

Hallan William Veiga

**MÉTODO PARA TESTE AUTOMATIZADO DE  
SISTEMAS INSTRUMENTADOS DE SEGURANÇA EM  
PLATAFORMAS DE PETRÓLEO**

Dissertação submetida ao Programa  
de Pós Graduação em Engenharia de  
Automação e Sistemas para a obtenção  
do Grau de Mestre.

Universidade Federal de Santa Cata-  
rina

Orientador: Prof. Dr. Max Hering de  
Queiroz

Universidade Federal de Santa Cata-  
rina

Coorientador: Prof. Dr. Jean-Marie  
Farines

Florianópolis

2018

Ficha de identificação da obra elaborada pelo autor,  
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Veiga, Hallan William  
Método para Teste Automatizado de Sistemas  
Instrumentados de Segurança em Plataformas de  
Petróleo / Hallan William Veiga ; orientador, Max  
Hering de Queiroz, coorientador, Jean-Marie  
Farines, 2018.  
95 p.

Dissertação (mestrado) - Universidade Federal de  
Santa Catarina, Centro Tecnológico, Programa de Pós  
Graduação em Engenharia de Automação e Sistemas,  
Florianópolis, 2018.

Inclui referências.

1. Engenharia de Automação e Sistemas. 2. Sistema  
Instrumentado de Segurança. 3. Teste de  
conformidade. 4. Controlador Lógico Programável. 5.  
Métodos Formais. I. de Queiroz, Max Hering . II.  
Farines, Jean-Marie . III. Universidade Federal de  
Santa Catarina. Programa de Pós-Graduação em  
Engenharia de Automação e Sistemas. IV. Título.

Hallan William Veiga

**MÉTODO PARA TESTE AUTOMATIZADO DE  
SISTEMAS INSTRUMENTADOS DE SEGURANÇA EM  
PLATAFORMAS DE PETRÓLEO**

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

Florianópolis, 16 de fevereiro 2018.

---

Prof. Daniel Ferreira Coutinho  
Coordenador  
Universidade Federal de Santa Catarina

**Banca Examinadora:**

---

Prof. Dr. Max Hering de Queiroz  
Universidade Federal de Santa Catarina  
Orientador

---

Prof. Dr. Jean-Marie Farines  
Universidade Federal de Santa Catarina  
Coorientador

---

Prof. Dr. Rômulo Silva de Oliveira  
Universidade Federal de Santa Catarina

---

Dr. Rodrigo Tacla Saad  
Universidade Federal de Santa Catarina

---

Eng. Marcelo Lopes de Lima  
CENPES - Petrobras

Este trabalho é dedicado à minha família,  
ao meus professores, aos meus colegas e  
amigos.



## AGRADECIMENTOS

Agradeço primeiramente à minha noiva Karolina Nunes, aos meus pais Geraldo Veiga e Jucélia Veiga e minha irmã Amanda Veiga pelo incentivo motivacional e financeiro para a realização desse trabalho.

Ao meu orientador Prof. Dr. Max Hering de Queiroz e ao coorientador Prof. Dr. Jean-Marie Farines pelos ensinamentos e disposição para discussão de ideias. Aos professoras do PPGEAS e os que ajudaram na minha formação.

Aos colegas do grupo de pesquisa pelo auxílio na resolução de dúvidas provenientes do trabalho, em especial pelos engenheiros Luiz Paulo Reis e Guilherme Saito, e aos graduandos Rafael Scarduelli e Thales Medeiros no suporte da programação da ferramenta automática de testes. Aos amigos que ajudaram a compartilhar experiência pessoal e profissional.

Agradecimentos também à CAPES e Petrobras pelo suporte financeiro. Aos engenheiros e pesquisadores da Petrobras/CENPES/P-DEP/TOOL e Petrobras/UO-RIO/IPP/AUT pela cooperação técnica.

À Deus por oferecer as mais grandiosas oportunidades e colocar em meu caminho todas essas grandes pessoas.





”Todos pensam em mudar a humanidade  
mas ninguém pensa em mudar a si mesmo.”  
(Leon Tolstoi)



## RESUMO

O Sistema Instrumentado de Segurança (SIS) é responsável pelo controle das funções de segurança na indústria de processos. Esse sistema automático conta com sensores para identificar situações de risco, um Controlador Lógico Programável (CLP) e atuadores para conduzir a planta para um estado seguro. As normas exigem testes para identificar falhas na lógica de intertravamento desse sistema. Esta dissertação apresenta um método para executar um teste automatizado do dispositivo de controle da lógica do SIS seguindo as especificações lógicas de segurança descritas na Matriz Causa e Efeito (MCE). A partir desse documento são gerados casos de teste utilizando uma estratégia caixa-preta de geração de casos de teste conhecida como CEG-BOR, e ainda modelos de observadores de Redes de Petri que são o oráculo do teste. Os modelos produzem um veredito que avalia o comportamento da lógica do CLP e compara com a especificação. Uma ferramenta experimental foi desenvolvida para editar a MCE, gerar os casos de teste e os observadores, executar o teste do CLP e ainda apresentar as não conformidades encontradas. A validação do método e da ferramenta foi realizada utilizando os programas de CLP e a documentação do SIS de uma plataforma de petróleo da Petrobras. Os resultados desse trabalho indicam que o método contribui para o diagnóstico de falhas de execução da implementação lógica complementando as práticas existentes de testes do SIS na indústria.

**Palavras-chave:** Teste de conformidade; Sistema Instrumentado de Segurança; métodos formais; CLP; plataformas de petróleo.



## ABSTRACT

The Safety Instrumented System (SIS) is responsible for the control of safety functions at process industry. This system has sensors to identify hazardous situations, Programmable Logic Controller (PLC) and actuators to lead the process back to a safe state, avoiding the origin of accidents. The standards require a test aiming to identify operational failures of this system. This work presents a method for automatic conformance testing of safety specifications represented by a Cause and Effect Matrix (CEM) for PLC in charge of SIS. Test cases are automatically generated from the CEM using a black-box strategy (CEG-BOR strategy). Also Petri nets models are generated from the CEM. These models are the oracle of the test and produce a verdict that evaluates the behavior of the PLC logic and compares with the specification. An experimental tool was developed to edit the CEM, generate the test cases and the models, perform the PLC test and also present the nonconformities encountered. The method and tool validation was performed using the PLC programs and the SIS documentation of a Petrobras oil platform. The results of this work indicate that the method contributes to the diagnosis of execution failures of the logical implementation complementing the existing SIS testing practices in the industry.

**Keywords:** Conformance test; Safety Instrumented Systems; formal methods; PLC; pffshore platforms.



## LISTA DE FIGURAS

Figura 1	Camadas de proteção.....	30
Figura 2	Processo de desenvolvimento do SIS na Petrobras.....	34
Figura 3	Estrutura Básica de uma Matriz Causa e Efeito.....	35
Figura 4	Simbologia definida pela ET-3000.00-1200-800-PGT-006 (2000).....	36
Figura 5	Matriz Causa e Efeito Exemplo.....	37
Figura 6	Visão Básica do Teste de Conformidade.....	38
Figura 7	Teste baseado em modelo de Autômatos temporizados.	41
Figura 8	Procedimento para teste automático da lógica do CLP.	42
Figura 9	Visão Detalhada do método.....	43
Figura 10	Grafo de Causa e Efeito.....	48
Figura 11	Aplicação do método CEG BOR - Grafo de Causa e Efeito ( $N5 = (a*b) + c$ ).....	50
Figura 12	Grafo de Causa e Efeito que representa uma votação de um grupo de entradas (2o3).....	52
Figura 13	Matriz Causa e Efeito Exemplo.....	54
Figura 14	Exemplo de Modelo de Oráculo.....	60
Figura 15	Módulo de Diagnóstico.....	62
Figura 16	Módulo Lógico.....	64
Figura 17	Módulo de Temporização - Período de ativação de T segundos.....	65
Figura 18	Módulo de Leitura/Votação.....	66
Figura 19	Matriz Causa e Efeito.....	66
Figura 20	Exemplo 1 - Oráculo do efeito <b>GC-600</b> .....	67
Figura 21	Oráculo do efeito <b>FC-700</b> .....	69
Figura 22	Solução para a automatização do Teste de Conformidade	72
Figura 23	Tela do Editor de Matriz Causa e Efeito.....	73
Figura 24	Tela Principal.....	78
Figura 25	Tela de Execução do teste da Matriz Causa e Efeito ...	79
Figura 26	Tela de configuração da comunicação.....	80
Figura 27	Matriz Causa e Efeito Exemplo.....	81
Figura 28	Código Ladder em conformidade com a especificação...	81
Figura 29	Código Ladder com inconformidades.....	82

Figura 30 Tela com o veredito do teste.....	83
Figura 31 Log do resultado .....	83
Figura 32 CLP e giga de testes.....	84
Figura 33 Causas/Efeitos da MCE com falhas seguras.....	86



## LISTA DE TABELAS

Tabela 1	Tabela de decisões do grafo da Figura 10 .....	49
Tabela 2	Definição de Casos de Teste- Efeito ESD-101 .....	55
Tabela 3	Casos de Teste - Efeito FD-101 .....	56
Tabela 4	Casos de Teste - Efeito FC-102 .....	56
Tabela 5	Tabela de Decisões do Exemplo.....	57
Tabela 6	Diagnóstico do Oráculo.....	61
Tabela 7	Resultados do Teste Proposto da MCE do subsistema de Fogo e Gás .....	84
Tabela 8	Resultados do Teste Exaustivo da MCE do subsistema de Fogo e Gás .....	85



## **LISTA DE ABREVIATURAS E SIGLAS**

SIS	Sistema Instrumentado de Segurança
CLP	Controlador Lógico Programável
SUT	System Under Test
SIL	Safety Integrity Level
TAF	Teste de Aceitação de Fábrica
P&ID	Piping and Instrumentation Diagram
FAT	Factory Acceptancy Test
MBT	Model Based Testing
FSM	Finite State Machine
LBT	Learning Based Testing
XML	eXtensible Markup Language



## LISTA DE SÍMBOLOS

- ⊗ Operação de Exclusão Mútua
- ∪ Operação de União



## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO</b> .....	25
1.1	OBJETIVOS .....	27
1.2	ESTRUTURA DO DOCUMENTO .....	28
<b>2</b>	<b>PROPOSTA DE MÉTODO DE TESTES</b> .....	29
2.1	SISTEMA INSTRUMENTADO DE SEGURANÇA .....	29
2.2	DESENVOLVIMENTO DO PROJETO DO SIS .....	30
<b>2.2.1</b>	<b>Causas de acidentes do SIS</b> .....	32
2.3	METODOLOGIA DE DESENVOLVIMENTO DA PE- TROBRAS .....	33
<b>2.3.1</b>	<b>Diretrizes para elaboração da Matriz Causa e Efeito</b>	34
2.4	MÉTODOS PARA TESTE DE PROGRAMAS DE CLP .	38
2.5	PROPOSIÇÃO DE MÉTODO AUTOMÁTICO DE TES- TES .....	41
2.6	CONSIDERAÇÕES FINAIS DO CAPÍTULO .....	44
<b>3</b>	<b>DEFINIÇÃO DE CASOS DE TESTES</b> .....	45
3.1	TÉCNICAS DE SELEÇÃO DE CASOS DE TESTE .....	45
<b>3.1.1</b>	<b>Geração Randômica</b> .....	46
<b>3.1.2</b>	<b>Particionamento de Equivalências</b> .....	47
<b>3.1.3</b>	<b>Tabela de Decisões e Grafo de Causa e Efeito</b> .....	48
<b>3.1.4</b>	<b>CEG-BOR e CEG-BOR-MI</b> .....	49
3.2	ESTRATÉGIA PARA DEFINIÇÃO DE CASOS DE TESTE	53
<b>3.2.1</b>	<b>Exemplo</b> .....	54
3.3	CONSIDERAÇÕES FINAIS .....	56
<b>4</b>	<b>GERAÇÃO DE ORÁCULOS</b> .....	59
4.1	CRIAÇÃO DE MODELOS DE ORÁCULOS DA MCE ..	59
<b>4.1.1</b>	<b>Módulo de Diagnóstico</b> .....	61
<b>4.1.2</b>	<b>Módulo de Controle</b> .....	63
<b>4.1.3</b>	<b>Módulo Lógico</b> .....	63
<b>4.1.4</b>	<b>Módulo de Temporização</b> .....	64
<b>4.1.5</b>	<b>Módulo de Leitura de Entradas e Saídas e Trata- mento de Votação</b> .....	65
4.2	EXEMPLO I .....	66
4.3	EXEMPLO II .....	68
4.4	CONCLUSÕES FINAIS DO CAPÍTULO .....	69
<b>5</b>	<b>VALIDAÇÃO DO MÉTODO</b> .....	71
5.1	EDITOR DE MCE .....	71
5.2	FERRAMENTA AUTOMÁTICA DE TESTES .....	73

5.2.1	Entrada de Dados .....	74
5.2.2	Gerador de Casos de Teste .....	75
5.2.3	Geração de Observadores .....	75
5.2.4	Interação da Ferramenta com o SUT via OPC ....	75
5.2.5	Execução dos Observadores .....	76
5.2.6	Interface com o Usuário .....	76
5.2.7	Geração do Relatório .....	77
5.3	ESTUDO DE CASO E RESULTADOS .....	77
5.3.1	Exemplo didático .....	80
5.3.2	Aplicação Real e Resultados .....	82
5.4	CONCLUSÕES DO CAPÍTULO .....	86
6	CONCLUSÃO .....	87
6.1	LIMITAÇÕES E TRABALHOS FUTUROS .....	88
	REFERÊNCIAS .....	89



## 1 INTRODUÇÃO

A segurança operacional das instalações da Indústria de Petróleo e Gás conta com a existência de múltiplas camadas independentes de proteção. Essas são responsáveis pela prevenção de acidentes e no pior dos casos na mitigação desses. O Sistema Instrumentado de Segurança (SIS) configura a última barreira de prevenção de acidentes e tem o propósito de levar o processo à uma condição segura de operação mediante uma situação de risco. Caso o SIS seja incapaz de realizar uma determinada ação prevista, graves acidentes podem ser causados às pessoas, ao meio ambiente e aos equipamentos da própria planta (GRUHN; CHEDDIE, 2006).

