devem ser considerados na escolha da ação a ser tomada. Na votação 1oo2, dois sinais chegam do campo e, se um deles estiver ativo, então o resultado da votação deve ser considerado ativo enquanto na votação 2oo3, é necessário que no mínimo 2 de 3 sinais estejam em concordância(e ativos) para que o resultado da votação seja considerado. As figuras 18 e 19 mostram os casos para votação 1oo2 e 2oo3 respectivamente.

Nesses modelos os lugares "sinal n"com n sendo 1, 2 ou 3 são os lugares que representam os sinais vindo de campo, o lugar "realizar votação"assegura que somente uma ficha seja repassada para o lugar "resultado votação".

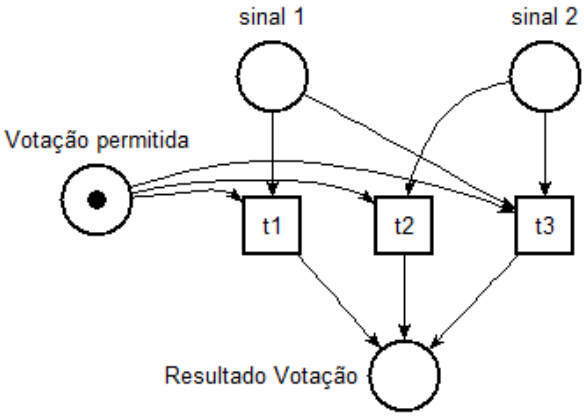


Figura 18: Modelo para um caso de votação 1oo2

4.2.1.4 Entradas boleanas com múltiplos efeitos temporizados

Outro caso comum é a existência de sequenciamentos dentro do sistema. Para exemplificar um sequenciamento, uma nota presente na matriz causa e efeito é apresentada na figura 20. Neste caso, a nota apresenta três passos que devem ser realizados, em ordem: é necessário fechar a primeira válvula de bloqueio, em seguida, fechar a válvula de ventilação e, após 5 segundos, fechar a segunda válvula de bloqueio. A rede na figura 20 possui 4 lugares que representam os sinais de entrada e saída sendo que os sinais "fechar primeira válvula", "abrir válvula de ventilação"e "fechar segunda válvula"são sinais de saída do CLP e "Pressão muito alta no piloto"é um sinal de entrada do CLP. Nesse modelo o caminho composto pelas transições t1, t4, t6 e t10 representam o caminho esperado para o funcionamento correto. As transições t0, t2, t5, t7, t9 e t12 apresentam possíveis problemas na execução do modelo.

t1, t4, t6 e t10 são respectivamente as transições que representam o fechamento da primeira válvula, a abertura da válvula de ventilação, a espera de 5 segundos antes do fechamento da segunda válvula e o fechamento da segunda válvula. Já t0, t2, t5, t7, t9 e t12 são, em ordem, um time-out para o não fechamento da primeira válvula, a abertura da válvula de ventilação antes do fechamento da primeira válvula de bloqueio, o *time out* da abertura da válvula de bloqueio, o fechamento da segunda válvula antes da abertura da válvula de ventilação, o fecha-

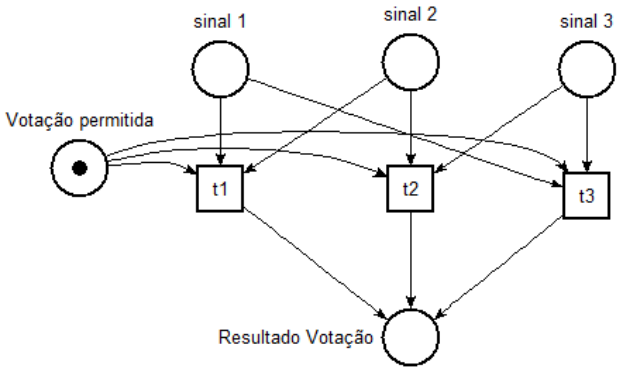


Figura 19: Modelo para um caso de votação 2oo3

mento da segunda válvula de bloqueio antes de passados 5 segundos e o *time out* para o fechamento da segunda válvula de bloqueio.

4.2.2 Composição de observadores

Um mesmo sinal de sensor de campo pode necessitar de diferentes tratamentos de acordo com sua relação com acionamentos. Assim, uma entrada boleana simples pode ainda ser também uma votação 1oo2 ou 2oo3. Caso o sinal seja deste tipo, é necessário uma fusão entre as redes correspondentes. Como visto na seção 2.1.2, a fusão de duas redes de Petri ou, nesse caso, observadores é feita pela união ou dos estados ou das transições equivalentes entre esses modelos. Este método permite a manutenção das características de modelos individuais enquanto permite a criação de um modelo maior e mais completo. Para exemplificar a fusão de observadores, a figura 21 apresenta a fusão entre a votação 1oo2 e o caso para uma entrada boleana simples apresentado anteriormente.

4.3 GERAÇÃO DE TESTES

Assim como apresentado na figura 15, paralelamente à geração dos observadores, existe a geração dos testes a serem realizados. Os testes devem ser decididos a partir das informações de sinais de campo e acionamentos presentes na matriz causa e efeito.

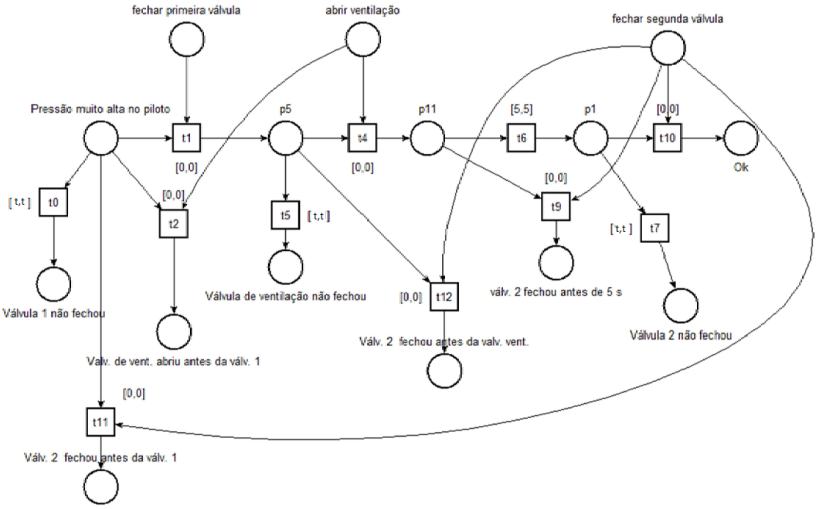


Figura 20: Modelo em redes de Petri para um caso onde uma nota determina uma ordem de execução

A quantidade total possível de testes a serem realizados cresce de acordo com o número de entradas existentes no sistema a ser testado. Como as entradas tratadas são binárias podendo assumir 2 valores diferentes, o número total de combinações de entradas existentes em um sistema com n entradas é de 2^n. A fim de se evitar esse crescimento exponencial, trabalha-se com particionamento de equivalência(BEIZER, 1990) para reduzir o número de testes realizados e não limitar a aplicação da metodologia à pequenos projetos. Entre todas as combinações possíveis que afetam um observador, opta-se por utilizar as entradas que afetam somente o observador em questão. Desta maneira, algumas regras que norteiam a escolha das entradas para os testes foram definidas. Para tal, as entradas foram divididas numa maneira análoga aos diferentes tipos de observadores. Os diferentes testes são apresentados em seguida.

O primeiro tipo básico de teste a ser realizado é o seguinte:

- Para cada Linha da matriz, forçar o sinal de campo correspondente àquela linha

- Comparar o desenvolvimento das saídas do CLP de acordo com as redes de Petri criadas

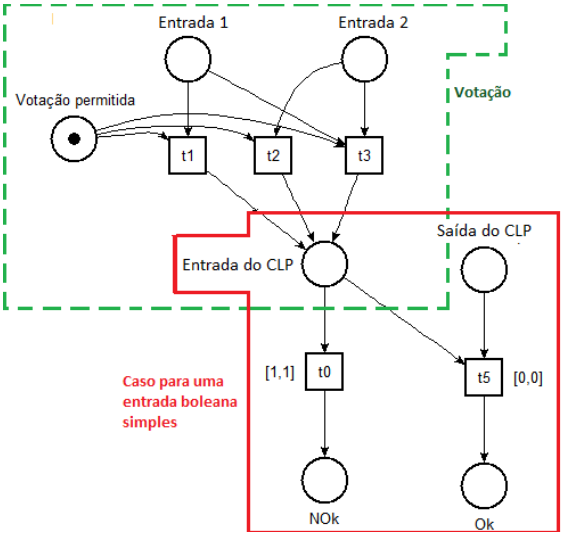


Figura 21: Modelo para um caso de votação 1oo2, contorno pontilhado, fundido com uma entrada boleana simples, contorno contínuo.

Esse primeiro teste simples permite afirmar, ao menos, que o programa executa os trechos correspondentes às linhas da matriz causa e efeito corretamente.

Um segundo tipo de teste gerado é aquele correspondente à votação, isto é, quando o sinal possui uma votação 1oo2 ou 2oo3. Para estes casos, são gerados diferentes casos de teste. Para a votação 1oo2, existem 3 diferentes testes sendo eles a ativação de somente um dos sinais de campo, a ativação do outro sinal de campo e a ativação simultânea de ambos os sinais de campo. Já para a votação 2oo3, existem um total máximo de quatro testes possíveis que consistem na combinação dos três sinais de campo existentes. Dados os sinais de campo A, B e C para o caso de votação 2oo3, os testes possíveis consistem na ativação das entradas: AB, AC, BC e ABC

O terceiro tipo de teste gerado é referente caso descrito na seção 4.2.1.2, quando número mínimo de um grupo de sinais de campo deve estar ativo para que se execute uma ação correspondente. Para este caso, assim como para a votação 2oo3, existe uma combinação de entradas possíveis para se realizar o teste. Dado m como o número mínimo de entradas necessárias para ativar a execução e t como o número total de sinais provenientes do campo, o número de testes pode ser dado pela

combinação de t m a m, ou ainda $C_m^t = t!/(m!(t-m)!)$.

4.3.1 Execução dos testes

Os testes gerados são armazenados na forma de uma lista que indica, para cada caso de teste, qual entrada deve ser ativada. Durante a execução dos testes, o seguinte é realizado:

- Acionamento/desativação de determinada entrada especificada na lista de testes
- Inicio do armazenamento das variáveis e do tempo, relativo ao início do teste, em que elas foram recebidas
- Aguardo do final do caso de teste
- Início do próximo caso de teste na lista

4.4 EXECUÇÃO DOS OBSERVADORES A PARTIR DOS RESULTADOS DOS TESTES

Ao final da realização dos testes, existem diversos dados armazenados(correspondentes às entradas e saídas do CLP). Para cada caso de teste realizado, os observadores equivalentes são selecionadas e, utilizando um jogador de redes de Petri e as informações armazenadas do teste(inclusive tempo), compara-se os dados armazenados com os modelos em redes de Petri. Um jogador de redes de Petri é responsável pela execução da rede de Petri e sua evolução dada a marcação inicial. No caso do jogador para esta metodologia, além da marcação inicial, ainda é considerado a inserção de fichas de acordo com a presença dos sinais de entrada e saída. O jogador funciona executando o seguinte ciclo enquanto restarem dados armazenados para cada caso de teste:

- Atualização do tempo dos arcos
- Disparo dos arcos que podem ser disparados
- Atualização dos valores das entradas e saídas dos CLP (Inserção de fichas nos lugares correspondentes)

Ao final, com os dados armazenados, o observador é analisado para verificar se um estado desejável ou indesejável foi alcançado. Para

exemplificar a abordagem, considere a figura 22. Ela apresenta duas execuções de testes diferentes, sendo que uma reconhece a existência de um erro enquanto outra é um teste executado com sucesso. Estes dados são referentes à uma linha cujo modelo em redes de Petri correspondente ao já apresentado na figura 16.

Tempo	Entrada do CLP	Saída do CLP
0	0	0
0.25	1	0
0.5	1	1
0.75	1	1
1	1	1
1.25	1	1

Tempo	Entrada do CLP	Saída do CLP
0	0	0
0.25	1	0
0.5	1	0
0.75	1	0
1	1	0
1.25	1	1

Figura 22: Dados armazenados após duas execuções de testes diferentes

Ao seguir o ciclo estipulado anteriormente, tem-se que, para a primeira execução (tabela da esquerda), o lugar "Ok"acaba com uma ficha enquanto, para o segundo caso (tabela da direita), o lugar "NOk"acaba com uma ficha. A partir dessa análise, pode-se afirmar que o teste referente à tabela esquerda finalizou com sucesso enquanto, em contrapartida, o teste relacionado com a tabela direita, indicou falha do programa de CLP.