O SIS é composto por um Controlador Lógico Programável (CLP) de segurança, responsável pelas funções de controle, e ainda por dispositivos de campo (representados por sensores e elementos finais) responsáveis pela identificação das condições da planta e atuação no processo. Erros na programação na lógica do CLP de segurança podem permanecer desconhecidos até a ocorrência de uma situação insegura quando o sistema é colocado à prova (SKOGDALEN; SMOGELI, 2011).

A norma IEC61511 (2003) apresenta práticas de engenharia apropriadas para o projeto do SIS na indústria de processos. A concepção de hardware e software para o SIS deve seguir metodologias rigorosas de desenvolvimento com o objetivo de verificar as especificações funcionais definidas pelas técnicas de análise de risco. Assim as normas e agências regulamentadoras defendem a execução do Teste de Conformidade o qual tem por objetivo avaliar a conformidade do dispositivo sob teste (*System Under Test* - SUT) .

O Teste de Conformidade utiliza uma abordagem caixa-preta, onde casos de teste são gerados a partir da especificação para que então esses sejam executados a fim de avaliar o valor das entradas e saídas do SUT. Os valores observados são então comparados com os valores esperados definidos pela especificação formal (NIDHRA, 2012; JORGENSEN, 2002; MYERS; THOMAS; SANDLER, 2011). Nessa abordagem a geração e execução do teste não baseia-se na estrutura interna da implementação, como é feito no teste caixa-branca.

A execução do Teste de Conformidade de CLP é realizada durante o Teste de Aceitação de Fábrica -TAF (*Factory Acceptancy Test* - FAT) e na prática segue um procedimento manual que explora um número superficial de condições e ainda exige agilidade pois a data de sua realização é muito próxima da data de entrega do dispositivo para

operação. Por conta disso, podem continuar desconhecidas possíveis inconformidades da implementação com a especificação de segurança. Há um crescente interesse da indústria por técnicas que melhorem as práticas de teste existentes e assim aumentar a confiabilidade do dispositivo testado. Nesse contexto, a automatização do procedimento de teste utilizando técnicas de teste de software pode ajudar a evitar alguns erros humanos, executar uma quantidade maior de casos de teste e ainda reduzir o seu custo.

Além da implementação de métodos de teste de software, métodos formais são adequados para síntese e verificação de programas de CLP de segurança (FREY; LITZ, 2000)(J. BELINFANTE, 1999). No entanto, a dificuldade de obter um modelo matemático correto de um programa de CLP e o risco de crescimento exponencial do espaço de estados de modelos formais, tornam as abordagens de teste caixa preta mais viáveis na prática da validação de aplicações complexas, como plataformas de petróleo.

Trabalhos recentes na literatura utilizam métodos formais na fundamentação de testes automáticos de CLP. Provost, Roussel e Faure (2011) propõe um teste de conformidade de CLP baseado em um modelo (*Model Based Testing* - MBT). Nesse trabalho a especificação, descrita por uma linguagem Grafset, é traduzida em um modelo de Máquina de Mealy. A partir de um algoritmo que analisa a máquina de estados, define-se uma sequência de teste que descreve comandos de entradas do dispositivo. Além disso, os valores esperados na saída do CLP acompanham os dados da sequência de testes, permitindo assim a comparação desses com os dados lidos na saída do CLP. A técnica limita-se na análise de funções com pequeno número de entradas/saídas, não pode ser aplicada em funções que exigem que a entrada fique ativa por um período especificado e ainda, pode existir um assincronismo durante a execução da sequência de teste.

Meinke e Sindhu (2013) propõe uma técnica caixa-preta conhecida como *Learning Based Testing* (LBT). Essa é capaz de gerar automaticamente casos de teste utilizando um algoritmo de aprendizagem e análise *model checking*. A ideia básica do algoritmo inicia-se com a criação de um modelo a partir dos dados observados nas entradas/saídas do SUT. Esse modelo é checado a partir de uma regra PLTL (*Propositional Linear Temporal Logic*) representando a especificação formal. A checagem gera um contra-exemplo que não atende a regra. Esse então é executado como um novo caso de teste e se a inconformidade for confirmada o teste é finalizado. O LBT ainda é uma técnica em estudo e exige que a especificação de entrada esteja repre-

sentada em PLTL. Porém é um método alternativo à técnica MBT e diminui a redundância de casos de teste a serem analisados.

Prati, Farines e Queiroz (2015) propõem um método de testes automático com o propósito de complementar o procedimento de testes de CLPs realizado na indústria de petróleo e gás. Esse método é capaz de criar automaticamente modelos que representem formalmente a especificação. Por sua vez esses modelos realizam o diagnóstico de falhas perigosas das funções de segurança durante a execução de uma sequência de comandos (casos de teste) na entrada do CLP. O método foi aplicado em um projeto de um forno industrial e limita-se às funções lógicas da aplicação estudada, não oferece o diagnóstico de alguns tipos de falhas e não emprega técnicas específicas na geração de casos de teste.

## 1.1 OBJETIVOS

O objetivo geral desse trabalho é propor um método para geração e execução de um teste de conformidade automatizado da lógica das funções de segurança de um Controlador Lógico Programável (CLP) que compõe um Sistema Instrumentado de Segurança (SIS). As contribuições desse trabalho envolvem o desenvolvimento, implementação e aplicação do método no sistema de uma plataforma offshore de produção de petróleo.

Pretende-se apresentar estratégias para geração de casos de teste e geração de oráculos. Essas técnicas devem assumir, como base de dados de entrada, a especificação lógica de segurança descrita pela Matriz de Causa e Efeito. Os oráculos devem representar, por meio de modelos formais de Redes de Petri, as funções lógicas de segurança e ainda avaliar o comportamento de execução da lógica de intertravamento do CLP.

A implementação do método deve ser realizada através de uma ferramenta computacional que permita uma interface com um testador humano. A ferramenta deve oferecer usabilidade para comunicação, execução e diagnóstico do teste do dispositivo.

A etapa de aplicação tem como objetivo a validação do método em um caso real seguindo as normas vigentes nas indústrias e obtenção de resultados do trabalho desenvolvido.

## 1.2 ESTRUTURA DO DOCUMENTO

A dissertação é composta por seis capítulos. O capítulo 2 apresenta uma visão geral do método proposto e uma fundamentação teórica dos conceitos básicos do SIS e de métodos para testes de CLP. A estratégia de geração dos casos de teste é detalhada no capítulo 3. Na sequência é mostrada a etapa de criação dos modelos de observadores de Redes de Petri os quais representam a especificação das funções lógicas de segurança e realizam o diagnóstico do comportamento do sistema. O desenvolvimento da ferramenta experimental e os resultados de sua aplicação em um caso real são detalhados no capítulo 5. Por fim, no capítulo 6, são discutidas as limitações, perspectivas de trabalhos futuros e conclusões do trabalho realizado.

## 2 PROPOSTA DE MÉTODO DE TESTES

Nesse capítulo é apresentado o processo de desenvolvimento do SIS na indústria e são discutidas as etapas de desenvolvimento da lógica do CLP do SIS em instalações marítimas da Petrobras, desde a preparação da especificação até a fase de testes. Ainda são explorados métodos de testes que trabalham com linguagem formal e utilizam uma abordagem caixa-preta a partir de estudos da literatura. Por fim, é apresentada uma visão geral do método sistemático de testes proposto nessa dissertação e que será explorado em detalhes nos capítulos subsequentes.

### 2.1 SISTEMA INSTRUMENTADO DE SEGURANÇA

As instalações da indústria de processo utilizam camadas de proteção para prevenir (camadas de prevenção) e mitigar (camadas de mitigação) falhas de operação. Estas camadas são definidas durante o projeto e buscam aumentar a segurança e o controle de riscos de acidentes catastróficos. As camadas de prevenção de riscos servem para reduzir a probabilidade de ocorrência de um evento perigoso que pode causar um acidente grave. Já as camadas de mitigação são responsáveis pelo controle ou mitigação das consequências do acidente não evitado pela camada de prevenção.

A Figura 1 apresenta através de um diagrama as camadas de proteção de uma planta convencional. O Sistema Instrumentado de Segurança (SIS) configura a última barreira de prevenção de acidentes e tem o propósito de levar o processo para uma condição segura de operação mediante a identificação de algum risco. O SIS é um sistema de controle composto por sensores, executores de lógica e elementos finais. Os executores da lógica são responsáveis pelas funções de controle do sistema. Estes são providos de uma tecnologia especificada no projeto que pode ser composta por um sistema pneumático, dispositivos a relé, CLPs convencionais, ou ainda por CLPs de segurança que possuem funções específicas de segurança.

Existe uma diferença entre o SIS e o BPCS (Sistema Básico de Controle de Processo) apresentando na Figura 1. O BPCS é um sistema ativo e é responsável pelas ações básicas do controle do processo. O BPCS, assim como o SIS, incorpora os elementos finais ou atuadores, o executor da lógica e ainda os sensores e medidores dos parâmetros

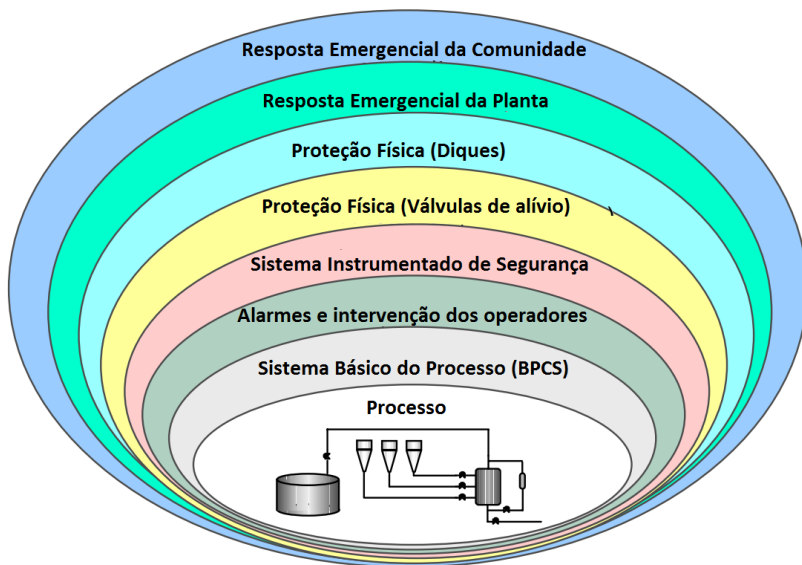


Figura 1 – Camadas de proteção.  
Adaptado de Gruhn e Cheddie (2006)

da planta. Esse ainda é responsável pelo nível de controle básico do processo, como por exemplo o controle de temperatura de um forno utilizando uma estratégia de controle PID. O SIS é um sistema que deve atuar somente na eminência de riscos, uma vez que esses não foram extinguidos anteriormente pelo BPCS, e independentemente da ação de operadores ou de qualquer intervenção humana.

## 2.2 DESENVOLVIMENTO DO PROJETO DO SIS

A implementação do SIS em grandes instalações, segue um processo de desenvolvimento sistemático e rigoroso visando fundamentalmente prevenir a ocorrência de falhas de operação. O projeto desses sistemas críticos mobiliza múltiplos times de engenharia e exige a organização de uma série de documentos padronizados. As etapas que configuram o desenvolvimento desse sistema são descritas pela norma IEC-61511.

O primeiro passo descrito pela norma trata do desenvolvimento

de uma análise de riscos que consiste na identificação de eventos perigosos e classificação desses de acordo com o grau de risco. O risco é uma função da frequência ou probabilidade de um evento, e a gravidade ou consequências do evento que podem afetar as pessoas, a produção, os bens de capital, o meio ambiente, a imagem da empresa, etc. A avaliação do risco pode ser qualitativa ou quantitativa.

A segunda etapa está associada a alocação das funções de segurança às camadas de proteção da planta. Para todas as funções de segurança deve ser determinado um nível de integridade de segurança, o (*Safety Integrity Level - SIL*). O SIL é calculado a partir da medida do desempenho necessário para controlar todos os riscos existentes no processo.

A próxima etapa refere-se à definição da lógica funcional das entradas e saídas das funções de segurança. Se um erro for cometido na definição, esse será propagado às próximas etapas do projeto que correspondem inicialmente à programação do dispositivo da lógica (CLP).

Na sequência são elaborados o projeto conceitual de engenharia e o projeto detalhado de engenharia. No projeto conceitual faz-se uma análise dos requisitos funcionais do projeto para que então sejam definidos os elementos e/ou equipamentos que irão compor o sistema. A escolha da tecnologia, quantidade e configurações dos componentes é definida durante essa etapa. O projeto detalhado de engenharia envolve as atividades de fabricação do sistema com base no projeto conceitual, e ainda, a elaboração da documentação do projeto. Alterações na lógica do equipamento ou da tecnologia escolhida são mais viáveis durante essa etapa do que após a instalação em campo.

Após o desenvolvimento do equipamento é realizado um Teste de Aceitação de Fábrica-TAF (*Factory Acceptancy Test - FAT*) e tem como finalidade a avaliação de conformidade da lógica do comportamento do dispositivo em relação à especificação. Caso alguma inconformidade seja encontrada durante os testes deve ser tomada uma ação para correção dessa. A entrega do equipamento e liberação para montagem na planta só se dá quando não existirem mais inconformidades.

A instalação, comissionamento e validação garantem que o sistema seja instalado e iniciado de acordo com o projeto e as especificações de segurança. Todo o sistema precisa ser checado, incluindo os instrumentos e equipamentos de campo, para que então seja dado o início da operação. Devem ser realizadas manutenções periódicas obedecendo todos os procedimentos de segurança.

Durante a etapa de operação do sistema podem existir dois tipos de falhas do SIS, são elas: a falha perigosa (*dangerous failure*) e a falha

segura (*safe failure*). Por exemplo, a primeira ocorre na situação em que um alarme, responsável por uma determinada ação do sistema de segurança, não é ativado corretamente quando necessário. Já a falha segura, ocorre na situação em que um determinado alarme é soado sem necessidade.

O primeiro tipo de falha resulta diretamente em graves acidentes e a segunda resulta em paradas na linha de produção. As paradas provocam perda de confiabilidade do sistema por parte dos operadores, os quais podem desabilitar partes do sistema a fim de forçar o reinício da operação. Dessa maneira, quando a falha de fato ocorre, o sistema ficará impossibilitado de executar as ações de segurança.

Muitas vezes as condições do processo mudam e o sistema de segurança precisa ser modificado. O impacto dessas alterações deve ser estudado e revisado por uma equipe especializada de engenharia. Além disso, as novas alterações devem ser testadas e documentadas. Por fim, ainda existe o procedimento de descomissionamento que visa garantir que a remoção ou substituição do sistema não impacte no processo e zele pela segurança dos operadores, equipamentos e meio ambiente.

### **2.2.1 Causas de acidentes do SIS**

Um levantamento de causas de acidentes apresentado por Gruhn e Cheddie (2006), indica que os acidentes podem ser provocadas por erros na especificação, erros de desenvolvimento do projeto e engenharia, erros de instalação e comissionamento, erros de operação e manutenção e erros provocados por mudanças após o comisionamento.

Na prática, o procedimento para identificação dos erros de desenvolvimento costuma ser realizado durante as etapas iniciais de concepção do sistema. Esse procedimento envolve a realização de testes visuais e funcionais, antes da entrega do dispositivo na fábrica. Essa é uma forma de evitar futuros problemas operacionais. O Controlador Lógico Programável (CLP), por exemplo, deve ser testado durante essa etapa.

Para garantir que a operação do sistema está de acordo com a especificação, um novo teste pode ser realizado após a partida ou comissionamento. Para evitar a parada da planta, um teste em laboratório pode ser realizado com a cópia da lógica do CLP e com toda a documentação técnica da especificação de segurança.



## 2.3 METODOLOGIA DE DESENVOLVIMENTO DA PETROBRAS

A metodologia de desenvolvimento da lógica de controle do CLP do Sistema Instrumentado de Segurança das instalações marítimas de produção da Petrobras segue um processo de desenvolvimento (Figura 2) definido por normas internas (N-1883, 2002) (N-2595, 2012). O *Piping and Instrumentation Diagram (P&ID)* é o primeiro documento criado pelos engenheiros do processo. Esse define a instrumentação (localização e descrição dos dispositivos de campo) e as malhas de controle. Serve como base para a definição da especificação.

A Matriz Causa e Efeito é um documento que especifica a lógica de segurança relacionando os eventos (causas) e ações (efeitos). A criação desse documento nos projetos de instalações marítimas de produção obedece a diretriz ET-3000.00-1200-800-PGT-006 (2000) tratada na Seção 2.3.1. As seqüências para início e desligamento da operação do BPCS são especificadas no Memorial Descritivo.

O próximo estágio de desenvolvimento é representado pela criação do Diagrama Lógico e pelo plano do FAT. O Diagrama Lógico descreve as lógicas de operação do sistema por meio de blocos e portas lógicas (conforme a norma ISA 5.2). As informações de uma mesma lógica devem estar contidas em uma única folha com o intuito de facilitar a compreensão. O propósito desse documento é auxiliar o programador no processo de criação do código do CLP.

O plano do TAF lista as funções de segurança que precisam ser testadas e serve como guia para a etapa de testes (TAF). Para facilitar a substituição de componentes ou alterações na lógica, caso necessário, o teste costuma ser realizado em laboratório próximo do local em que o dispositivo foi desenvolvido. A execução do teste é realizada de forma manual: o testador humano recebe o CLP programado juntamente com o plano de testes; prepara-se um ambiente de simulação para diagnóstico do comportamento do dispositivo desenvolvido onde as entradas do CLP são forçadas e as suas saídas são observadas conforme os eventos prescritos no plano do FAT. As inconsistências identificadas são então listadas em um relatório para que futuras alterações sejam realizadas. Quando não forem identificadas mais inconsistências, o dispositivo é então liberado para entrega e instalação na planta.

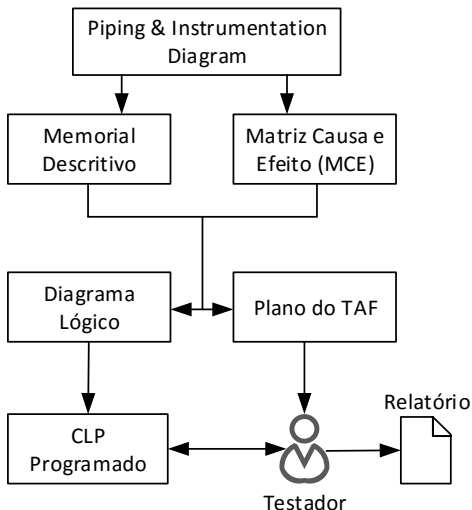


Figura 2 – Processo de desenvolvimento do SIS na Petrobras  
Adaptado de Prati (2014)

### 2.3.1 Diretrizes para elaboração da Matriz Causa e Efeito

A Matriz Causa e Efeito (MCE) é um recurso utilizado para especificar os intertravamentos do sistema de segurança. A estrutura das matrizes é formada por uma tabela que possui a mesma configuração para diversas aplicações, como mostra a Figura 3. Basicamente, o campo das causas pode listar os sensores da planta e o campo dos efeitos, os atuadores da planta. As interseções, formadas entre as linhas e colunas, correlacionam as causas aos efeitos e podem conter valores lógicos ou numéricos.

A representação da relação lógica deve obedecer uma simbologia padrão descrita por norma. A principal norma interna de referência para a elaboração das MCEs do SIS dos projetos de instalações marítimas de produção da Petrobras é a diretriz ET-3000.00-1200-800-PGT-006 (2000). Essa estabelece uma simbologia, regras de preenchimento e um *template* (Anexo A) visando definir um padrão para a formatação e documentação dessas informações. Basicamente o documento possui cinquenta linhas e cinquenta colunas de preenchimento e é formado pelas seguintes seções:

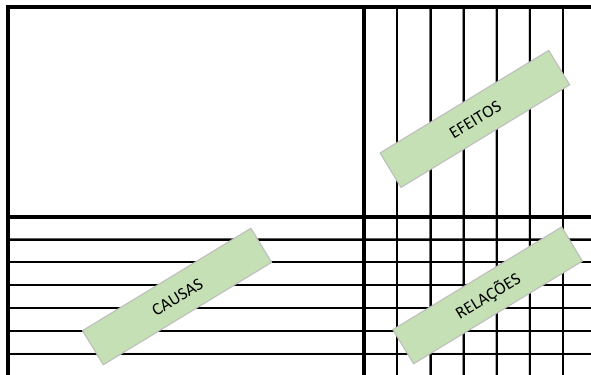


Figura 3 – Estrutura Básica de uma Matriz Causa e Efeito

- Legenda ou Simbologia: estabelece a simbologia utilizada para o preenchimento do documento;
- Causas: Caracteriza os sinais referentes aos eventos causadores de uma ação;
- Efeitos: Caracteriza os sinais referentes as ações a serem tomadas frente à ocorrência de um evento;
- Relações (Interseções): Contém a lógica que relaciona cada causa ao(s) efeito(s) associados;
- Notas: Referencia cada causa ou intersecção a uma nota presente na área de “Notas gerais”;
- Notas gerais: Possui a descrição detalhada de cada nota inserida na matriz;
- Documentos referenciados: Contém uma lista de documentos que estão associados à essa matriz. Podem ser mencionados os fluxogramas dos dispositivos citados na matriz ou ainda outras matrizes com causas e efeitos correlacionados;
- Rótulo: Serve para identificar o subsistema que a matriz pertence; o responsável técnico, data de criação e número da revisão do documento.

As operações lógicas entre as causas e os efeitos são estabelecidas pela simbologia da Figura 4. Essa simbologia pode representar operações do tipo “AND”, “OR”, “NOR” ou ainda operações que levam em conta o tempo. A simbologia não compreende operações de maior complexidade e que envolvam sequenciamento de acionamento. Essas últimas situações devem ser descritas por notas laterais ou no diagrama lógico.



FUNÇÃO	SÍMBOLO	OBSERVAÇÃO
Lógica OU		
Lógica NOR		OU negado
Lógica E	A Ou Ax	X é numero inteiro a ser usado quando necessário para distinguir conjuntos de lógicas E entre si quando os mesmos estão em uma única coluna.
Lógica Temporizada	Tx	X é o tempo em segundos escolhido para a lógica.