4.5 COMENTÁRIOS FINAIS

Neste capítulo foram foram abordados os aspectos da metodologia proposta para realização de testes em projetos utilizando os documentos já existentes na metodologia de desenvolvimento de projeto da Petrobras. Foram detalhadas as características componentes dessa abordagem bem como sua aplicação para a geração de testes. Os observadores abordados aqui são modelados com o objetivo de verificar a presença de falsos negativos. A identificação dos falsos positivos também é importante uma vez que um CLP que ativa todas as suas saídas para qualquer sinal de entrada passaria no teste de falsos negativos. Para se adicionar a detecção de falsos positivos, novos observadores deveriam ser modelados com essa finalidade. Eles seriam comportados pela meto-

dologia proposta, sendo necessário somente sua modelagem e adição no processo de testes. Para tal modelagem, seguindo os mesmos princípios dos observadores apresentados, seria necessário definir a saída não desejada e relacioná-la com um lugar não desejado, um "NOk", numa lógica contrária àquela utilizada na definição dos observadores apresentados.

5 DESENVOLVIMENTO DE UMA FERRAMENTA PARA SUPORTAR A METODOLOGIA PROPOSTA

Neste capítulo, será apresentada uma ferramenta desenvolvida para a realização de testes em um estudo de caso baseado no forno já abordado no capítulo 3. Serão abordados os módulos funcionais da ferramenta, sua funcionalidade e também as limitações desta implementação em questão.

5.1 DESENVOLVIMENTO DE UMA FERRAMENTA DE TESTES E VALIDAÇÃO DA METODOLOGIA

Para a criação da ferramenta, foi utilizada uma abordagem top-down. A ferramenta foi definida partindo de visões gerais e então, foi especializada e especificada. Numa primeira interação o interfaceamento mais alto nível da ferramenta foi definido, ou seja, a entrada dos dados contidos na matriz causa e efeito e a saída em formato de um Log para posterior consulta. Como próximo passo, foram decididos os aspectos básicos da ferramenta e, assim, o fluxo de informações dentro da ferramenta foi especificado. Como entrada, tem-se as informações da matriz causa e efeito e, a partir dessas informações, dois principais "itens"devem ser gerados:

- Sequências de *inputs* utilizados para execução de sequências no CLP
- Modelos a serem utilizados para comparação da execução real do programa de CLP com a execução esperada

A sequência gerada é então utilizada na troca de informação entre o CLP e a planta simulada. Essa troca produz uma série de valores(variáveis da planta simulada e valores de saída do CLP), os quais são armazenados para uso posterior.

Após a execução das sequências criadas, os valores armazenados são utilizados como base para comparação contra o comportamento esperado. Esse comportamento esperado é obtido a partir dos modelos gerados no passo anterior. Da comparação, um log é gerado e finalmente apresentado ao usuário. Esta é a abordagem já detalhada no capítulo 4.

A estrutura que compõe a ferramenta foi subdividida em diferentes módulos responsáveis, cada um, por parte das tarefas realizadas

durante a geração e realização de testes. Os módulos são os seguintes:

- Interface: responsável por permitir a entrada de dados da matriz causa e efeito e também processamento desses dados para utilização pelos módulos seguintes
- Gerador de observadores: realiza a transformação dos dados em observadores baseados em redes de Petri para serem utilizados por outros módulos
- Gerador de comandos: gera uma lista de testes a serem realizados baseado nas informações contidas na interface
- Gravador de resultados do CLP: grava os dados do CLP e da planta simulada durante o teste para futura comparação
- Comparador dos resultados com os observadores e gerador de Log: utiliza dados gravados durante os testes para comparar os comportamentos dos observadores e do sistema a fim de definir se o comportamento foi o correto
- Comunicação OPC: realiza a comunicação OPC entre planta simulada e ferramenta de testes

Os módulos especificados interagem entre si trocando os dados necessários. Isso gera um fluxo dos dados e de transformação dos dados dentro da ferramenta desde a informação crua da matriz causa e efeito até o Log final gerado. A figura 23 apresenta os módulos e também sua interação.

5.2 ENTRADA DE DADOS

O objetivo da interface é permitir que o usuário insira as informações contidas matriz causa e efeito. Como visto no capítulo 3, a matriz possui algumas informações textuais além da relação direta entre sinal de campo e sinal de entrada. O usuário também deve ser capaz de inserir essas informações de maneira que elas possam ser utilizadas posteriormente para a criação de modelos.

Para a representação dos dados, foi escolhido a transformação de toda a informação visual contida na matriz causa e efeito para uma informação textual que possa ser facilmente analisada, transformada e utilizada. Este arquivo de texto contém a informação da matriz causa e efeito de uma maneira ordenada e é dividido em duas partes principais:

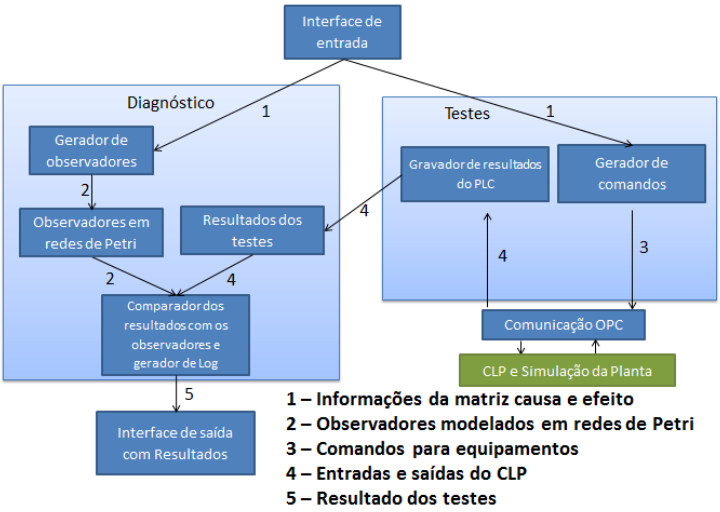


Figura 23: Módulos da ferramenta e fluxo de informação entre módulos

uma série de sinais que correspondem às colunas e, ao fim destes, uma séries de sinais que correspondem às linhas.

5.3 GERAÇÃO DE OBSERVADORES

Um dos aspectos chave da ferramenta criada é a geração de observadores baseados em redes de Petri a partir das informações contidas na matriz causa e efeito. A ferramenta deve ser capaz de criar várias instâncias de observadores, cada um adaptado para um sinal de campo(causa) e atuador(efeito) diferentes. Para atingir tal objetivo, foram definidas estruturas genéricas para as redes de Petri utilizadas para os observadores. Essas estruturas são mais tarde especificadas para adequar o observador à um conjunto de sinais específicos. A figura 24 sumariza os passos para a obtenção de observadores.

O exemplo de observador dado para a figura 25 é parte do banco de observadores presente na ferramenta. Os locais dessa rede possuem os nomes genéricos "fromfieldsignal1" e "tofieldsignal1" propositalmente. Eles representam os sinais de entrada do campo e de saída para o campo. Ao instanciar um observador, a ferramenta o torna único substituindo esses nomes pelos sinais existentes na matriz causa e efeito. Assim, a instanciação de um observador na ferramenta funciona da se-

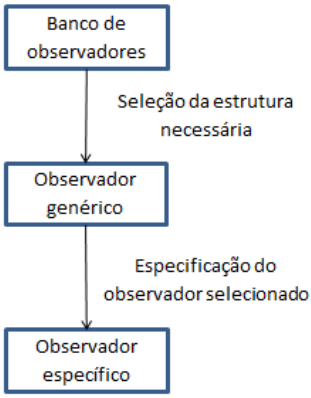


Figura 24: Passos para se obter os observadores a partir do banco de observadores

guinte maneira: é escolhida uma definição já existente, os nomes dos lugares são substituídos de maneira a representar corretamente uma relação na matriz causa e efeito e, em seguida, pode ser armazenada para posterior uso. Para o caso descrito, "fromfieldsignal1"deve ser substituído pelo nome da variável contida na linha da matriz causa e efeito enquanto "tofieldsignal1"deve ser substituído pelo nome da variável na coluna da matriz causa e efeito. Levando-se em consideração a primeira linha da matriz causa e efeito da planta exemplo e a sétima coluna da mesma, temos como causa o sinal "PSHH-014A/B/C"e como efeito, o equipamento "XY-037". Para a obtenção da primeira rede necessária para o observador correspondente, o nome "fromfieldsignal1"é substituido por "PSHH-014A/B/C"enquanto o nome "tofieldsignal1"é substituido por "XY-037". O resultado é o visto na figura 26.

Um outro caso importante na criação dos observadores é quando se faz necessário a fusão de duas redes diferentes para se obter o observador desejado. Para tal fim, assim como descrito anteriormente, os observadores já existentes nos bancos de observadores são especificados de acordo com os sinais correspondentes na matriz causa e efeito e, então, suas informações são fusionadas a fim de se obter o observador completo. Para o mesmo conjunto de sinais "PSHH-014A/B/C"e "XY-037"observa-se que é necessário ainda a especificação de uma votação 2oo3, assim como está definido na matriz causa e efeito na coluna "notas". A votação 2oo3 também possui um observador já definido com nomes genéricos e a substituição destes nomes pelos conjuntos de sinais

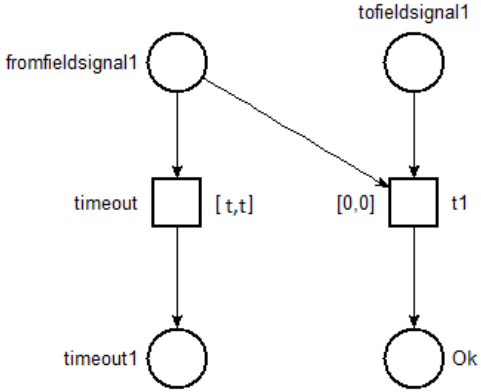


Figura 25: Exemplo de observador que compõe o banco de observadores

"PSHH-014A/B/C"e "XY-037"resulta na rede vista na figura 27.

O último passo para a obtenção do observador completo que relaciona os sinais "PSHH-014A/B/C"e "XY-037"é então a fusão das duas redes apresentadas nas figuras 26 e 27. Como resultado desta fusão obtém-se a rede de Petri apresentada na figura 28. Esta rede é o observador completo para os sinais "PSHH-014A/B/C"e "XY-037".

5.4 DEFINIÇÃO DOS CASOS DE TESTE E GERAÇÃO DE COMANDOS

A partir das entradas já lidas, a ferramenta pode, agora, definir quais serão as sequências impostas às variáveis de acordo com a metodologia explicada no capítulo 4. Para tal, são observados tanto os sinais de campo(sensores) quanto possíveis detalhes existentes que afetem aquele sinal. No caso da primeira linha da matriz, as entradas "PSHH-014A", "PSHH-014B"e "PSHH-014C"possuem uma relação de votação 1oo3. Isso significa, assim como visto no capítulo 4 que serão testadas todas as combinações entre essas três entradas. Esse módulo da ferramenta cria uma lista com o nome das variáveis de campo e os valores que elas devem assumir de maneira a executar uma das linhas da matriz causa e efeito.

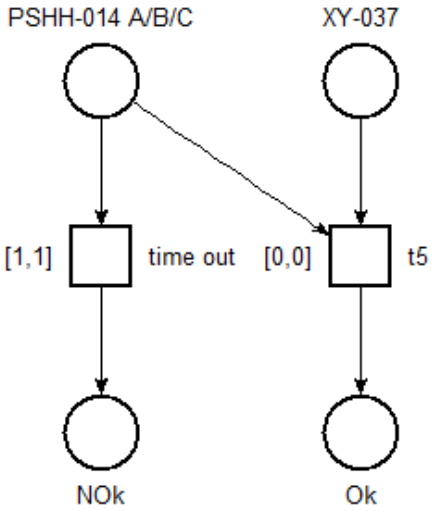


Figura 26: Primeiro passo no desenvolvimento do observador

Um exemplo do teste gerado para as variáveis "PSHH-014A", "PSHH-014B"e "PSHH-014C"(variáveis da primeira linha da matriz causa e efeito) é o seguinte: "PSHH-014A"→ 0, "PSHH-014C"→ 0. Isso significa que em um dos testes gerados, as entradas "PSHH-014A"e "PSHH-014C"seriam forçadas para 0(0 aqui é considerado o valor que ativa a entrada). Da mesma maneira, para esta linha, ainda seriam gerados mais três testes, cada um considerando uma combinação diferente para as entradas "PSHH-014A", "PSHH-014B"e "PSHH-014C", sempre forçando seu valor para 0 em cada teste.