Figura 4 – Simbologia definida pela ET-3000.00-1200-800-PGT-006 (2000)

No SIS é comum encontrar redundância nos elementos de entrada do sistema. Pode ainda existir votação entre os elementos de um grupo específico, como por exemplo sensores de gás da zona A. O caso da votação dos sinais de entrada pode ser representado em uma das colunas da MCE. A Figura 5 apresenta um exemplo de MCE simplificada que utiliza a simbologia das funções apresentadas. Esse exemplo deve ser interpretado da seguinte forma:

- **ESD-101** deve ser acionado quando **HSS-101100** ou **YST-101200** está ativo;
- **FD-101** deve ser acionado quando **YST-101200** ou **HSS-101100**

EXEMPLO DE MCE			EFEITO			
			Equipamento	TAG	ESD-101	FD-101
CAUSA			Emerg. Alarm	Fire Detected	Fire Confirmed	
Equipamento	Votação	TAG				
Manual Fire Alarm		HSS-101100	X	A1		
Smoke Fire Detector		YST-101200	X	X	T10	
Flame Fire Detector	1oo3	UST-101001		A1		
Flame Fire Detector	2oo3	UST-101002			X	
Flame Fire Detector		UST-101003				

Figura 5 – Matriz Causa e Efeito Exemplo

está ativo e ao menos uma das entradas do grupo **UST-101001**, **UST-101002**, **UST-101003** está ativa;

- **FC-102** deve ser acionado quando **YST-101200** permanece ativo por 10 segundos ou ao menos duas das entradas do grupo copreendido por **UST-101001**, **UST-101002**, **UST-101003** estão ativas;

Nota-se no exemplo que as entradas **UST-101001**, **UST-101002**, **UST-101003** fazem parte de um grupo de votação. Esse grupo possui dois tipos de votação: 1oo3 (*one out of three*) e 2oo3 (*two out of three*). O primeiro tipo de votação define que o efeito deve ser ativado por ao menos uma das três entradas do grupo; já o segundo tipo define que o efeito deve ser ativado por pelo menos duas das três entradas do grupo. Uma atenção deve ser tomada na interpretação dessa função de segurança. O símbolo presente na interseção da linha da votação não refere-se exclusivamente a lógica da causa dessa linha, mas ao grupo de entradas dessa votação.

## 2.4 MÉTODOS PARA TESTE DE PROGRAMAS DE CLP

No Teste de Conformidade de CLPs o dispositivo sob teste é tratado como uma caixa-preta, não existindo acesso ao código implementado. A Figura 6 apresenta uma visão geral do Teste de Conformidade. O princípio do teste parte da definição de casos de teste com base na especificação. Essa definição pode ser realizada a partir da experiência do testador ou de algoritmos para geração de casos de teste. Então é realizado o acionamento das entradas do dispositivo com base na seleção criada. Os valores dos sinais das saídas são observados e comparados com o valor esperado. Assim é feito um veredito diagnosticando conformidade ou desconformidade do dispositivo (NIDHRA, 2012; JORGENSEN, 2002; PROVOST; ROUSSEL; FAURE, 2011) com determinada função de segurança.

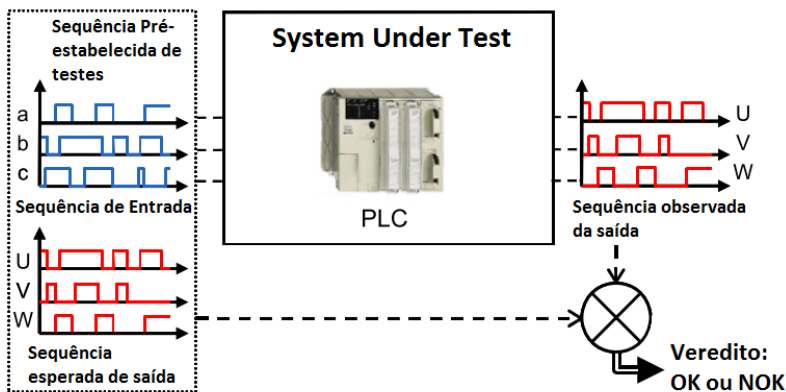


Figura 6 – Visão Básica do Teste de Conformidade.

Adaptado de Provost, Roussel e Faure (2011)

Na prática esse teste é executado durante o *Factory Acceptancy Test (FAT)* e segue um procedimento manual. Costuma exigir tempo e agilidade de uma equipe especializada formada por técnicos, engenheiros e operadores, visto que é a última fase de desenvolvimento antes da entrega do produto. O método manual explora um número superficial de situações (casos de teste). Por conta disso, alguns erros podem ser negligenciados permitindo a ocorrência de futuras falhas durante a operação. Há um crescente interesse da indústria por técnicas que melhorem as práticas já existentes (PROVOST; ROUSSEL; FAURE, 2014).

Nesse contexto, a automatização do procedimento de testes pode melhorar o diagnóstico de falhas causadas por erros de desenvolvimento e otimizar o tempo de criação/execução do teste. A utilização de estratégias de testes de software pode auxiliar na concepção de um conjunto que abrange grande parte dos testes de funcionalidades e que evita casos redundantes.

Além da implementação dessas estratégias, métodos formais são adequados para síntese e verificação de programas de CLP de segurança (FREY; LITZ, 2000). No entanto, a dificuldade de obter um modelo matemático correto de um programa de CLP e o risco de crescimento exponencial do espaço de estado de modelos formais tornam as técnicas de caixa preta abordagens preferidas para validação de aplicações complexas, como plataformas de petróleo.

Na literatura são utilizados métodos formais na fundamentação de testes automáticos de CLP. Provost, Roussel e Faure (2011) propõe um teste de conformidade de CLP utilizando uma técnica baseada em modelo conhecida como *Model-Based Testing* (MBT). A técnica MBT é definida como uma automatização do procedimento de testes caixa-preta (*the automation of the design of black-box tests*) (UTTING; LEGEARD, 2006). Essa técnica constrói um modelo relativo ao comportamento esperado da implementação que respeita a especificação. A partir desse modelo é possível gerar casos de teste para serem executados no dispositivo (UTTING; LEGEARD, 2006). O formalismo dos modelos pode ser definido por: FSM (*Finite State Machine*) (PROVOST; ROUSSEL; FAURE, 2014), autômatos temporizados (PEIXOTO, 2010)(DIAS; PERKUSICH, 2008)(OLIVEIRA, 2009), ou outros. Os principais passos da técnica são os seguintes: criação do modelo, geração dos casos de teste, execução dos casos de teste e análise dos resultados (UTTING; LEGEARD, 2006). Além da identificação de erros na implementação, problemas nos requisitos acabam ficando explícitos com a concepção de um modelo.

Em Provost, Roussel e Faure (2011) o comportamento esperado da implementação é concebido por um modelo equivalente de Máquina de Mealy. Esse modelo é preparado com base em uma especificação Grafset e através de um algoritmo é definida uma sequência de casos de teste. Esse algoritmo faz uma checagem das transições dos estados da máquina de Mealy e define uma sequência capaz de percorrer todos os estados do modelo. Em Lee e Yannakakis (1996) são reunidos e discutidos diversos outros algoritmos existentes para a construção de sequências de casos a partir de máquinas de estado. A partir da sequência executa-se então o teste com o CLP. A desvantagem da uti-

lização desse método está relacionada com a complexidade dos modelos quando um grande número de entradas/saídas é analisado e esse não trata de funções temporizadas. Ainda pode sofrer um assincronismo entre a sequência estabelecida através do modelo e o próprio comportamento do sistema que ocorre quando mais de uma entrada é comandada ao mesmo tempo.

Meinke e Sindhu (2013) propõe uma outra abordagem caixa-preta, alternativa ao método MBT, conhecida como *Learning Based Testing* (LBT). A ideia básica do método é gerar um modelo representativo do comportamento do dispositivo a partir dos dados observados nas entradas/saídas do dispositivo sob teste. Então esse modelo é checado a partir de uma regra PLTL que representa a especificação formal. Durante a checagem gera-se um contra-exemplo que será executado como um novo caso de teste. A execução desse novo caso gera um veredito com dois possíveis resultados: um que aponta uma inconformidade e outro uma conformidade do comportamento do dispositivo com a especificação. Na ocorrência do primeiro resultado, o teste é finalizado indicando a falha encontrada. Já no segundo, o modelo obtido é então revisto e uma nova iteração é realizada a fim de buscar um novo contra-exemplo. O LBT ainda é uma técnica em estudo e exige que a especificação de entrada esteja representada em PLTL.

Oliveira (2009) apresenta outro método para o teste de conformidade entre o programa do CLP e a especificação. A Figura 7 apresenta uma visão geral do método descrito. Esse consiste em gerar automaticamente modelos de autômatos temporizados seguindo uma sintaxe e semântica da ferramenta Uppaal. Os modelos são preparados a partir da especificação dada por Diagramas de Lógica Binária (ISA 5.2) e do código Ladder da implementação. Após a geração destes modelos, testes de conformidade são realizados usando a ferramenta de teste Uppaal-TRON que confronta os dois modelos e faz um diagnóstico de falhas.

A técnica apresentada por Oliveira (2009) exige acesso à lógica do CLP para a criação de um modelo correspondente à implementação. A geração automática do modelo é inviável em casos em que a implementação envolve um número grande de entradas e saídas. Além disso o método para geração de casos de teste e diagnóstico de falhas é dependente das ferramentas Uppal e Uppal-TRON.



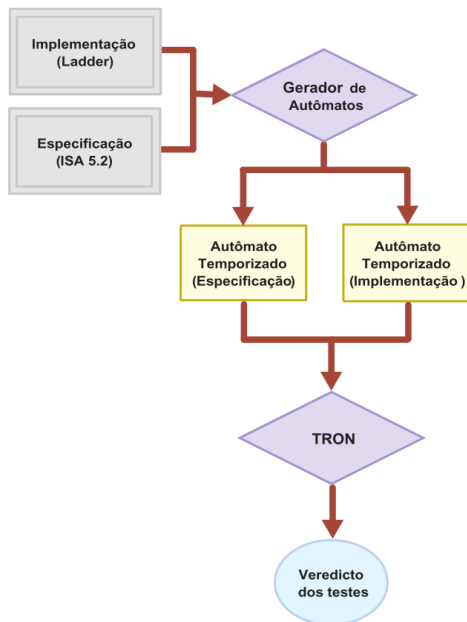


Figura 7 – Teste baseado em modelo de Autômatos temporizados.  
 Fonte:Oliveira (2009)

## 2.5 PROPOSIÇÃO DE MÉTODO AUTOMÁTICO DE TESTES

Como mencionado anteriormente, o método atual de preparação e execução de testes da lógica do CLP do SIS nas instalações marítimas da Petrobras segue um procedimento manual. A proposta desse trabalho é propor um método automático de geração e execução de um teste funcional complementar, visando melhorar o diagnóstico de falhas de execução da lógica do CLP.

O método proposto parte de uma abordagem caixa-preta na qual a geração do teste é baseada no documento que representa a especificação. A estratégia não é baseada na estrutura lógica da implementação como em outras abordagens. O método automático de testes integra-se na etapa do TAF da metodologia de desenvolvimento do SIS, como mostra a Figura 8.

Fundamentalmente o método de testes é caracterizado por duas etapas principais: a geração e a execução do teste. A Figura 9 apresenta

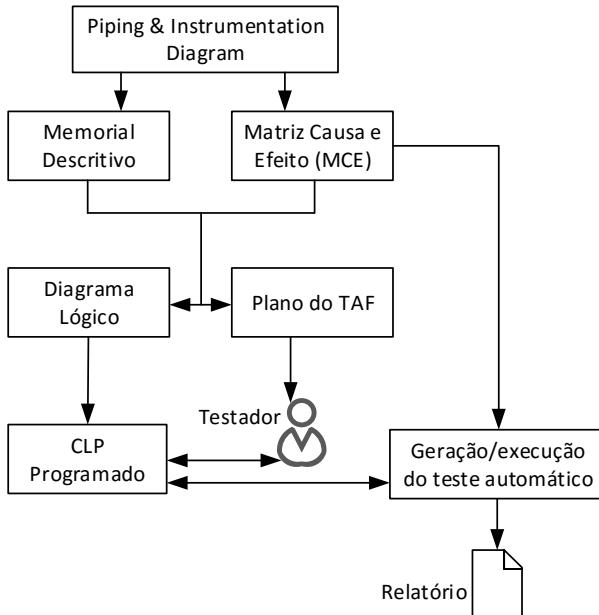


Figura 8 – Procedimento para teste automático da lógica do CLP

uma visão detalhada do método considerando as diversas etapas.

A MCE é um documento que representa de forma organizada as funções de segurança da lógica do CLP. A geração do teste automático é baseado nesse documento. Durante essa etapa, os casos de testes são definidos e organizados em uma lista que define os valores para as entradas do dispositivo sob teste. Essa lista deve compreender condições que considerem a análise de todas as funções de segurança.

Além disso, modelos de oráculos são gerados para diagnosticar falhas oferecendo um veredito de conformidade ou não conformidade para cada caso de teste executado. Os modelos de oráculos possuem duas funções básicas: expressar as funções específicas de segurança da lógica do SIS, e ainda, diagnosticar a existência de falhas seguras e perigosas decorrentes de um problema no comportamento da lógica do CLP. Os modelos gerados possuem um módulo específico para o diagnóstico de falhas seguras e perigosas do SIS. Esse módulo é capaz de analisar os sinais de entrada e saída do CLP e compará-los aos valores esperados definidos pela especificação lógica de segurança descrita pela MCE.

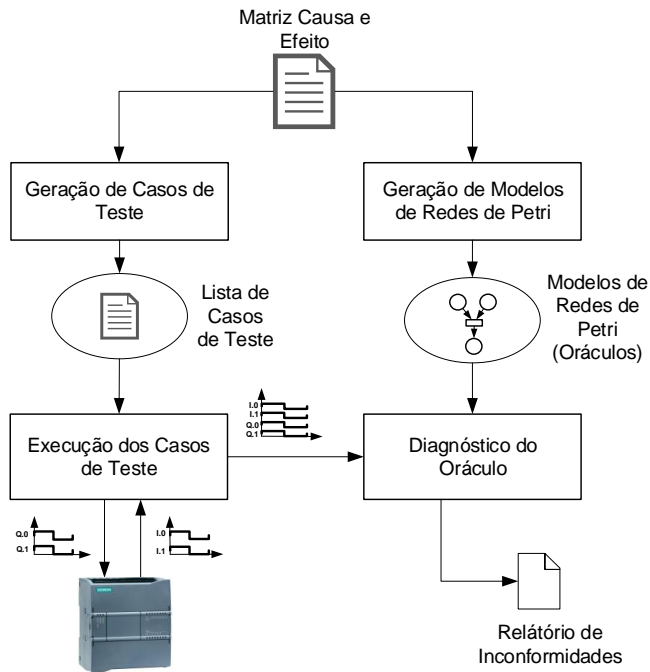


Figura 9 – Visão Detalhada do método

A Rede de Petri é uma ferramenta gráfica e matemática que favorece a representação de funções que envolvem eventos e ações de um sistema (BERTHOMIEU; DIAZ, 1991). Essa linguagem é utilizada para modelagem e execução de oráculos, a fim de garantir consistência e facilitar a automação da ferramenta de teste. Além das vantagens do desenvolvimento de software baseado em modelo, o uso de um modelo formal permite o uso de operações matemáticas para a composição e análise de oráculos (SELIC, 2012) (MURATA, 1989). Além disso, a linguagem é adequada para representar múltiplas especificações cronometradas de entradas concorrentes do PLC (VEIGA et al., 2017a).

Na execução do teste, as entradas do sistema sob teste (CLP) são ativadas com base na lista de casos de teste. Ainda, o comportamento das entradas/saídas é avaliado a fim de verificar inconformidades do sistema com as especificações impostas no documento da Matriz Causa e Efeito.

A partir dos resultados obtidos pela execução dos modelos, é

criado um relatório identificando as inconformidades encontradas entre o comportamento observado do sistema e a especificação.

O método proposto nessa dissertação tomou como ponto de partida o trabalho apresentado em Prati (2014) que também analisa funções de segurança de uma MCE através de modelos de Redes de Petri. Dentre as principais contribuições em relação à esse trabalho, destaca-se, o desenvolvimento da técnica de geração de casos de casos, a proposição de modelos de oráculos que permitem o diagnóstico de falhas seguras e perigosas (sinais falsos positivos e sinais falsos negativos) e ainda a validação do método em um caso real da indústria.

## 2.6 CONSIDERAÇÕES FINAIS DO CAPÍTULO

O procedimento de testes do dispositivo lógico do SIS aplicado hoje nas indústrias explora um número de condições (casos de teste) que não é suficiente para identificar alguns tipos de falhas. Além disso existe a necessidade de empregar uma equipe de técnicos especializada por um período de tempo significativo. Durante a operação do SIS essas falhas podem causar um grave acidente. Dessa forma, destaca-se a importância do desenvolvimento de um método automatizado de testes do dispositivo da lógica do SIS. Esse método, por sua vez, utiliza um documento, conhecido como Matriz Causa e Efeito (MCE), que serve para derivar casos de testes e modelos de oráculos. Então, durante a execução do teste, realiza-se um diagnóstico de inconformidades.

Os dois capítulos seguintes exploram em detalhes as etapas do método proposto nesse capítulo. Serão discutidas as estratégias para definição de casos de teste e a criação de modelos de oráculos para diagnóstico de inconformidades da lógica de segurança.

### 3 DEFINIÇÃO DE CASOS DE TESTES

Nesse capítulo serão apresentadas algumas técnicas de geração de casos de teste já conhecidas na literatura e será definida uma estratégia de seleção de casos de teste que busca abranger situações em que as falhas da implementação do SIS podem ser identificadas. O método proposto nesse trabalho estabelece uma estratégia que utiliza o documento da especificação MCE como base de entrada para o algoritmo de geração de casos de teste.

#### 3.1 TÉCNICAS DE SELEÇÃO DE CASOS DE TESTE

Para verificar o comportamento de um dispositivo é necessário preparar um teste efetivo que execute determinados casos de teste. Um caso de teste é formado por um conjunto de entradas do dispositivo sob teste e no qual um valor é definido para cada uma dessas entradas. O caso de teste ainda pode conter uma lista de valores esperados para as saídas do dispositivo (MYERS; THOMAS; SANDLER, 2011) (JORGENSEN, 2002).

O conjunto de entradas pode assumir diferentes valores no qual o número de combinações cresce exponencialmente em relação ao número de entradas do dispositivo, inviabilizando a execução de um teste exaustivo que considera todas as combinações possíveis. Em vista desse problema, algumas estratégias para seleção de casos de testes foram propostas.

Cada estratégia é definida por algumas regras que têm por objetivo estabelecer uma quantidade suficiente de casos de teste que seja capaz de: testar as funcionalidades do dispositivo e identificar possíveis inconformidades com a especificação. Além disso, é recomendável que a preparação de um teste utilize mais de uma dessas estratégias ou que ainda utilize uma combinação dessas (HOWDEN, 1980) (MYERS; THOMAS; SANDLER, 2011).

Existem duas abordagens fundamentais na seleção de casos de teste de sistemas de controle. A primeira delas é conhecida como *functional testing* ou teste caixa-preta. Essa abordagem possui uma visão “externa” do SUT e a ideia básica é verificar o comportamento do dispositivo forçando sinais nas entradas e observando os sinais de saída. Nesse teste a única fonte de informação para definição de casos de teste, que servirão de base para a geração de sinais de entrada, é a especi-

ficação do sistema (JORGENSEN, 2002). A outra abordagem refere-se ao *structural testing*, ou teste caixa-branca, que em contraste à abordagem anterior possui conhecimento da implementação, ou seja, possui acesso a estrutura do código e utiliza essas informações para definição dos casos de teste.

Segundo Jorgensen (2002) os métodos de seleção caixa-preta cobrem maior parte do conjunto de casos de teste referentes a especificação. Isso se dá justamente porque a geração desses casos é baseada na informação da própria especificação. Já na abordagem caixa-branca ocorre o contrário, a cobertura da seleção de casos de teste compreende maior parte do conjunto de casos que representam o comportamento da implementação. Uma boa seleção de casos de teste pode ser preparada utilizando métodos de definição caixa-preta e então suplementada por técnicas caixa-branca que analisam a estrutura do código (MYERS; THOMAS; SANDLER, 2011).

Nas subseções seguintes serão tratadas estratégias bem estabelecidas para a definição de casos de teste que utilizam uma abordagem caixa-preta, são elas: teste randômico, particionamento de equivalência (*equivalence partitioning*), teste baseado por tabela de decisões (*decision table-based testing*) e o grafo de causa e efeito baseado em operações booleanas (CEG-BOR).

### 3.1.1 Geração Randômica

O método de geração randômica de casos de teste não possui uma natureza sistemática. O número de entradas que serão analisadas é a única informação relevante para a estratégia. Duas principais vantagens estão relacionadas à utilização dessa abordagem: baixo custo computacional na criação de um grande número de casos de teste e independência estatística entre os pontos de teste a serem analisados (SINDHU, ) (DURAN; NTAPOS, 1984).

Caso um testador humano experiente estivesse preparando casos manualmente, ele selecionaria novos casos utilizando uma heurística que verificasse as funcionalidades ainda não exploradas. A estratégia de seleção de casos aleatórios não aproveita informações de casos antecessores já calculados. Alguns autores citam ainda uma estratégia conhecida como anti-randômica que permite a geração de novos casos aproveitando as informações dos casos que já foram testados (WU; MALAIYA; JAYASUMANA, 1998).

### 3.1.2 Particionamento de Equivalências

A estratégia de particionamento de equivalências parte do princípio da divisão do domínio de entradas e saídas da especificação em um finito número de classes de equivalência. A partir disso são definidos casos de teste da combinação dos valores das entradas de cada um dos conjuntos (MYERS; THOMAS; SANDLER, 2011; JORGENSEN, 2002).

Normalmente a especificação abrange uma lógica com muitas entradas. A estratégia de particionamento de equivalências evita a necessidade de um teste exaustivo que leva em conta a combinação de valores entre todas as entradas do sistema sob teste. Por exemplo, para uma classe com duas entradas, são gerados quatro casos ( $2^2$ ); para três entradas, são gerados oito ( $2^3$ ) e assim por diante. Um outro exemplo, se um sistema de 10 entradas for particionado em duas classes de equivalência, referentes a dois grupos desacoplados de cinco entradas, poderão ser realizados 64 ( $(2^5) + (2^5)$ ) casos de teste para abranger todas as combinações de cada classe isoladamente, ao invés de se testar todas as 1024 combinações de entradas do sistema.

Uma heurística pode ser estabelecida para determinação das entradas de cada classe. Por exemplo, no teste funcional, cada classe deve conter somente as entradas relacionados a uma determinada lógica de intertravamento. Assim serão geradas todas as combinações possíveis de valores de entradas que ativam uma determinada função de segurança. Essa estratégia pode ser aplicada a partir da Matriz Causa e Efeito que define claramente as relações entre as entradas (causas) e as saídas (efeitos). Porém, o emprego destas heurísticas pode deixar de fora da seleção alguns casos que verificam a ocorrência sinais de falso positivo (*safe failures*).

Ainda, na estratégia de particionamento de equivalências algumas falhas pode mascarar algumas falhas durante o diagnóstico. Isso pode ocorrer, por exemplo, quando um efeito está relacionado por uma lógica OU entre duas causas e uma delas possui uma falha real. Se o caso de teste definido estabelece o acionamento das duas causas, o efeito será ativado e a falha de uma das causas não será identificada.

Além das dificuldades já citadas, o particionamento de equivalências constrói casos redundantes, o qual limitam a cobertura de identificação de falhas. (MYERS; THOMAS; SANDLER, 2011; JORGENSEN, 2002) recomendam a utilização de outras técnicas que complementem a definição dos casos de teste.

### 3.1.3 Tabela de Decisões e Grafo de Causa e Efeito

A estratégia de particionamento de equivalências pode produzir um número grande de casos de teste dos quais boa parte são redundantes e não revelam a entrada específica causadora de uma falha. Para minimizar esse problema existe uma técnica conhecida como *Cause-Effect Graphing* ou ainda Tabela de Decisões (ELMENDORF, 1970). Essa técnica possibilita a determinação de um número mínimo de casos de teste capaz de testar todas as funcionalidades descritas na especificação (ADRION; BRANSTAD; CHERNIAVSKY, 1982) (SRIVASTAVA; PATEL; CHATROLA, 2009) (MALEKZADEH; AINON, 2010; PARADKAR; TAI; VOUK, 1997).

O grafo causa e efeito é representado por uma linguagem formal, semelhante a um circuito lógico digital (uma rede de combinação lógica), mas que utiliza uma representação mais simples. Diversos tipos de lógica podem ser representados utilizando o grafo de causa e efeito. A Figura 10 apresenta dois grafos com as lógicas “OR” e “AND”, respectivamente. Cada relação entre causa (ou entrada) e efeito (saída) da especificação pode ser expressa pelo grafo e cada condição pode assumir um valor verdadeiro ou falso (MOGYORODI; MATH, 2010).