5.5 INTERAÇÃO DA FERRAMENTA COM A PLANTA SIMULADA

Para que os testes possam ser realizados é necessário uma simulação da planta a ser automatizada e também de comunicação com o CLP já programado. De maneira a utilizar um protocolo já conhecido para os engenheiros, foi escolhido o OPC(OLE for Process Control) como ponte entre o CLP e planta simulada. OPC é um padrão de comunicação industrial entre equipamentos de diferentes fabricantes criado em 1996 e oferece diferentes versões com objetivos específicos.

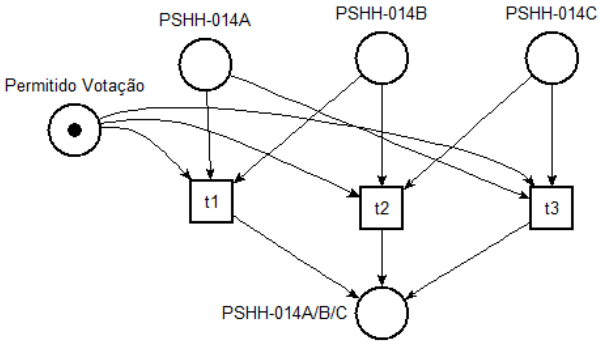


Figura 27: Votação para o sinal "PSHH-014A/B/C"

Neste trabalho foi utilizado o OPC DA(*data access*), que oferece uma interface simplificada para acesso de variáveis de processo através de identificadores chamados de *tags*. É um protocolo bem estabelecido no ambiente industrial, muito utilizado em sistemas supervisórios e, além da ampla aceitação por parte da indústria, é amplamente suportado pela comunidade de desenvolvimento de software através de diferentes bibliotecas de acesso.

Após os comandos definidos, a ferramenta pode iniciar a execução do teste, forçando cada variável assim como explicado anteriormente. Nesse momento, existe comunicação entre a planta e a ferramenta e se faz necessário especificar como a ferramenta é capaz de alterar o valor de uma variável simulada na planta.

Foi definido um princípio para a criação de uma planta simulada tal que permite à ferramenta forçar corretamente os valores de campo quando necessário. Deve ser adicionada à simulação, assim como explicitado na figura 29, uma lógica que dê prioridade ao valor enviado pela ferramenta(neste caso, valor 0). Aqui pode-se ver que a planta executa a simulação normalmente, mas o valor final da variável de campo é decidido pelo comando enviado da ferramenta quando essa deseja forçar o valor a 0.

5.5.1 Limitações temporais da ferramenta desenvolvida

Um ciclo de CLP demora na ordem de milissegundos dependendo do equipamento em questão enquanto um servidor OPC tem tempo de

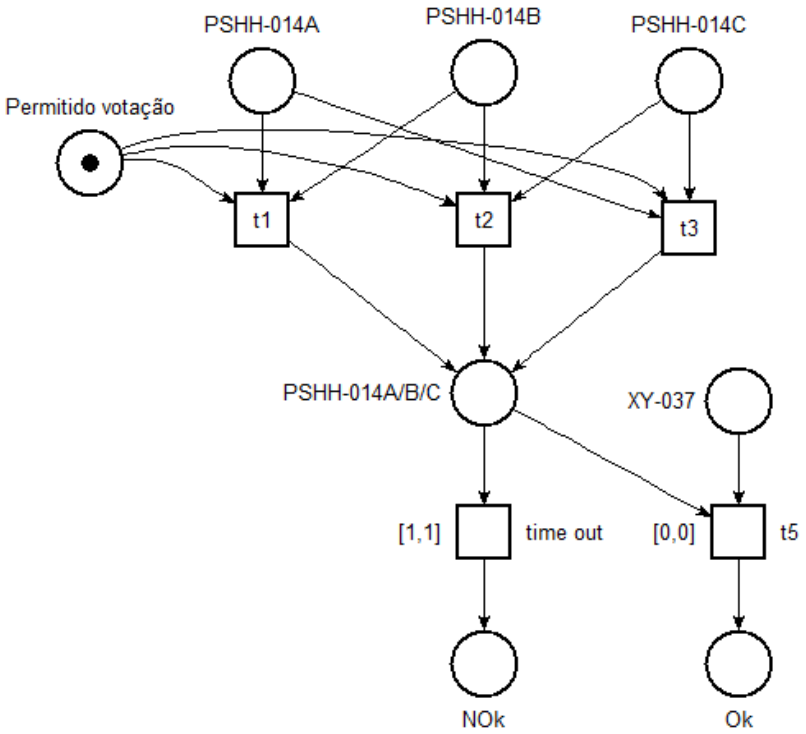


Figura 28: Observador criado para a relação "PSHH-014A/B/C"e "XY-037"

atualização entre 0,2 segundos e alguns segundos dependendo do servidor e configurações utilizados. Essa diferença deve ser considerada ao se estipular o tempo limite para ser considerado um *time out* na execução de um teste. Um tempo para *time out* não deve ser menor que o tempo mínimo para aquisição dos dados, caso contrário, pode-se encontrar uma falha onde ela, na realidade, não existe. Assim, no pior cenário, o tempo mínimo para *time out* a partir do tempos de aquisição de dados e processamento do CLP pode ser calculado da seguinte maneira: $timeout >= tempoparaescritanoservidorOCP + tempodeprocessamentodoCLP + tempoparaleituradoservidorOPC$. Este tempo deve ser considerado na realização dos testes para não gerar um falso negativo como resultado.

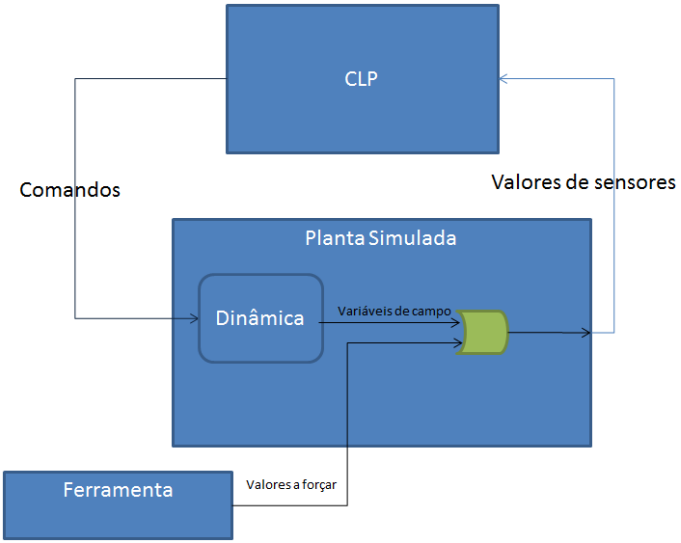


Figura 29: Paradigma utilizado para forçar os valores das variáveis de campo na planta simulada

5.6 UTILIZAÇÃO DOS OBSERVADORES PARA VALIDAÇÃO DOS TESTES

Enquanto os testes são realizados, uma tabela de valores lidos é montada para cada teste. Esta tabela deve conter as variáveis utilizadas no teste em questão assim como a informação do tempo decorrido entre as leituras. Para isso, antes do início do teste, são verificadas quais variáveis são pertinentes aos teste em questão, ou seja, a causa e o efeito de determinado teste. Essas variáveis são armazenadas e, a cada ciclo, tem seus valores lidos e armazenados. Além disso, em cada ciclo, é obtido, a partir do sistema operacional, o tempo decorrido desde a última leitura.

Após a realização de todos os testes, resta a etapa da comparação. Nesta etapa, os observadores gerados anteriormente e os valores gravados durante a execução dos testes serão utilizados para determinar se o programa de CLP se comportou como esperado. Neste momento, existem diversos observadores gerados (um para cada relação da matriz causa e efeito) e diversas tabelas com valores gravados durante a execução do teste. Para que a comparação seja executada, os

seguintes passos são seguidos:

- Para cada observador, relacionar a tabela de valores que trata as mesmas variáveis
- Executar o observador levando em consideração os valores contidos na tabela relacionada
- Analisar o estado alcançado, identificando uma execução com sucesso ou com falha

O segundo passo, a execução do observador, é o verdadeiro responsável pela validação do teste. Essa execução pode ser subdividido da seguinte maneira:

- Ler valores de uma linha das variáveis da tabela
- Alterar valores dos lugares correspondentes dos observadores (lugares referentes aos valores de campo e equipamentos)
- Atualizar tempo das transições habilitadas
- Disparar transições que possam ser disparadas
- Passar para a próxima linha da tabela e repetir

Com essa execução, pode-se observar então se os testes resultaram na descoberta de algum erro ou se o programa executou como esperado.

5.7 COMENTÁRIOS FINAIS

Neste capítulo, o desenvolvimento de uma ferramenta utilizada na validação da metodologia proposta no capítulo 4 foi apresentada. Aqui foi explicado em detalhes como a informação é tratada dentro da ferramenta e como essa informação é utilizada dentro dos diversos módulos nela contidos além da execução da tarefa específica à cada módulo.

6 AVALIAÇÃO DA METODOLOGIA PROPOSTA

Para a aplicação da metodologia proposta foi utilizada uma planta da Petrobras na realização de testes. A planta é a mesma de um forno já descrito no capítulo 3. Neste capítulo, serão apresentados os tópicos pertinentes à validação da metodologia, sua aplicação, erros introduzidos ao código, testes realizados e resultados encontrados. Serão apresentados os modelos desenvolvidos para este estudo de caso proposto e os resultados obtidos ao se aplicar a metodologia nesta planta exemplo.

6.1 CÓDIGO DE AUTOMAÇÃO

Para a validação da ferramenta, foi criado um código Ladder com o objetivo de automatizar a planta em questão. O código foi criado com respeito à matriz causa e efeito encontrada no anexo B e com a finalidade de seguir os requisitos impostos pela matriz. Para exemplificação, a figura 30 apresenta uma pequena parte do programa Ladder, sua lógica completa pode ser vista no anexo A

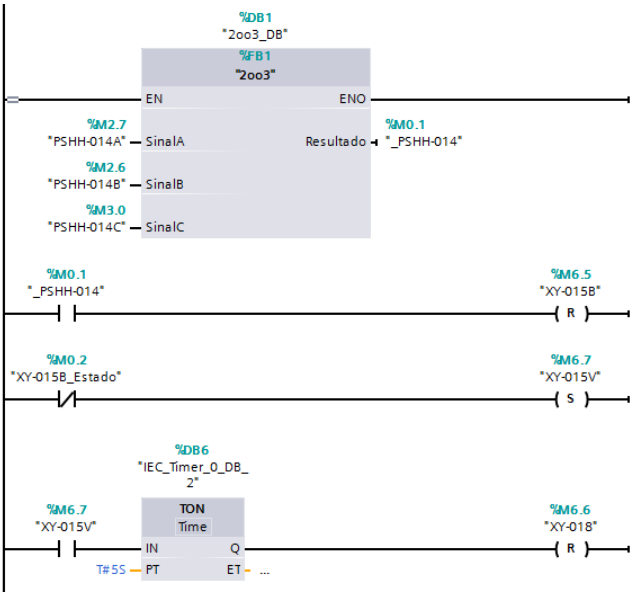


Figura 30: Exemplo de programa Ladder do caso de estudo

Para a criação do programa, se utilizou uma abordagem subdividindo a matriz causa e efeito em linhas. Cada linha da matriz gerou um trecho de código, chamado de *"net"* no ambiente de programação específico. Essa subdivisão em trechos distintos foi escolhida por facilitar uma possível correção de código caso fosse encontrado algum erro.

6.2 SIMULAÇÃO DA PLANTA

Para a simulação da planta, foi utilizado o ambiente simulink do software matlab. Essa escolha foi feita por facilitar o desenvolvimento da simulação da planta e possibilitar o desenvolvimento da planta no nível de abstração necessário, no entanto, outro software de simulação poderia ser utilizado dado que gravasse as informações de variáveis da planta simulada em servidor OPC.

Visto que o programa de CLP executa uma das linhas da matriz causa e efeito uma vez que um sinal correspondente é ativado e que, para a realização dos testes, é necessário a verificação dos sinais de sensores que chegam do campo e dos sinais enviados à atuadores, a planta simulada possui apenas algumas dinâmicas contínuas, as quais são importantes para a correta execução do programa. A figura 31 apresenta uma parte do ambiente simulink com a simulação da planta.