A partir dos grafos monta-se uma Tabela de Decisões especificando os diferentes casos de teste existentes (MYERS; THOMAS; SANDLER, 2011) (JORGENSEN, 2002). A Tabela 1 representa a Tabela de Decisões com os casos de teste dos grafos da Figura 10. As interseções

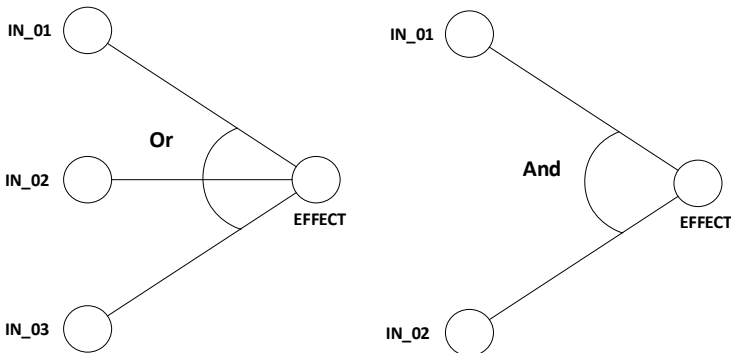


Figura 10 – Grafo de Causa e Efeito.  
Fonte: (MYERS; THOMAS; SANDLER, 2011)



	OR								AND			
	TEST 01	TEST 02	TEST 03	TEST 04	TEST 05	TEST 06	TEST 07	TEST 08	TEST 01	TEST 02	TEST 03	TEST 04
IN-01	F	T	F	F	T	F	T	T	F	T	F	T
IN-02	F	F	T	F	T	T	F	T	F	F	T	T
IN-03	F	F	F	T	F	T	T	T	-	-	-	-
Effect	F	T	T	T	T	T	T	T	F	F	F	T

Tabela 1 – Tabela de decisões do grafo da Figura 10

das tabelas representam os valores de cada causa ou efeito e esses podem assumir valor verdadeiro (T) ou falso (F).

Elmendorf (1970) e Myers, Thomas e Sandler (2011) desenvolveram um algoritmo eficiente para preparação de uma seleção de casos de teste capaz de revelar a causa de falhas de operação da implementação. Esse algoritmo exclui casos de teste redundantes ou condições que mascaram a identificação de alguma falha. No exemplo da Tabela 1, para a condição “OR”, apenas os casos Test#02, Test#03, Test#04 são selecionados para se testar uma falha perigosa, visto que esses casos são capazes de revelar a entrada específica que não ativa um efeito no caso de uma falha. Já na condição “AND”, somente o caso Test#04 é interessante, visto que nas outras condições a funcionalidade de ativação do efeito não é verificada.

### 3.1.4 CEG-BOR e CEG-BOR-MI

Paradkar, Tai e Vouk (1997) justificam que o número de casos de teste definidos pelo algoritmo *Cause-Effect Graphing* cresce exponencialmente em relação ao número de nós do grafo causa e efeito ou operadores lógicos de uma mesma função de segurança. Paradkar, Tai e Vouk (1997) propõe uma estratégia conhecida como CEG-BOR (*Cause-Effect Graphing - Boolean Operator*) que gera um conjunto de casos de teste onde o seu tamanho cresce linearmente com o número de operações lógicas de uma determinada função de segurança.

Será utilizado o grafo da Figura 11 como exemplo para ilustrar a aplicação da estratégia CEG-BOR. A expressão booleana que representa o grafo é dada por  $E = (a*b) + c$ , onde a,b,c são variáveis de entrada

e podem assumir os valores: (t) quando o valor da variável é Verdade e (f) quando o valor da variável é Falso.

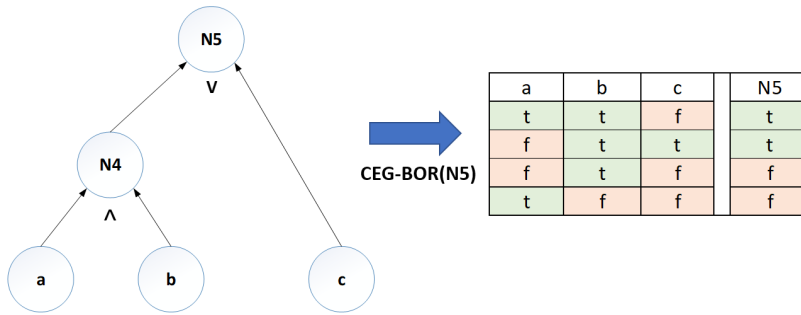


Figura 11 – Aplicação do método CEG BOR - Grafo de Causa e Efeito (N5 = (a\*b) + c)

O algoritmo recursivo CEG-BOR, proposto em Paradkar, Tai e Vouk (1997), obedece 4 regras básicas para geração do conjunto de casos de teste. Essas regras bem como as considerações são apresentadas a seguir:

Assumindo  $C_{\varepsilon}GBOR(N1)$  e  $C_{\varepsilon}GBOR(N2)$  como sendo os conjuntos de casos de teste gerados pela estratégia CEG-BOR para os nós  $N1$  e  $N2$ , respectivamente. Ainda, considerando  $C_{\varepsilon}GBOR(N)$  como sendo o conjunto de casos de teste gerado a partir da estratégia CEG-BOR para a variável  $N$ , então:

1. Se  $N$  é uma causa,  $C_{\varepsilon}GBOR(N)$  é dado por  $\{(t),(f)\}$

2. Se  $N = N_1.N_2$ ,  $C_{\varepsilon}GBOR(N)$  é obtido por:

$$C_{\varepsilon}GBOR_T(N) = C_{\varepsilon}GBOR_T(N1) \otimes C_{\varepsilon}GBOR_T(N2)$$

$$C_{\varepsilon}GBOR_F(N) = (C_{\varepsilon}GBOR_F(N1) \times \{t_{N2}\}) \cup (\{t_{N1}\} \times C_{\varepsilon}GBOR_F(N2))$$

aonde:

$$t_{N1} \in C_{\varepsilon}GBOR_T(N1)$$

$$t_{N2} \in C_{\varepsilon}GBOR_T(N2)$$

$$(t_{N1}, t_{N2}) \in C_{\varepsilon}GBOR_T(N)$$

3. Se  $N = N_1 + N_2$ ,  $C_{\varepsilon}GBOR(N)$  é obtido por:

$$C_{\varepsilon}GBOR_F(N) = C_{\varepsilon}GBOR_F(N1) \otimes C_{\varepsilon}GBOR_F(N2)$$

$$C_{\varepsilon}GBOR_T(N) = (C_{\varepsilon}GBOR_T(N1) \times \{f_{N2}\}) \cup (\{f_{N1}\} \times C_{\varepsilon}GBOR_T(N2))$$

aonde:

$$\begin{aligned} f_{N1} &\in C\varepsilon GBOR_F(N1) \\ f_{N2} &\in C\varepsilon GBOR_F(N2) \\ (f_{N1}, f_{N2}) &\in C\varepsilon GBOR_F(N) \end{aligned}$$

4. Se  $N = \overline{N1}$ ,  $C\varepsilon GBOR(N)$  é obtido por:

$$C\varepsilon GBOR_F(N) = C\varepsilon GBOR_T(N1) C\varepsilon GBOR_T(N) = C\varepsilon GBOR_F(N1)$$

Considerando agora a expressão booleada do exemplo do grafo da Figura 10, deseja-se obter um conjunto de casos de teste utilizando o algoritmo CEG BOR. O cálculo para obtenção dos casos é apresentado a seguir:

$$\begin{aligned} N4 &= ab; \\ N5 &= N4 + c; \end{aligned}$$

Queremos  $C\varepsilon GBOR(N5)$ , pela Regra 3:

$$(1) C\varepsilon GBOR_F(N5) = C\varepsilon GBOR_F(N4) \otimes C\varepsilon GBOR_F(c)$$

$$(2) C\varepsilon GBOR_T(N5) = (C\varepsilon GBOR_T(N4) \times \{f_c\}) \cup (\{f_{N4}\} \times C\varepsilon GBOR_T(c))$$

Obs:  $\otimes$  - Operação de Exclusão Mútua

Calculando  $C\varepsilon GBOR_F(N4)$  e  $C\varepsilon GBOR_T(N4)$ , pela Regra 2;

$$(3) C\varepsilon GBOR_T(N4) = C\varepsilon GBOR_T(a) \otimes C\varepsilon GBOR_T(b)$$

$$(4) C\varepsilon GBOR_F(N4) = (C\varepsilon GBOR_F(a) \times \{t_b\}) \cup (\{t_a\} \times C\varepsilon GBOR_F(b))$$

Pela Regra 1:

$$\begin{aligned} C\varepsilon GBOR_T(a) &= \{t\}; C\varepsilon GBOR_F(a) = \{f\} \\ C\varepsilon GBOR_T(b) &= \{t\}; C\varepsilon GBOR_F(b) = \{f\} \\ C\varepsilon GBOR_T(c) &= \{t\}; C\varepsilon GBOR_F(c) = \{f\} \end{aligned}$$

Substituindo em 3 e 4

$$\begin{aligned} C\varepsilon GBOR_T(N4) &= \{(t, t)\} \\ C\varepsilon GBOR_F(N4) &= \{(f, t); (t, f)\} \end{aligned}$$

Substituindo em 1 e 2

$$\begin{aligned} C\varepsilon GBOR_F(N5) &= \{(f, t, f); (t, f, f)\} \\ C\varepsilon GBOR_T(N5) &= \{(t, t, f); (f, t, t)\} \end{aligned}$$

Logo:

$$C_{\varepsilon}GBOR(N5) = C_{\varepsilon}GBOR_F(N5) \cup C_{\varepsilon}GBOR_T(N5)$$

$$C_{\varepsilon}GBOR(N5) = \{(f, t, f); (t, f, f); (t, t, f); (f, t, t)\}$$

O método CEG-BOR é limitado para expressões booleanas singulares, ou seja, quando cada variável booleana ocorre apenas uma única vez na expressão lógica. Assim esse método não abrange funções lógicas como a votação de um grupo de sensores presente nas lógicas de intertravamento do SIS. Paradkar e Tai (1995) propõe ainda uma estratégia complementar ao método CEG-BOR. Essa estratégia é conhecida como MI (Meaning Impact) e trabalha com as não singularidades das expressões booleanas e podendo cobrir assim casos de votação de entradas. O grafo da Figura 12 representa uma lógica de votação de um grupo de entradas do tipo 2oo3 ( $ab + ac + bc$ ), onde o método CEG-BOR-MI é o mais adequado para aplicação.

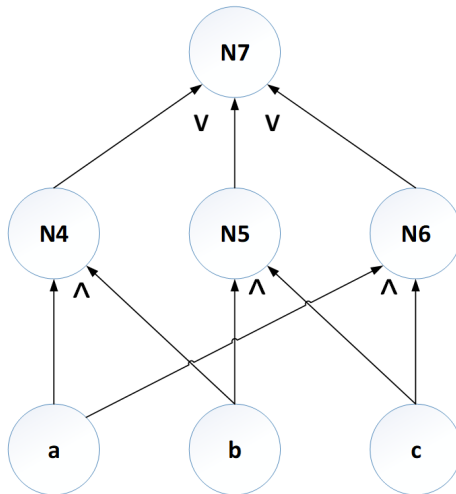


Figura 12 – Grafo de Causa e Efeito que representa uma votação de um grupo de entradas (2oo3)

O exemplo a seguir ilustra a aplicação da estratégia MI:

Considerando a expressão booleana:  $E = (a + b)(c + d)$

Para determinar os casos de teste dessa expressão, são conside-

rados dois conjuntos de valores. Um conjunto contendo casos de teste que fazem o valor de  $E$  ser verdadeiro e um conjunto contendo casos de teste que fazem o valor de  $E$  ser falso.

Os termos da expressão são representados por :  $ac + ad + bc + bd$

Os casos de teste que fazem o primeiro termo ser verdade são:

$\{(t, f, t, f); (t, f, t, t); (t, t, t, f); (t, t, t, t)\}$

Nota-se por exemplo que o segundo caso de teste faz com que o segundo termo da expressão também seja verdadeiro, o terceiro caso de teste faz com que o terceiro termo da expressão seja verdadeiro , e o quarto caso de teste faz com que todos os termos sejam verdade. Então o unico caso de teste que faz com que o primeiro termo seja verdade é  $\{(t, f, t, f)\}$

A partir dessa ideia obtem-se o primeiro conjunto:

$\{(t, f, t, f), (t, f, f, t), (f, t, t, f), (f, t, f, t)\}$ .

Assim cada caso de teste faz com que um único termo da expressão assuma um valor verdadeiro.