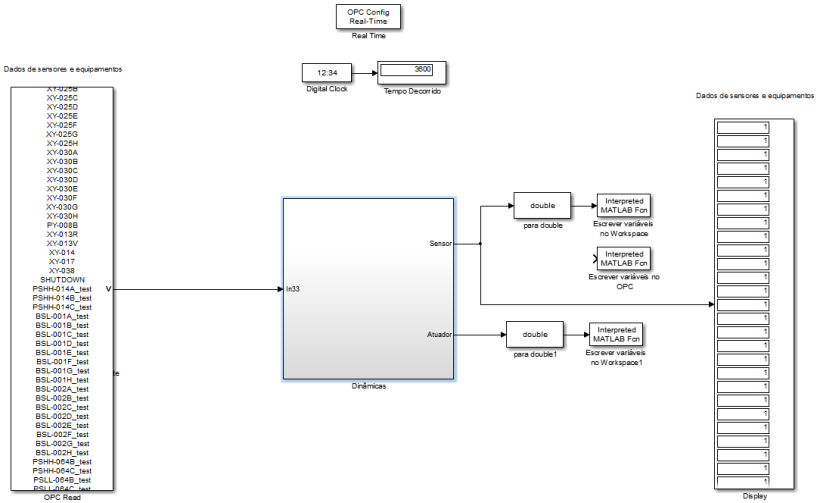


Figura 31: Exemplo de planta simulada no ambiente simulink

6.3 EXPERIMENTOS REALIZADOS E RESULTADOS OBTIDOS

Para uma melhor visualização dos testes sendo realizados, foi desenvolvida uma representação gráfica através da ferramenta WinCC. O objetivo deste sinóptico desenvolvido é apenas a visualização, logo, não foram criadas entradas de dados ou comandos, apenas elementos de visualização foram utilizados. A figura 32 apresenta o sinóptico criado.

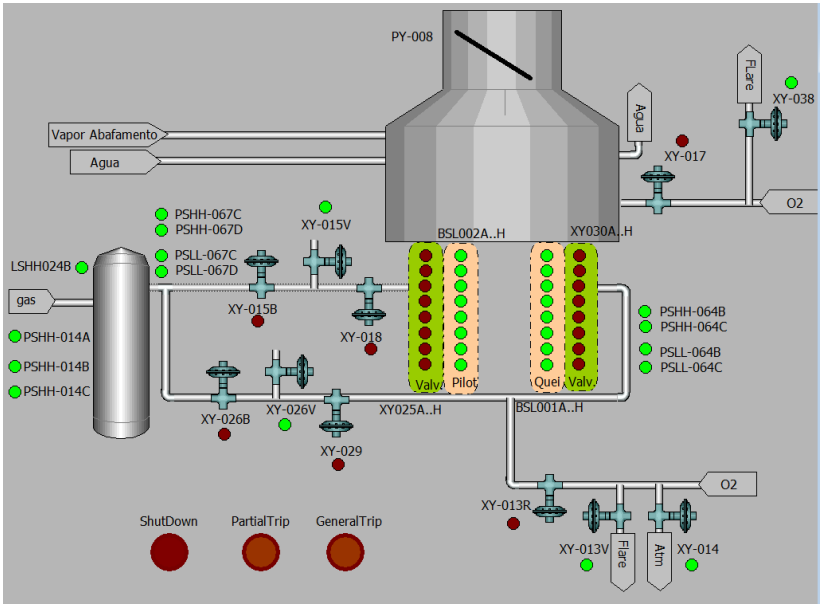


Figura 32: Supervisório criado para visualização dos testes

Para a validação da ferramenta, erros foram introduzidos no programa Ladder, e, em seguida, se observou se a ferramenta era capaz ou não de apontar a existência do erro. Erros foram introduzidos sistematicamente em cada seção do código a fim de se testar a capacidade de reconhecimento de diferentes erros. Para a demonstração do funcionamento da ferramenta, serão apresentados os erros inseridos no programa e em seguida o resultado gerado pela ferramenta e gravado no log.

- Primeiro caso de teste: neste primeiro caso, o algoritmo responsável por definir o correto funcionamento da votação 2oo3 no bloco de função visto na figura 30 foi alterado de acordo com o que pode ser visto na figura 33. O resultado do teste, deve apresentar

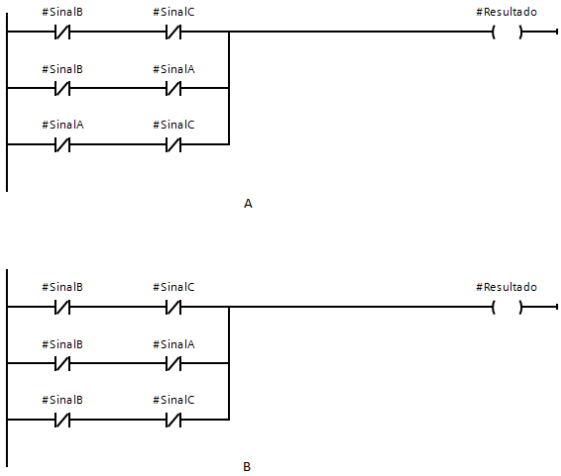


Figura 33: Primeiro exemplo de mudança realizada ao código para realização de testes. A figura A representa o código antes da mudança enquanto a figura B representa a figura após a mudança.

erro em todos os testes que envolverem a votação 2oo3. Como resultado no log, pôde-se observar erros nos testes que utilizavam as variáveis PSHH014-A, PSHH014-B e PSHH014-C, assim como pode ser visto na figura 34. Neste Log, cada ensaio é utilizado para execução de um observador e é apresentado como uma simulação no Log. Cada simulação lida com um sinal de entrada e os respectivos sinais de saída como definido pela matriz $C\&E$ e apresenta as variáveis usadas nos testes, as entradas e saídas, bem como uma breve sentença relacionando o erro que ocorreu. "PSHH"é um sinal de entrada e que indica alta pressão, enquanto "XY"são válvulas que devem ser fechadas ou abertas com a chegada de um sinal "PSHH". Os sinais PSHH- 014A, PSHH-014B e PSHH-014C são, respectivamente, a As entradas de sinal A, B e C para o bloco de votação 2oo3. A partir do resultado apresentado no log, dos equipamentos envolvidos, dos sinais de campo envolvidos, pode-se restringir possíveis locais onde o código falha em sua execução.

- Segundo caso de teste: para esse caso, utilizou-se o programa sem a introdução de erros para se verificar se a ferramenta não gerava erros não existentes. O resultado do teste não deveria apresentar

```
Log:
Total de Observadores gerados: 140
Total de testes realizados: 49

Possível Erro encontrado
Sinais envolvidos: PSHH-014A, PSHH-014B, PSHH-014C
Equipamentos envolvidos: XY-015V, XY-015B, XY-018
Válvula de ventilação acionada antes de válvula de bloqueio


Possível Erro encontrado
Sinais envolvidos: PSHH-014A, PSHH-014B, PSHH-014C
Equipamentos envolvidos: XY-026V, XY-026B, XY-029
Válvula de ventilação acionada antes de válvula de bloqueio


Possível Erro encontrado
Sinais envolvidos: Manual Shutdown
Equipamentos envolvidos: XY-026V, XY-026B, XY-029
Válvula de ventilação acionada antes de válvula de bloqueio
                        .
                        .
                        .
```

Figura 34: Parte do log que apresenta a existência de erros na execução do teste

nenhum erro gerado, apresentando somente testes realizados com sucesso. A figura 35 apresenta parte do log para o teste sem introdução de erro.

- Terceiro caso de teste: neste terceiro caso de teste, uma limitação existente na metodologia será demonstrada. Devido ao fato de a ferramenta trabalhar com leituras cíclicas do servidor OPC, e, esse realizar atualizações também cíclicas, é possível que dois sinais que surgem em ciclos diferentes no CLP sejam interpretados como tendo surgindo no mesmo ciclo dentro da ferramenta devido à alta velocidade de execução do CLP e baixa taxa de atualização do servidor OPC. Para melhor clarificação, leva-se em relação uma caso já apresentado, que encontra-se na figura 36

  Caso, o programa apresente um erro onde os sinais abrir "ventilação"e "fechar segunda válvula"ocorram dentro de um tempo menor que o tempo de atualização do servidor OPC, dois caminhos de execução da rede de petri se abrem, permitindo que dois estados diferentes sejam alcançados. Levando-se em consideração somente os lugares P = p5, p11, abrir ventilação, fechar segunda válvula, Válv. 2 fechou antes da valv vent., válv. 2 fechou antes de 5 s, como marcação inicial M_0=1,0,0,0,0,0 e ainda, que na próxima leitura, a ferramenta adicione fichas de maneira a deixar o sistema com a seguinte marcação M_1= 1,0,1, 1,0,0 as duas execuções possíveis são:

```
Log:
Total de Observadores gerados: 140
Total de testes realizados: 49

Teste realizado com sucesso
Sinais envolvidos: PSHH-014A, PSHH-014B, PSHH-014C
Equipamentos envolvidos: XY-015V, XY-015B, XY-018


Teste realizado com sucesso
Sinais envolvidos: PSHH-014A, PSHH-014B, PSHH-014C
Equipamentos envolvidos: XY-026V, XY-026B, XY-029


Teste realizado com sucesso
Sinais envolvidos: Manual Shutdown
Equipamentos envolvidos: XY-026V, XY-026B, XY-029
                         .
                         .
                         .
```

Figura 35: Parte do log que apresenta somente testes realizados com sucesso

- execução da transição t12 e marcação final Mf=0 0,1,0,1,0
- execução da transição t4 levando à marcação M2 = 0,1,0,1,0,0 e depois da transição t9 levando o sistema à marcação final Mf = 0,0,0,0,0,1

Ambas as marcações finais implicam erros no sistema, contudo, devido ao recebimento simultâneo dos dois sinais pelo servidor, a ferramenta não pode apontar com exatidão o erro ocorrido. Para esse caso, a ferramenta apresenta as duas possibilidades de erros ao usuário no log. Para testar essa situação, o seguinte erro foi introduzido no programa de CLP, assim como demonstrado na figura 37. O resultado do log apresentado para o usuário pode ser visto na figura 38.

- Para que seja mantida a concisão do documento, não serão apresentados mais testes. Os testes escolhidos para apresentação neste documento são aqueles que continham características pertinentes para a validação da ferramenta. Embora outros testes tenham sido realizados, eles não apresentam novidades relevantes de tal forma que sua abordagem aqui tornaria esta seção demasiada longa.

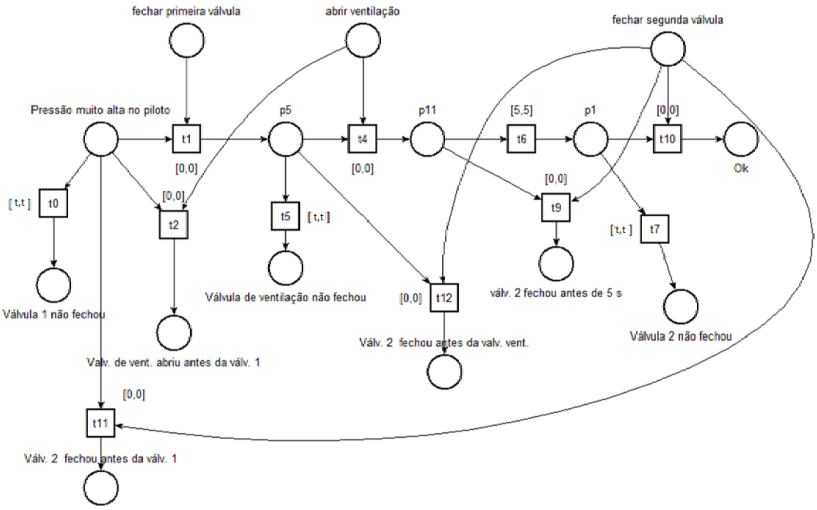


Figura 36: Rede de petri correspondente à nota 5

6.4 COMENTÁRIOS FINAIS

Neste capítulo, foi apresentada a aplicação de uma ferramenta construída com base na metodologia proposta nesta dissertação. Foi também apresentado um estudo de caso utilizado para aplicar a metodologia e validar a ferramenta desenvolvida, assim como os resultados obtidos ao se aplicar a metodologia em questão ao estudo de caso. Tudo aquilo desenvolvido para aplicação da metodologia desenvolvida, desde o ambiente de simulação até a ferramenta estão disponíveis em: `http://1drv.ms/1zfbUHv` ou `https://onedrive.live.com/redir?resid=8B54AFE170F06603!215405&authkey=!AEOWYCqHbyKHato&ithint=folder%2c`.