Já para o segundo conjunto são obtidos os seguintes valores  $\{(f, f, t, f), (t, f, f, f), (f, f, f, t), (f, t, f, f)\}$ .

### 3.2 ESTRATÉGIA PARA DEFINIÇÃO DE CASOS DE TESTE

Os algoritmos CEG-BOR e CEG-BOR-MI podem ser aplicados no problema desse trabalho com algumas adaptações. A especificação do SIS já é representada formalmente por uma matriz e não é necessária a criação de uma grafo para a definição de casos de teste.

Assim, nessa sessão, é proposta uma estratégia capaz de preparar uma seleção de casos de teste para todas as lógicas descritas pela MCE, e ainda que é eficiente na busca das causas das falhas de operação da implementação. O algoritmo desenvolvido obedece três regras definidas a seguir:

*Regra 1:* Um conjunto de casos de teste é definido para cada coluna ou efeito da Matriz Causa e Efeito. Cada conjunto considera apenas as causas que estão relacionadas com esse efeito. No caso de lógicas singulares, a estratégia CEG-BOR é utilizada. Em caso de uma não singularidade (votação de múltiplas causas (2o3)), a estratégia CEG-BOR-MI é utilizada.

*Regra 2:* As causas que não pertencem ao conjunto definido pela Regra 1 assumirão um valor *Verdade* (t) quando o valor esperado do efeito do conjunto é falso, e um valor *Falso* (f) quando o valor

esperado do efeito do conjunto é verdadeiro. Dessa maneira, é possível diagnosticar a ocorrência de sinais de falsos-positivos ou falsos-negativos nos efeitos por causas que não fazem parte da função lógica.

*Regra 3:* Os casos de teste definidos pelas regras anteriores são listados em uma Tabela de Decisões. Essa Tabela não considera casos idênticos, ou seja, que apresentam os mesmos valores de entrada.

### 3.2.1 Exemplo

Deseja-se preparar uma seleção de casos de teste para a MCE exemplo da Figura 13 utilizando a estratégia de geração proposta na seção anterior. Esse exemplo didático compreende lógicas que estão presentes no caso real do SIS.

EXEMPLO DE MCE			EFEITO			
			Equipamento	Emerg. Alarm	Fire Detected	Fire Confirmed
CAUSA			TAG	ESD-101	FD-101	FC-102
Equipamento	Votação	TAG				
Manual Fire Alarm		HSS-101100		X	A1	
Smoke Fire Detector		YST-101200		X	X	T10
Flame Fire Detector	1oo3	UST-101001			A1	
Flame Fire Detector	2oo3	UST-101002				X
Flame Fire Detector		UST-101003				

Figura 13 – Matriz Causa e Efeito Exemplo

Iniciando pela Regra 1, um conjunto de casos de teste é gerado para cada um dos efeitos “ESD-101”, “FD-101”, “FC-102”.

O efeito “**ESD-101**” está relacionado por uma condição “OR” entre “**HSS-101100**” e “**YST-101200**”. Essa relação é singular, portando o algoritmo CEG-BOR pode ser utilizado para definição dos casos de teste (Regra 1). Os vetores da Tabela 2 estabelecem os valores de cada causa da MCE em cada um dos casos de teste gerados. Cada vetor contém um valor para as cinco causas ou entradas da matriz, o qual pode assumir *Verdade* (t), ou *Falso* (f), ou ainda Indefinido (x) (nenhum valor atribuído).

Nota-se, que para a relação “OR” entre “**HSS-101100**” entre “**YST-101200**”, o caso de teste que possui o valor em *Verdade*, nessas duas entradas, não é selecionado pelo algoritmo, pois esse caso pode mascarar uma falha perigosa quando uma das causas estiver com problema. As outras entradas que não pertencem a lógica “OR” assumem valor *Verdade* quando “**ESD-101**” possui valor *Falso*, ou ainda, podem assumir *Falso* quando “**ESD-101**” possui valor *Verdade* (Regra 2).

CEG-BOR (ESD-101)	Casos de Teste da MCE	ESD-101
(f,f,x,x,x)	(f,f,t,t,t)	f
(f,t,x,x,x)	(f,t,f,f,f)	t
(t,f,x,x,x)	(t,f,f,f,f)	t

Tabela 2 – Definição de Casos de Teste- Efeito ESD-101

Na Tabela 3 é definido o conjunto de casos de teste do efeito “**FD-101**”. Esse efeito é acionado por uma votação 1003 de um grupo de sensores de detecção de fogo. Essa votação é uma função lógica singular pois é equivalente à uma lógica “OR” entre as causas do grupo. Além disso, o grupo de sensores faz uma relação “AND” com o alarme manual de fogo. Ainda, o resultado dessa lógica faz uma relação “OR” com o detector de fumaça. A expressão lógica formada é singular e pode ser utilizada a estratégia CEG-BOR para definição dos casos de teste (Regra 1).

A lógica da última coluna, referente ao efeito “**FC-102**”, inclui uma não singularidade. Isso se deve à votação 2003 entre os sensores de detecção de fogo. Nessa caso, o algoritmo CEG-BOR-MI é aplicado para obtenção dos casos de teste da Tabela 4 (Regra 1) e na sequência, utilizando a Regra 2, são definidos os valores da entrada de “**HSS-101100**”.

Nota-se que existe um termo “T10” (tempo que a causa deve permanecer ativada) nas relações desse último efeito. Quando for execu-

CEG-BOR (FD-101)	Casos de Teste da MCE	FD-101
(t,f,f,f)	(t,f,f,f)	f
(f,f,t,f)	(f,f,t,f)	f
(t,f,t,f)	(t,f,t,f)	t
(t,f,f,t)	(t,f,f,t)	t
(t,f,f,t)	(t,f,f,t)	t
(t,t,f,f)	(t,t,f,f)	t

Tabela 3 – Casos de Teste - Efeito FD-101

CEG-BOR (FC-102)	Casos de Teste da MCE	FC-102
(x,f,t,f)	(t,f,t,f)	f
(x,f,f,t)	(t,f,f,t)	f
(x,f,f,t)	(t,f,f,f,t)	f
(x,f,t,t)	(f,f,t,t)	t
(x,f,f,t)	(f,f,f,t)	t
(x,f,t,t)	(f,f,t,f,t)	t
(x,t,t,f)	(f,t,t,f)	t

Tabela 4 – Casos de Teste - Efeito FC-102

tado os casos de teste em que “YST-101200” deve assumir *Verdade*, deve ser considerado um período de tempo suficiente para testar a ativação do efeito. Só assim o diagnóstico de falha pode atestar uma inconformidade.

Por fim todos os casos de testes gerados para os três efeitos, são então agrupados para formar a Tabela de Decisões da Tabela 5. A repetição dos seguintes casos de teste ((t,f,f,f), (t,f,t,f), (t,f,f,t) e (t,f,f,t)) não são consideradas na Tabela(Regra 3).

### 3.3 CONSIDERAÇÕES FINAIS

A geração de casos de teste compõe uma das etapas do método de testes proposto nesse trabalho. A abordagem escolhida projeta casos de teste baseando-se na especificação. Algumas estratégias conhecidas foram investigadas a fim de aplicar o teste da lógica das funções de segurança do SIS. Verificou-se a viabilidade da aplicação das técnicas CEG-BOR e CEG-BOR-MI no caso de estudo considerando algumas regras para implementação dessas estratégias. O próximo capítulo tra-



Casos de Teste da MCE	ESD-101	FD-101	FC-102
(f,f,t,t,t)	f	f	t
(f,t,f,f,f)	t	t	t
(t,f,f,f,f)	t	f	f
(f,f,t,f,f)	f	f	f
(t,f,t,f,f)	t	t	f
(t,f,f,t,f)	t	t	f
(t,f,f,f,t)	t	t	f
(t,t,f,f,f)	t	t	t
(f,f,t,t,f)	f	f	t
(f,f,f,t,t)	f	f	t
(f,f,t,f,t)	f	f	t
(f,t,t,f,f)	f	f	t

Tabela 5 – Tabela de Decisões do Exemplo

tará da geração dos modelos de oráculos que serão responsáveis pelo diagnóstico de inconformidades da lógica durante a execução dos casos de teste obtidos pela técnica proposta nesse capítulo.



## 4 GERAÇÃO DE ORÁCULOS

O diagnóstico do comportamento de operação de um dispositivo é um dos aspectos mais importantes durante um teste. Um oráculo é capaz de automatizar esse processo comparando os valores observados do comportamento das saídas do dispositivo sob teste com os valores das saídas esperadas definidas na especificação (HAMLET, 1994). O diagnóstico de conformidade ou falha pode ser feito pelos oráculos através da execução dos casos de teste.

As Redes de Petri favorecem a representação de sistemas discretos onde as variáveis de estado variam bruscamente a certos instantes. Os modelos de redes de Petri permitem representar de forma organizada a lógica de operação de um sistema e podem ser muito bem aplicados como oráculos para as lógicas de segurança do SIS. Através de modelos de Redes de Petri Temporizadas, é possível ainda representar funções de segurança que levam em conta um período de tempo de acionamento de uma entrada (BERTHOMIEU; DIAZ, 1991).

Os modelos possuem duas funções básicas bem definidas: expressar as funções específicas de segurança da lógica do CLP, e ainda, diagnosticar a existência de falhas seguras e perigosas decorrentes de um problema no comportamento da lógica do CLP. Esse capítulo descreve inicialmente a etapa de concepção dos modelos de oráculos e posteriormente a etapa de execução e obtenção do diagnóstico de inconformidades.

### 4.1 CRIAÇÃO DE MODELOS DE ORÁCULOS DA MCE

Buscando avaliar de forma independente o comportamento de cada saída do dispositivo sob teste e permitir a análise da ocorrência de sinais falso positivos (falhas seguras) e sinais falso negativos (falhas perigosas), propõe-se a criação de um modelo de oráculo para cada efeito (coluna) da MCE. Essa criação é feita antes da execução dos casos de teste e deve seguir regras para concepção do modelo.

A estrutura de cada modelo é dividida por módulos que possuem finalidades diferentes. A modularização permite que o modelo fique mais claro e seja construído de uma forma padronizada. A estrutura de um oráculo pode conter até cinco módulos, são eles: módulo de diagnóstico, módulo lógico, módulo de temporização, módulo de leitura e módulo de controle. Os detalhes da estrutura de cada módulo

serão apresentados nas seções a seguir, bem como alguns exemplos que utilizam algumas funções lógicas de uma MCE. A Figura 14 mostra um exemplo de oráculo contendo os cinco módulos definidos. Após a definição dos módulos é feita uma composição desses para obtenção do modelo completo. Essa operação de composição se dá através de uma operação matemática conhecida como fusão de lugares de Redes de Petri.

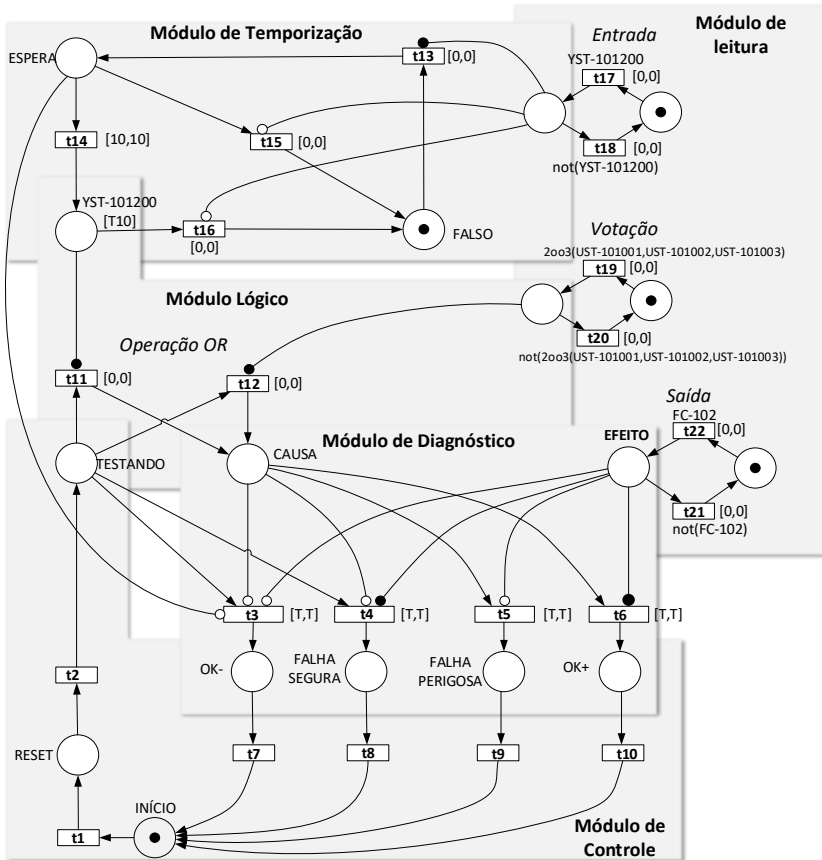


Figura 14 – Exemplo de Modelo de Oráculo

Os modelos de Redes de Petri são compostos basicamente por alguns elementos gráficos, são eles: as transições (representadas por

retângulos ou barras), os lugares ( representados por circunferências) os arcos (representados por linhas com setas, circunferências ou círculos na extremidade) e as fichas (representadas por círculos).