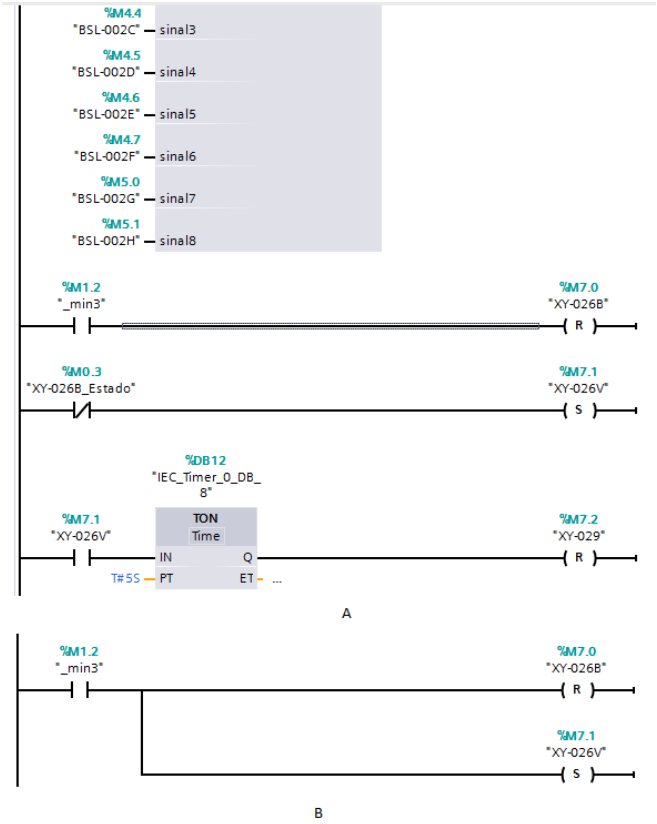


Figura 37: Exemplo utilizado para o terceiro teste. A figura A representa o código antes da mudança enquanto a figura B representa a figura após a mudança.

```
Log:
Total de Observadores gerados: 140
Total de testes realizados: 49

Sinais envolvidos: BSL-002E,BSL-002E,BSL-002E,BSL-002E,BSL-002E,BSL-002E,BSL-
002E,BSL-002E
Equipamentos envolvidos: XY-015V, XY-015B, XY-018
válvula de ventilação acionada antes de válvula de bloqueio
                ou
válvula de bloqueio acionada antes do tempo

                        .
                        .
                        .
```

Figura 38: Parte do log para o terceiro teste.

7 CONCLUSÃO

Neste trabalho foi abordado o estudo e a realização automática de testes em um forno utilizada na indústria de petróleo e gás. Foi desenvolvida uma nova metodologia para a realização de testes baseada na utilização de modelos formais construídos com redes de Petri. Esta abordagem se difere daquela atualmente utilizada ao definir automaticamente os testes realizados utilizando como base os conteúdos da matriz causa e efeito, documento criado durante o desenvolvimento do projeto, e executar os testes para posteriormente comparar os resultados àqueles previstos pela matriz causa e efeito. Nesta metodologia, os requisitos de projeto contidos na matriz causa e efeito são transformadas em redes de Petri que tem como função a representação, através de uma ferramenta formal, os mesmos requisitos. De posse dessas redes, testes são gerados e realizados de maneira a executar uma conferência do programa de CLP já existente. Os dados dos testes realizados são utilizados para, através da execução da rede de Petri, validar a execução de cada teste. O resultado é obtido através do estado alcançado pela rede o que determina a correta ou não correta execução por parte do CLP.

Foi desenvolvida uma ferramenta a fim de validar a metodologia e realizar testes sobre uma planta de uma forno utilizada pela Petrobras. Com essa ferramenta foram realizados testes sobre um CLP programado cujo objetivo era seguir os requisitos impostos pela matriz causa e efeito. Esse programa de CLP podia ou não conter erros inseridos de maneira a desrespeitar o correto funcionamento esperado. Nos testes realizados, ao se inserir um erro que desrespeitava diretamente alguma relação presente na matriz causa e efeito, esse erro era encontrado pela ferramenta.

As principais contribuições deste trabalho serão relacionadas a seguir:

- Definição de uma nova metodologia para realização automática de testes
- Relacionamento entre requisitos da matriz de causa e efeito com modelos em redes de petri
- Desenvolvimento de uma ferramenta para validar a nova metodologia

Com relação a utilização de redes de Petri para se realizar testes caixa preta de programas de CLP, vale lembrar que não existem registros deste tipo de abordagem na literatura. A realização de testes mostrou-se eficaz para encontrar erros do programa de CLP em relação aos requisitos impostos pela matriz causa e efeito e essa metodologia mostra-se como uma boa aliada durante o desenvolvimento de um projeto de automação industrial podendo diminuir o tempo gasto na realização de testes e também evitando que um programa com erros chegue a ser levado a campo para posterior descoberta dos erros. A utilização de modelos baseados em redes de Petri adicionam um formalismo no processo de automação não encontrado anteriormente. Esse formalismo permitiu não somente o desenvolvimento da nova metodologia como permite também sua expansão para a aplicação de novas técnicas que se baseiam em modelos formais. Essa nova metodologia se insere na atual metodologia de desenvolvimento de projeto utilizada pela Petrobras de maneira sutil, sem a alteração brusca da maneira com a qual o projeto é desenvolvido. Isso foi procurado para que a metodologia possuísse uma maior aceitação por parte dos engenheiros responsáveis. Este trabalho foi submetido para *IFAC Workshop on Automatic Control in Offshore Oil and Gas Production* sob o título:*Automatic Test of Safety Specifications for PLC Programs in The Oil and Gas Industry*

7.1 DIRETRIZES PARA TRABALHOS FUTUROS

A metodologia apresentada neste trabalho incluiu a utilização de um dos documentos utilizados durante um projeto de automação. Outros documentos de grande importância ainda podem ser utilizados na criação e realização de testes dos softwares de CLP. Além disso, a expansão do trabalho já realizado permite a criação de uma ferramenta mais abrangente e que pode se fazer útil em projetos de outra natureza. Levando isso em consideração, alguns temas a serem investigados futuramente, com o objetivo de se criar novos métodos de teste ou aperfeiçoar os testes realizados, são propostos:

- Expansão dos observadores disponíveis a fim de se abranger mais projetos.
- Formalização de outros documentos existente no projeto de automação da Petrobras como o memorial descritivo para utilização conjunta na realização de testes.
- Realização dos testes utilizando como base, não o CLP progra-

mado, mas o Diagrama lógico para que seja realizada uma validação prévia a programação do CLP, utilizando até *model checking* e economizando assim, tempo de projeto.

- Introdução da ferramenta em um ambiente integrado, com especializações para a inserção de informações de projeto, de dados para a simulação e para verificação de programas de CLP na indústria de petróleo e gás.

REFERÊNCIAS

ABID, N. et al. *Verification of Real-Time Specification Patterns on Time Transition Systems*. [S.l.], 2011.

ADAM, N. R.; ATLURI, V.; HUANG, W.-K. Modeling and analysis of workflows using petri nets. *Journal of Intelligent Information Systems*, Springer, v. 10, n. 2, p. 131–158, 1998.

BAIER, C.; KATOEN, J.-P. et al. *Principles of model checking*. [S.l.]: MIT press Cambridge, 2008.

BEIZER, B. *Software Testing Techniques*. New York: Van Nostrand Reinhold Company, 1990.

BERTHOMIEU, B. et al. Fiacre: an intermediate language for model verification in the topcased environment. In: *ERTS 2008*. [S.l.: s.n.], 2008.

BERTHOMIEU, P.-O. R. e. F. V. B. The tool tina - construction of abstract state spaces for petri nets and time petri nets. *European Congress on Embedded Real Time Software*, 2004.

BLACK, R. *Pragmatic Software Testing: Becoming an Effective and Efficient Test Professional*. New York: [s.n.], 2007.

BURNSTEIN, I. *Practical Software Testing*. [S.l.]: Springer-Verlag, 2003.

COMISSãO DE NORMAS TéCNICAS. *Norma técnica N-1883*. [S.l.]: Petrobras, 2002.

COMISSãO DE NORMAS TéCNICAS. *Norma técnica N-2595*. [S.l.]: Petrobras, 2012.

FARINES, J.; QUEIROZ, M. H.; CARPES, A. M. M. A model-driven engineering approach to formal verification of plc programs. *Emerging Technologies and Factory Automation*, p. 1–8, 2011.

FRANCêS, C. *Introdução à redes de Petri*. [S.l.], 2003.

GóIS, G. M. *Um Sistema Tutor Multi-Agentes no Domínio de Redes de Petri*. Dissertação (Mestrado) — UFPA, Campina Grande, 2000.

HAMADI, R.; BENATALLAH, B. A. A petri net-based model for web service composition. *Australian Computer Society*, v. 14, 2003.

HAN, R. et al. A petri net theory-based method for modeling web service-based systems. p. 1–7, 2008.

HOFFMAN, D.; STROOPER, P. A case study in class testing. *CASCON*, p. 472–782, 1993.

JACOBSON, I. et al. *The unified software development process*. [S.l.]: Addison-Wesley Reading, 1999.

JúNIOR, R. S. *Diagnóstico e tratamento de falhas críticas em sistemas instrumentados de segurança*. Dissertação (Mestrado) — USP, São Paulo, 2011.

JOUAULT, F.; KURTEV, I. Transforming models with atl. In: SPRINGER. *Satellite Events at the MoDELS 2005 Conference*. [S.l.], 2006. p. 128–138.

KANESHIRO, P. J. I. et al. Modeling of collision resolution algorithm in lonworks networks. p. 743–749, 2007.

KARHELA, T.; VILLBERG, A.; NIEMSTÖ, H. Open ontology-based integration platform for modeling and simulation in engineering. *International Journal of Modeling, Simulation, and Scientific Computing*, World Scientific, v. 3, n. 02, 2012.

KIEPUSZEWSKI, B.; HOFSTEDE, A. H. ter; AALST, W. M. van der. Fundamentals of control flow in workflows. *Acta Informatica*, Springer, v. 39, n. 3, p. 143–209, 2003.

KUHN, D. R.; KACKER, R. N.; LEI, Y. *Practical combinatorial testing*. [S.l.], 2010. v. 800, 142 p.

KUHN, D. R.; WALLACE, D. R.; GALLO, J. A. Software fault interactions and implications for software testing. *Software Engineering, IEEE Transactions on*, IEEE, v. 30, n. 6, p. 418–421, 2004.

LEE, J.-S.; ZHOU, M.-C.; HSU, P.-L. An application of petri nets to supervisory control for human-computer interactive systems. *Industrial Electronics, IEEE Transactions on*, IEEE, v. 52, n. 5, p. 1220–1226, 2005.

LI, Z.; ZHOU, M. Control of elementary and dependent siphons in petri nets and their application. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, IEEE, v. 38, n. 1, p. 133–148, 2008.

MACIEL, P. R.; LINS, R. D.; CUNHA, P. R. *Introdução às redes de Petri e aplicações*. [S.l.]: UNICAMP-Instituto de Computacao, 1996.

MALDONADO, J. C. *Critérios potenciais usos: Uma contribuição ao teste estrutural de software*. Tese (Doutorado), 1991.

MALDONADO, J. C. et al. Introduçao ao teste de software. *São Carlos*, 2004.

MCGREGOR, J. D. *Functional testing of classes*. San Francisco, 1994.

MERZ, S. Model checking: A tutorial overview. In: *Modeling and verification of parallel processes*. [S.l.]: Springer, 2001. p. 3–38.

MORALES, R. A. G.; MELO, J. G.; MIYAGI, P. E. Diagnosis and treatment of faults in productive systems based on bayesian networks and petri net. In: IEEE. *Automation Science and Engineering, 2007. CASE 2007. IEEE International Conference on*. [S.l.], 2007. p. 357–362.

MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, IEEE, v. 77, n. 4, p. 541–580, 1989.

MYERS, G. J. *The art of software testing*. New York: Wiley, 1979.

NASSAR, M. G. V. et al. Modeling and analysing of the material entry flow system in a pickling line process using petri net. In: *ABCM Symposium Series in Mechatronics*. [S.l.: s.n.], 2008. v. 3, p. 444–453.

OLIVEIRA, C. Simulação de redes de petri em ambiente java. 2006.

OLIVEIRA LEANDRO DIAS DA SILVA, A. P. K. C. G. Kezia de V. Uma abordagem para geração e execução de casos de teste em programas de sistemas instrumentados de segurança. *CBA*, 2012.

PAKONEN, A. et al. A toolset for model checking of plc software. p. 1–6, 2013.

PATTON, R. *Software testing*. [S.l.]: Sams Indianapolis, 2001.

PERRY, D. E.; KAISER, G. E. Adequate testing and object-oriented programming. *Journal of Object-Oriented Programming*, SIGS Publications, v. 2, n. 5, p. 13–19, 1990.

PRESSMAN, R. S. *Software Engineering - A Practitioner?s Approach*. [S.l.]: McGraw-Hill, 1997.

SILVA, M. K. *Metodologia para detecção e tratamento de falhas em sistemas de manufatura através de rede de Petri*. Tese (Doutorado) — USP, São Paulo, 2002.

SILVA, M. K. *Pré-Detalhamento da Instrumentação e Automação de um Forno Industrial de um Complexo Petroquímico*. Dissertação (Mestrado) — DAS/UFSC, Florianópolis, 2009.

SKOGDALE, J. E.; SMOGELI, O. *Looking Forward-Reliability of Safety Critical Control Systems on Offshore Drilling Vessels*. [S.l.], 2011.

VERNADAT, F. et al. The topcased project-a toolkit in open-source for critical applications and system development. *Data Systems In Aerospace (DASIA), Berlin, Germany*, v. 22, n. 05, p. 2006–25, 2006.

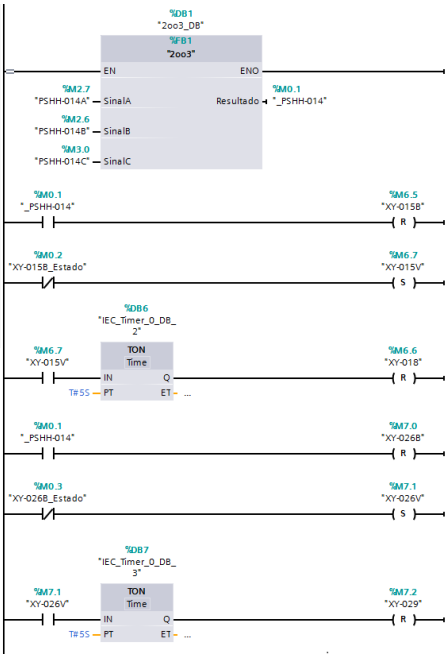
XU, X. et al. A novel modeling design method for automated storage and retrieval system based on petri nets. p. 2046–2051, 2007.

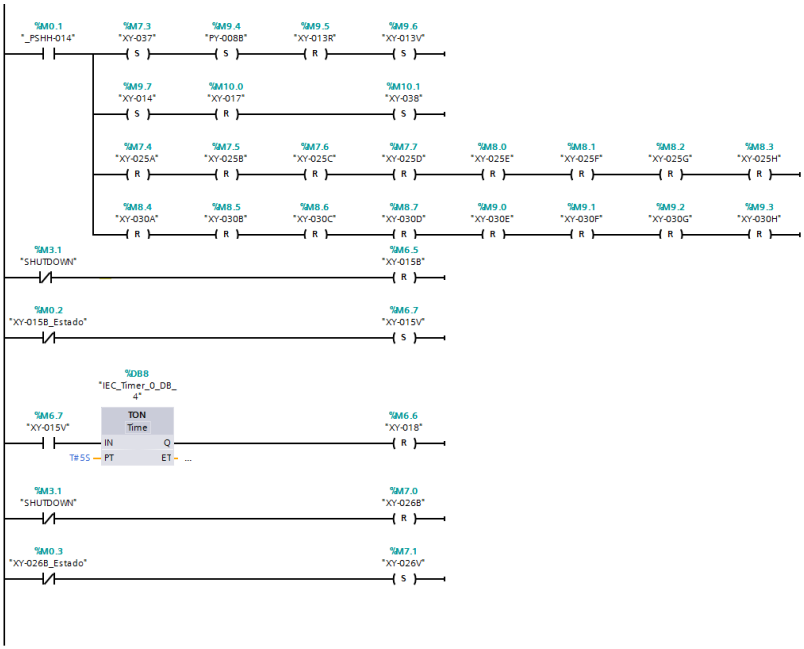
YOO, T.; JEONG, B.; CHO, H. A petri nets based functional validation for services composition. *Expert Systems with Applications*, Elsevier, v. 37, n. 5, p. 3768–3776, 2010.

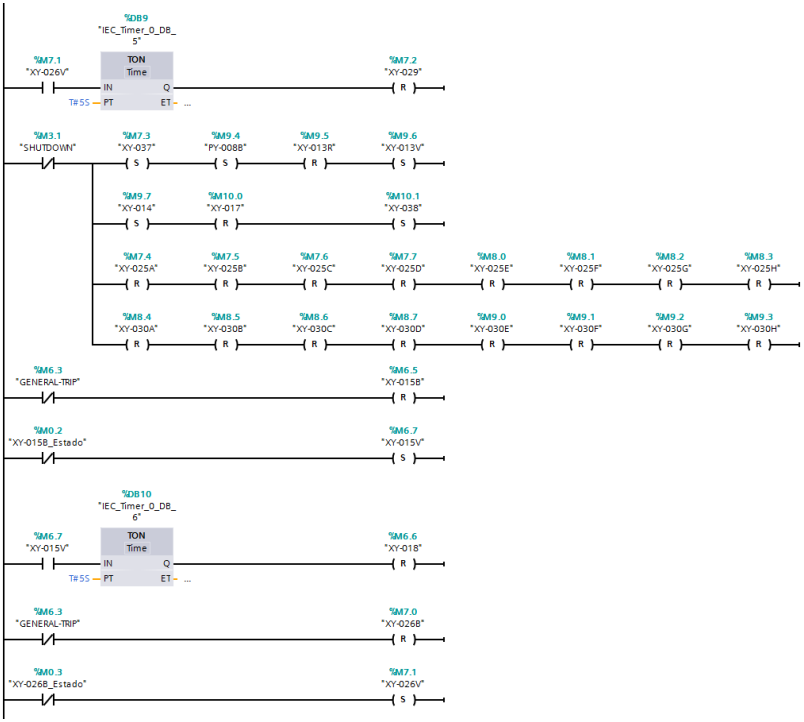
ZURAWSKI, R.; ZHOU, M. Petri nets and industrial applications: A tutorial. *Industrial Electronics, IEEE Transactions on*, IEEE, v. 41, n. 6, p. 567–583, 1994.

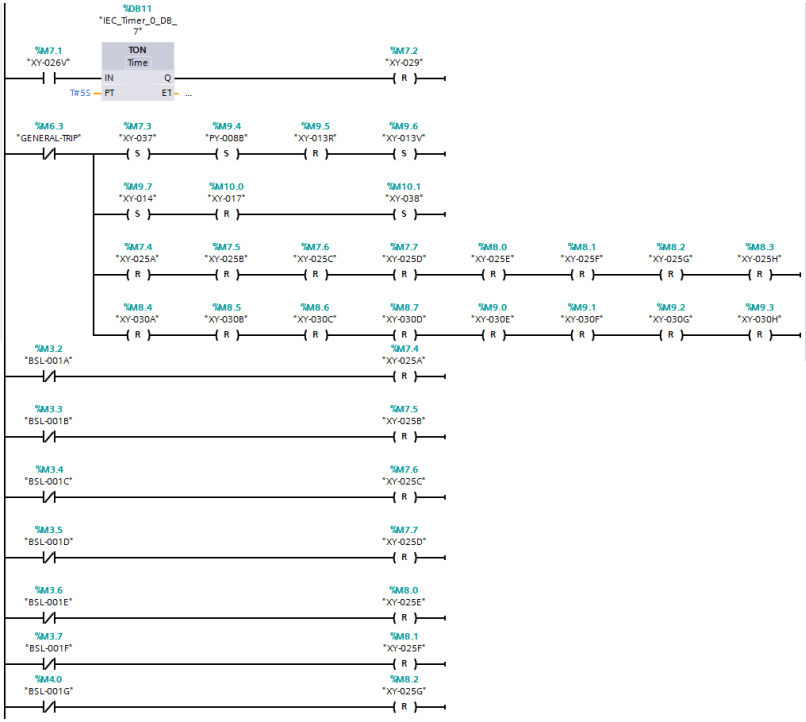
ZURAWSKI, R.; ZHOU, M. Software fault interactions and implications for software testing. *IEEE Trans. on Software Engineering*, v. 30, 2004.

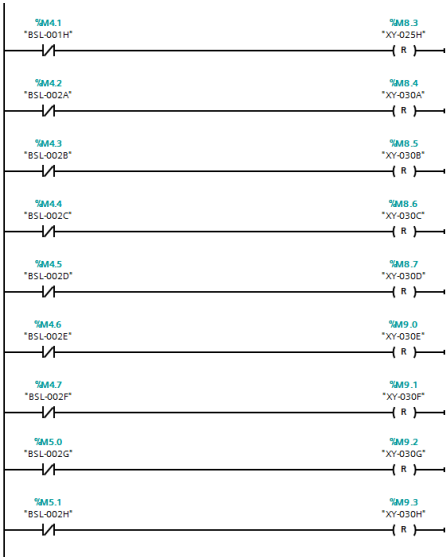
APÊNDICE A – Código Ladder Utilizado para Validação

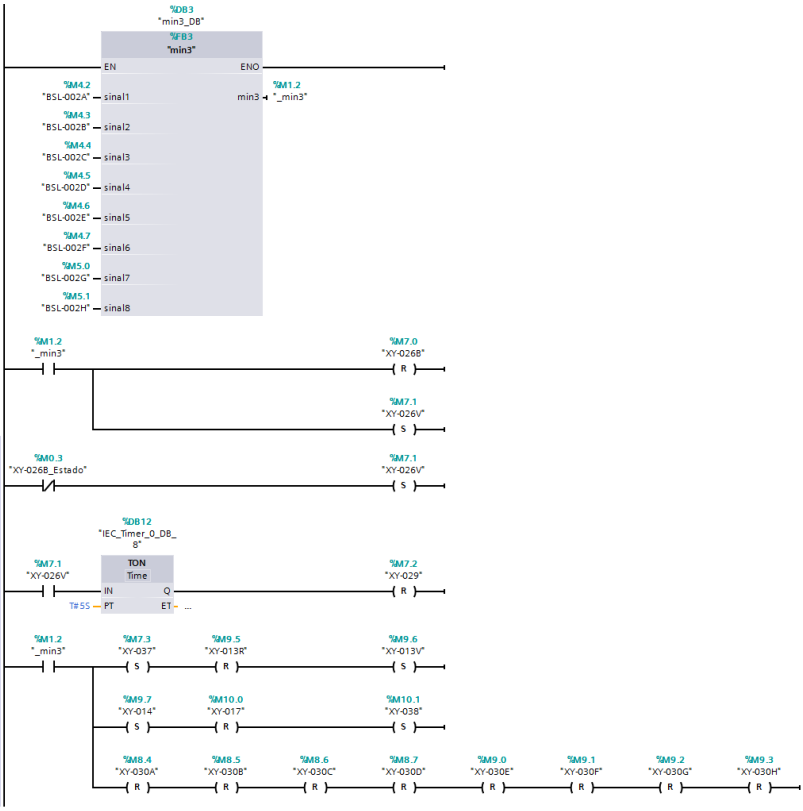

%DB1
"2oo3_DB"
%FB1
"2oo3"
EN
ENO
%M2.7
"PSHH-014A"
SinalA
Resultado
%M0.1
"_PSHH-014"
%M2.6
"PSHH-014B"
SinalB
%M3.0
"PSHH-014C"
SinalC
%M0.1
"_PSHH-014"
%M6.5
"XY-015B"
R
%M0.2
"XY-015B_Estado"
%M6.7
"XY-015V"
S
%DB6
"IEC_Timer_0_DB_2"
%M6.7
"XY-015V"
TON
Time
IN
Q
T#5S
PT
ET
%M6.6
"XY-018"
R
%M0.1
"_PSHH-014"
%M7.0
"XY-026B"
R
%M0.3
"XY-026B_Estado"
%M7.1
"XY-026V"
S
%DB7
"IEC_Timer_0_DB_3"
%M7.1
"XY-026V"
TON
Time
IN
Q
T#5S
PT
ET
%M7.2
"XY-029"
R

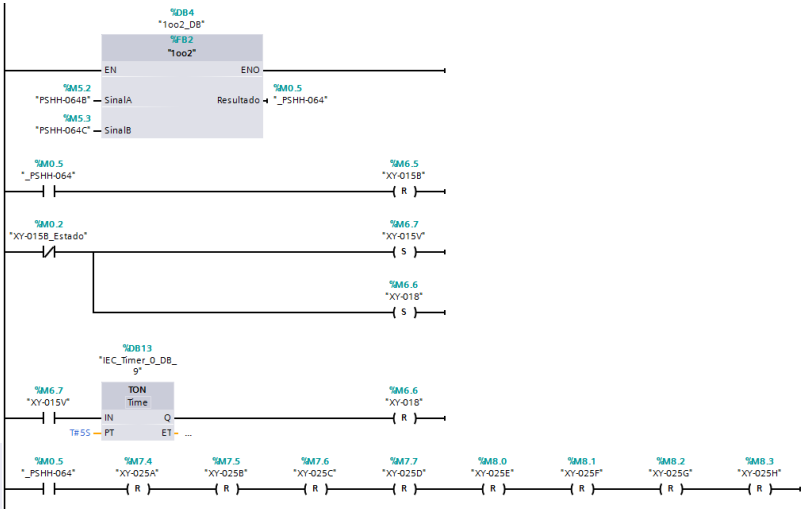

%M0.1
"_PSHH-014"
%M7.3
"XY-037"
%M9.4
"PY-008B"
%M9.5
"XY-013R"
%M9.6
"XY-013V"
%M9.7
"XY-014"
%M10.0
"XY-017"
%M10.1
"XY-038"
%M7.4
"XY-025A"
%M7.5
"XY-025B"
%M7.6
"XY-025C"
%M7.7
"XY-025D"
%M8.0
"XY-025E"
%M8.1
"XY-025F"
%M8.2
"XY-025G"
%M8.3
"XY-025H"
%M8.4
"XY-030A"
%M8.5
"XY-030B"
%M8.6
"XY-030C"
%M8.7
"XY-030D"
%M9.0
"XY-030E"
%M9.1
"XY-030F"
%M9.2
"XY-030G"
%M9.3
"XY-030H"
%M3.1
"SHUTDOWN"
%M6.5
"XY-015B"
%M0.2
"XY-015B_Estado"
%M6.7
"XY-015V"
%DB8
"IEC_Timer_0_DB_
4"
%M6.7
"XY-015V"
TON
Time
IN
Q
T#5S
PT
ET
%M6.6
"XY-018"
%M3.1
"SHUTDOWN"
%M7.0
"XY-026B"
%M0.3
"XY-026B_Estado"
%M7.1
"XY-026V"

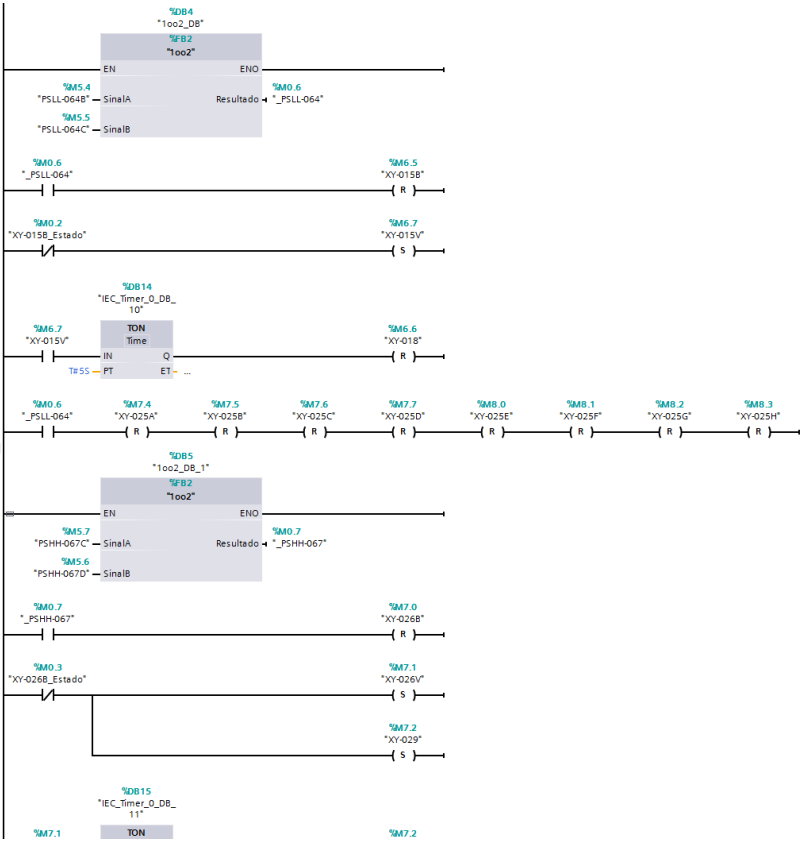

%DB9
"IEC_Timer_0_DB_5"
%M7.1
"XY-026V"
TON
Time
IN
Q
T#5S
PT
ET
%M7.2
"XY-029"
R
%M3.1
"SHUTDOWN"
%M7.3
"XY-037"
%M9.4
"PY-008B"
%M9.5
"XY-013R"
%M9.6
"XY-013V"
S
%M9.7
"XY-014"
%M10.0
"XY-017"
%M10.1
"XY-038"
%M7.4
"XY-025A"
%M7.5
"XY-025B"
%M7.6
"XY-025C"
%M7.7
"XY-025D"
%M8.0
"XY-025E"
%M8.1
"XY-025F"
%M8.2
"XY-025G"
%M8.3
"XY-025H"
%M8.4
"XY-030A"
%M8.5
"XY-030B"
%M8.6
"XY-030C"
%M8.7
"XY-030D"
%M9.0
"XY-030E"
%M9.1
"XY-030F"
%M9.2
"XY-030G"
%M9.3
"XY-030H"
%M6.3
"GENERAL-TRIP"
%M6.5
"XY-015B"
%M0.2
"XY-015B_Estado"
%M6.7
"XY-015V"
%DB10
"IEC_Timer_0_DB_6"
%M6.6
"XY-018"
%M7.0
"XY-026B"
%M0.3
"XY-026B_Estado"
%M7.1
"XY-026V"

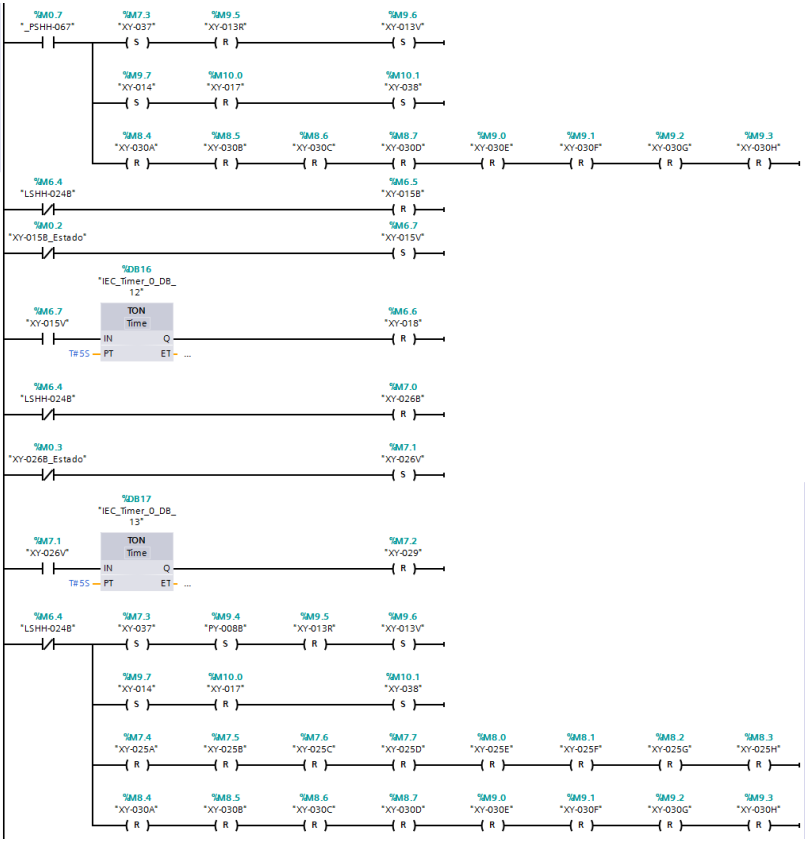

%DB11
"IEC_Timer_0_DB_7"
TON
Time
IN
PT
Q
ET
T#5S
%M7.1
"XY-026V"
%M7.2
"XY-029"
%M6.3
"GENERAL-TRIP"
%M7.3
"XY-037"
%M9.4
"PY-008B"
%M9.5
"XY-013R"
%M9.6
"XY-013V"
%M9.7
"XY-014"
%M10.0
"XY-017"
%M10.1
"XY-038"
%M7.4
"XY-025A"
%M7.5
"XY-025B"
%M7.6
"XY-025C"
%M7.7
"XY-025D"
%M8.0
"XY-025E"
%M8.1
"XY-025F"
%M8.2
"XY-025G"
%M8.3
"XY-025H"
%M8.4
"XY-030A"
%M8.5
"XY-030B"
%M8.6
"XY-030C"
%M8.7
"XY-030D"
%M9.0
"XY-030E"
%M9.1
"XY-030F"
%M9.2
"XY-030G"
%M9.3
"XY-030H"
%M3.2
"BSL-001A"
%M7.4
"XY-025A"
%M3.3
"BSL-001B"
%M7.5
"XY-025B"
%M3.4
"BSL-001C"
%M7.6
"XY-025C"
%M3.5
"BSL-001D"
%M7.7
"XY-025D"
%M3.6
"BSL-001E"
%M8.0
"XY-025E"
%M3.7
"BSL-001F"
%M8.1
"XY-025F"
%M4.0
"BSL-001G"
%M8.2
"XY-025G"

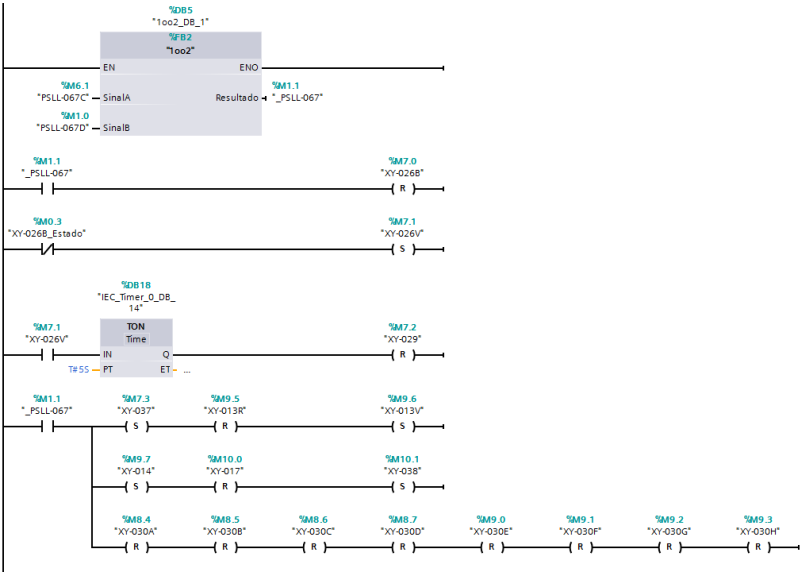

%M4.1
"BSL-001H"
%M8.3
"XY-025H"
R
%M4.2
"BSL-002A"
%M8.4
"XY-030A"
R
%M4.3
"BSL-002B"
%M8.5
"XY-030B"
R
%M4.4
"BSL-002C"
%M8.6
"XY-030C"
R
%M4.5
"BSL-002D"
%M8.7
"XY-030D"
R
%M4.6
"BSL-002E"
%M9.0
"XY-030E"
R
%M4.7
"BSL-002F"
%M9.1
"XY-030F"
R
%M5.0
"BSL-002G"
%M9.2
"XY-030G"
R
%M5.1
"BSL-002H"
%M9.3
"XY-030H"
R

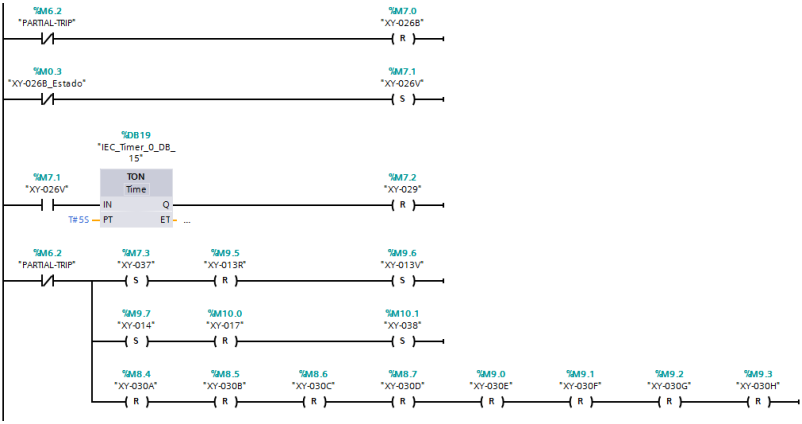

%DB3
"min3_DB"
%FB3
"min3"
EN
ENO
%M4.2
"BSL-002A"
sinal1
%M1.2
min3
"_min3"
%M4.3
"BSL-002B"
sinal2
%M4.4
"BSL-002C"
sinal3
%M4.5
"BSL-002D"
sinal4
%M4.6
"BSL-002E"
sinal5
%M4.7
"BSL-002F"
sinal6
%M5.0
"BSL-002G"
sinal7
%M5.1
"BSL-002H"
sinal8
%M1.2
"_min3"
%M7.0
"XY-026B"
R
%M7.1
"XY-026V"
S
%M0.3
"XY-026B_Estado"
%M7.1
"XY-026V"
S
%DB12
"IEC_Timer_0_DB_
8"
%M7.1
"XY-026V"
TON
Time
IN
Q
T#5S
PT
ET
%M7.2
"XY-029"
R
%M1.2
"_min3"
%M7.3
"XY-037"
S
%M9.5
"XY-013R"
R
%M9.6
"XY-013V"
S
%M9.7
"XY-014"
S
%M10.0
"XY-017"
R
%M10.1
"XY-038"
S
%M8.4
"XY-030A"
R
%M8.5
"XY-030B"
R
%M8.6
"XY-030C"
R
%M8.7
"XY-030D"
R
%M9.0
"XY-030E"
R
%M9.1
"XY-030F"
R
%M9.2
"XY-030G"
R
%M9.3
"XY-030H"
R


%DB4
"1oo2_DB"
%FB2
"1oo2"
EN
ENO
%M5.2
"PSHH-064B"
SinalA
Resultado
%M0.5
"_PSHH-064"
%M5.3
"PSHH-064C"
SinalB
%M0.5
"_PSHH-064"
%M6.5
"XY-015B"
R
%M0.2
"XY-015B_Estado"
%M6.7
"XY-015V"
S
%M6.6
"XY-018"
S
%DB13
"IEC_Timer_0_DB_9"
%M6.7
"XY-015V"
TON
Time
IN
Q
%M6.6
"XY-018"
R
T#5S
PT
ET
%M0.5
"_PSHH-064"
%M7.4
"XY-025A"
%M7.5
"XY-025B"
%M7.6
"XY-025C"
%M7.7
"XY-025D"
%M8.0
"XY-025E"
%M8.1
"XY-025F"
%M8.2
"XY-025G"
%M8.3
"XY-025H"

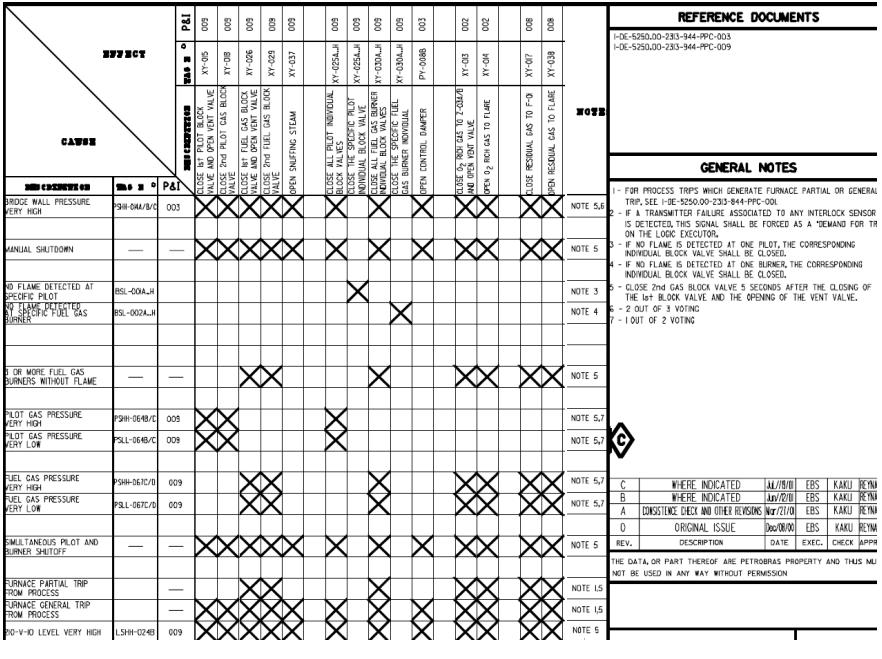

%DB4
"1oo2_DB"
%FB2
"1oo2"
EN
ENO
%M5.4
"PSLL-064B"
SinalA
Resultado
%M0.6
"_PSLL-064"
%M5.5
"PSLL-064C"
SinalB
%M0.6
"_PSLL-064"
%M6.5
"XY-015B"
R
%M0.2
"XY-015B_Estado"
%M6.7
"XY-015V"
S
%DB14
"IEC_Timer_0_DB_10"
%M6.7
"XY-015V"
TON
Time
IN
Q
%M6.6
"XY-018"
T#5S
PT
ET
%M0.6
"_PSLL-064"
%M7.4
"XY-025A"
%M7.5
"XY-025B"
%M7.6
"XY-025C"
%M7.7
"XY-025D"
%M8.0
"XY-025E"
%M8.1
"XY-025F"
%M8.2
"XY-025G"
%M8.3
"XY-025H"
%DB5
"1oo2_DB_1"
%FB2
"1oo2"
EN
ENO
%M5.7
"PSHH-067C"
SinalA
Resultado
%M0.7
"_PSHH-067"
%M5.6
"PSHH-067D"
SinalB
%M0.7
"_PSHH-067"
%M7.0
"XY-026B"
R
%M0.3
"XY-026B_Estado"
%M7.1
"XY-026V"
S
%M7.2
"XY-029"
S
%DB15
"IEC_Timer_0_DB_11"
%M7.1
TON
%M7.2


%M0.7
"_PSHH-067"
%M7.3
"XY-037"
S
%M9.5
"XY-013R"
R
%M9.6
"XY-013V"
S
%M9.7
"XY-014"
S
%M10.0
"XY-017"
R
%M10.1
"XY-038"
S
%M8.4
"XY-030A"
R
%M8.5
"XY-030B"
R
%M8.6
"XY-030C"
R
%M8.7
"XY-030D"
R
%M9.0
"XY-030E"
R
%M9.1
"XY-030F"
R
%M9.2
"XY-030G"
R
%M9.3
"XY-030H"
R
%M6.4
"LSHH-024B"
%M6.5
"XY-015B"
R
%M0.2
"XY-015B_Estado"
%M6.7
"XY-015V"
S
%DB16
"IEC_Timer_0_DB_
12"
%M6.7
"XY-015V"
TON
Time
IN
Q
T#5S
PT
ET
...
%M6.6
"XY-018"
R
%M6.4
"LSHH-024B"
%M7.0
"XY-026B"
R
%M0.3
"XY-026B_Estado"
%M7.1
"XY-026V"
S
%DB17
"IEC_Timer_0_DB_
13"
%M7.1
"XY-026V"
TON
Time
IN
Q
T#5S
PT
ET
...
%M7.2
"XY-029"
R
%M6.4
"LSHH-024B"
%M7.3
"XY-037"
S
%M9.4
"PY-008B"
S
%M9.5
"XY-013R"
R
%M9.6
"XY-013V"
S
%M9.7
"XY-014"
S
%M10.0
"XY-017"
R
%M10.1
"XY-038"
S
%M7.4
"XY-025A"
R
%M7.5
"XY-025B"
R
%M7.6
"XY-025C"
R
%M7.7
"XY-025D"
R
%M8.0
"XY-025E"
R
%M8.1
"XY-025F"
R
%M8.2
"XY-025G"
R
%M8.3
"XY-025H"
R
%M8.4
"XY-030A"
R
%M8.5
"XY-030B"
R
%M8.6
"XY-030C"
R
%M8.7
"XY-030D"
R
%M9.0
"XY-030E"
R
%M9.1
"XY-030F"
R
%M9.2
"XY-030G"
R
%M9.3
"XY-030H"
R


%DB5
"1oo2_DB_1"
%FB2
"1oo2"
EN
ENO
%M6.1
"PSLL-067C"
SinalA
Resultado
%M1.1
"_PSLL-067"
%M1.0
"PSLL-067D"
SinalB
%M1.1
"_PSLL-067"
%M7.0
"XY-026B"
R
%M0.3
"XY-026B_Estado"
%M7.1
"XY-026V"
S
%DB18
"IEC_Timer_0_DB_14"
%M7.1
"XY-026V"
TON
Time
IN
Q
%M7.2
"XY-029"
T#5S
PT
ET
%M1.1
"_PSLL-067"
%M7.3
"XY-037"
%M9.5
"XY-013R"
%M9.6
"XY-013V"
%M9.7
"XY-014"
%M10.0
"XY-017"
%M10.1
"XY-038"
%M8.4
"XY-030A"
%M8.5
"XY-030B"
%M8.6
"XY-030C"
%M8.7
"XY-030D"
%M9.0
"XY-030E"
%M9.1
"XY-030F"
%M9.2
"XY-030G"
%M9.3
"XY-030H"


%M6.2
"PARTIAL-TRIP"
%M7.0
"XY-026B"
R
%M0.3
"XY-026B_Estado"
%M7.1
"XY-026V"
S
%DB19
"IEC_Timer_0_DB_15"
%M7.1
"XY-026V"
TON
Time
IN
Q
%M7.2
"XY-029"
T#5S
PT
ET
%M6.2
"PARTIAL-TRIP"
%M7.3
"XY-037"
%M9.5
"XY-013R"
%M9.6
"XY-013V"
%M9.7
"XY-014"
%M10.0
"XY-017"
%M10.1
"XY-038"
%M8.4
"XY-030A"
%M8.5
"XY-030B"
%M8.6
"XY-030C"
%M8.7
"XY-030D"
%M9.0
"XY-030E"
%M9.1
"XY-030F"
%M9.2
"XY-030G"
%M9.3
"XY-030H"

APÊNDICE B – Matriz Causa & Efeito Utilizado no Estudo de Caso



REFERENCE DOCUMENTS

I-DE-5250.00-233-944-PPC-003
I-DE-5250.00-233-944-PPC-009

GENERAL NOTES

1 - FOR PROCESS TRIPS WHICH GENERATE FURNACE PARTIAL OR GENERAL TRIP, SEE I-DE-5250.00-233-944-PPC-001
2 - IF A TRANSMITTER FAILURE ASSOCIATED TO ANY INTERLOCK SENSOR IS DETECTED, THIS SIGNAL SHALL BE FORCED AS A "DEMAND FOR TR ON THE LOGIC EXECUTOR.
3 - IF NO FLAME IS DETECTED AT ONE PILOT, THE CORRESPONDING INDIVIDUAL BLOCK VALVE SHALL BE CLOSED.
4 - IF NO FLAME IS DETECTED AT ONE BURNER, THE CORRESPONDING INDIVIDUAL BLOCK VALVE SHALL BE CLOSED.
5 - CLOSE 2nd GAS BLOCK VALVE 5 SECONDS AFTER THE CLOSING OF THE 1st BLOCK VALVE AND THE OPENING OF THE VENT VALVE.
6 - 2 OUT OF 3 VOTING
7 - 1 OUT OF 2 VOTING

REV.	DESCRIPTION	DATE	EXEC.	CHECK	APPR
C	WHERE INDICATED	JUL/15/11	EBS	KAKU	REYNA
B	WHERE INDICATED	Jun/2/11	EBS	KAKU	REYNA
A	CONSISTENCE CHECK AND OTHER REVISIONS	Mar/21/0	EBS	KAKU	REYNA
0	ORIGINAL ISSUE	Dec/08/00	EBS	KAKU	REYNA