Os lugares de um modelo podem representar os estados do sistema e as transições os eventos que ocorrem durante a sua execução. As fichas por sua vez, podem ser alocadas somente nos lugares e podem indicar se um estado está ativo. As fichas são disparadas pelas transições que são interligadas com os lugares através dos arcos. Os arcos possuem ainda três principais tipos: os arcos regulares (linha ou arco com seta na extremidade), os arcos inibidores (linha ou arco com uma circunferência na extremidade) e os arcos de leitura (linha ou arco com um círculo na extremidade).

Os arcos regulares permitem o disparo de uma ficha por uma transição associada, e ainda, uma ficha é consumida no momento desse disparo. O arco de leitura também permite o disparo de uma transição, porém nessa situação, uma ficha não é consumida. Já o arco inibidor bloqueia o disparo de uma ficha caso o lugar associado tenha posse de uma ficha.

#### 4.1.1 Módulo de Diagnóstico

Durante a execução de um caso de teste é necessária a verificação dos sinais das saídas do CLP e avaliação desses com um resultado esperado. Dessa forma é possível oferecer um diagnóstico de inconformidades do que foi testado. Os resultados do diagnóstico são determinados pelas situações descritas na Tabela 6. Podem ser revelados quatro diferentes tipos de diagnóstico, dois deles correspondem as inconformidades (*Safe failure* ou *Dangerous Failure*) e os outros dois aos resultados de conformidade entre o sinal de saída observado e o sinal de saída esperado.

O módulo de diagnóstico possui uma estrutura genérica repre-

Causa (Efeito Esperado)	Efeito Observado	Diagnóstico
False	False	Conformidade (OK -)
False	True	Safe Failure
True	False	Dangerous Failure
True	True	Conformidade (OK +)

Tabela 6 – Diagnóstico do Oráculo

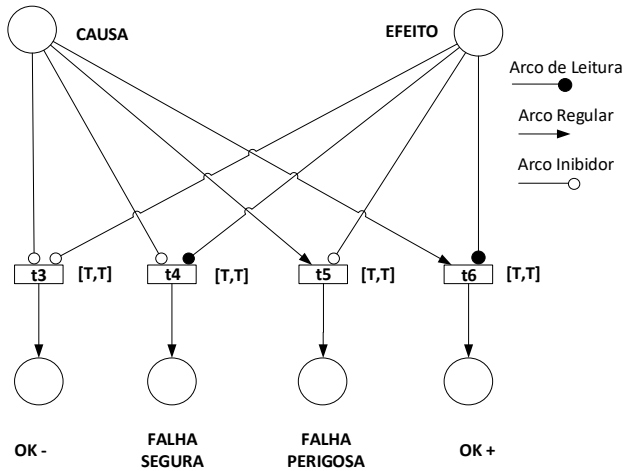


Figura 15 – Módulo de Diagnóstico

sentada pela Figura 15 e possui sempre a mesma forma para cada um dos oráculos. Cada módulo é composto por lugares de Redes de Petri os quais obtendo uma ficha, podem representar: ativação dos sinais de causa e efeito (CAUSA, EFEITO) e o resultado do diagnóstico (FALHA SEGURA, FALHA PERIGOSA, OK -, OK +).

O veredito do teste é obtido através da verificação das fichas nos quatro lugares que correspondem ao diagnóstico. Essas fichas são depositadas através do disparo das transições  $t_3$ ,  $t_4$ ,  $t_5$  e  $t_6$  e apenas um dos quatro lugares receberá uma ficha indicando o resultado. O disparo dessas transições é determinado pela presença de fichas nos lugares CAUSA e/ou EFEITO e obedecem as relações representadas pela Tabela 6. Além disso um tempo de disparo é estabelecido em cada uma dessas transições a fim de estabelecer um tempo de tolerância para o diagnóstico de alguma falha. O tempo de tolerância  $T$  deve considerar o ciclo de varredura do CLP (*scan cycle*) e o tempo de comunicação entre o testador e o CLP. Essa tolerância garante a observação de uma ação efetiva do SUT.

Por exemplo, o disparo da transição  $t_4$  ocorrerá quando existir uma ficha no lugar EFEITO e não existir uma ficha no lugar CAUSA por  $[T]$  segundos. Isso indicará a presença de uma falha do tipo *Safe Failure*.

As fichas dos lugares CAUSA ou EFEITO são depositadas através de outros módulos. O lugar EFEITO possuirá uma ficha quando o sinal correspondente de saída do CLP é ativo. Já o lugar CAUSA recebe uma ficha através do módulo lógico capaz de observar a lógica de intertravamento das entradas do dispositivo de acordo com a especificação da MCE.

#### 4.1.2 Módulo de Controle

Após a execução de um caso de teste e obtenção do diagnóstico de conformidade ou inconformidade é necessário que o oráculo se prepare para avaliar o próximo caso de teste. Cada modelo de oráculo possui um módulo de controle capaz de reinicializá-lo. Na Figura 14, as transições t7, t8, t9, e t10 são responsáveis por consumir a ficha do lugar correspondente ao diagnóstico do caso de teste anterior.

Com o disparo da transição t1, uma ficha é colocada no lugar RESET, indicando que o valor das entradas do CLP deve retornar ao seu estado inicial. A transição t2, por sua vez é responsável por enviar uma ficha ao lugar TESTANDO que indica o estado de avaliação do novo caso de teste.

#### 4.1.3 Módulo Lógico

A fim de tratar as funções lógicas “AND” e “OR” da MCE, descritas pela diretriz, propõe-se um módulo lógico. A Figura 16 define a estrutura do módulo proposto. Esse é formada pelos lugares que representam: a condição do teste (TESTANDO), o status dos sinais de entrada do CLP (IN\_01 e IN\_02) e o resultado da lógica de entradas do CLP (CAUSA).

O resultado da lógica é obtido pelo disparo das transições t11, para o caso da lógica “AND”, ou t12 para o caso da lógica “OR”. O disparo das transições é determinado pela leitura de fichas dos lugares correspondentes ao valor das entradas do CLP. O lugar TESTANDO permite o depósito de uma única ficha no lugar referente a causa. Se existir um próximo caso de teste para análise, uma ficha é recolocada através da seção de controle.

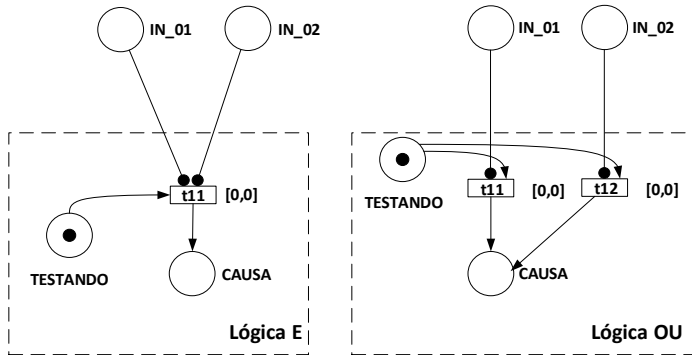


Figura 16 – Módulo Lógico

#### 4.1.4 Módulo de Temporização

Na especificação da MCE pode existir uma condição que determina um atraso para a ativação de um efeito por meio de um comando contínuo no sinal de entrada ou causa. O período em que o sinal de entrada deve ficar ativo é especificado na própria matriz. A Figura 17 define a estrutura do módulo de temporização para um período de T segundos.

Nesse modelo, o sinal de entrada IN\_01 deve manter um comando contínuo durante um tempo “T” unidades de tempo. Esse tempo é especificado na transição t14. Para que o modelo não forneça um diagnóstico antes do cumprimento desse prazo propõe-se a inclusão do lugar ESPERA. Na concepção do modelo completo, um arco inibidor que sai do lugar ESPERA, é ligado à transição t3 do módulo de diagnóstico, impedindo o diagnóstico de OK- antes do término do período de ativação da entrada, como mostra a Figura 14.

Se em determinado instante da contagem do tempo, a entrada deixa de manter o estado ativo, ocorrerá o disparo da transição t15 que consumirá a ficha do lugar ESPERA e enviará uma ficha ao lugar FALSO. Já no caso de cumprimento do prazo, a transição t14 será disparada e enviará uma ficha ao lugar IN\_01[TEMP].



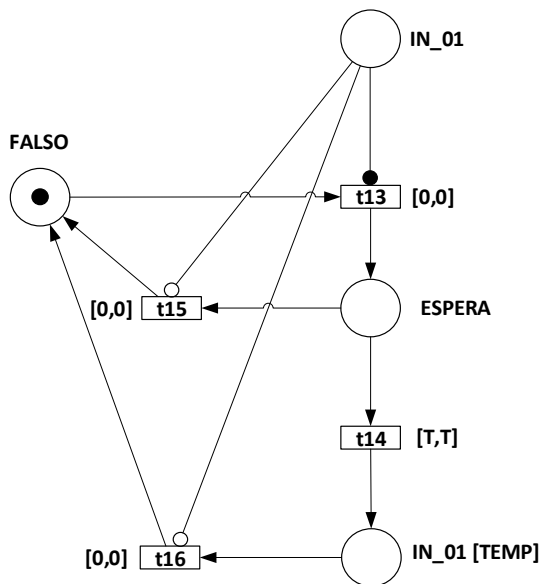


Figura 17 – Módulo de Temporização - Período de ativação de T segundos

#### 4.1.5 Módulo de Leitura de Entradas e Saídas e Tratamento de Votação

O módulo de leitura é responsável pela verificação dos valores das entradas e saídas do CLP. Cada entrada ou saída possui um lugar que corresponde ao sinal ativo e um lugar que corresponde ao sinal desligado. As transições que interligam esses lugares disparam quando há uma mudança do valor de entrada ou saída correspondente.

O módulo de leitura é capaz de tratar a leitura de um grupo de sensores quando houver uma operação de votação. Nestes casos as transições possuem uma transição que dispara caso a sua condição, votação entre os sensores, seja verdadeira. A Figura 18 compara a estrutura do módulo de leitura convencional (sem votação) e o módulo com tratamento da votação. No caso onde não há votação o disparo da transição t19 se dá quando o sinal de IN\_4 passa para o estado ativo, e o disparo t20 se dá quando o sinal de IN\_4 é desativado.

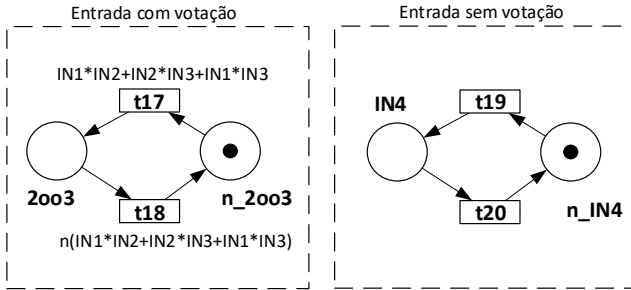


Figura 18 – Módulo de Leitura/Votação

4.2 EXEMPLO I

Tomando como base a MCE da Figura 19 deseja-se construir um oráculo que seja capaz de verificar a ocorrência de falhas do efeito da saída **GC-600** da tabela. A partir das definições feitas nas seções anteriores e utilizando uma estratégia de fusão de lugares de Redes de Petri obtém-se o modelo da Figura 20.

EXEMPLO DE MCE			EFEITO			
			Equipamento	TAG	GC-600	FC-700
CAUSA			Gas Confirmed	Fire Confirmed		
Equipamento	Votação	TAG				
Smoke Detector A	1oo3	UST-100A				
Smoke Detector B	2oo3	UST-100B				T,A1
Smoke Detector C		UST-100C				
Heat Fire Detector		TST-400			A1	A1
Gas CH4		AST-500			A1	X

Figura 19 – Matriz Causa e Efeito

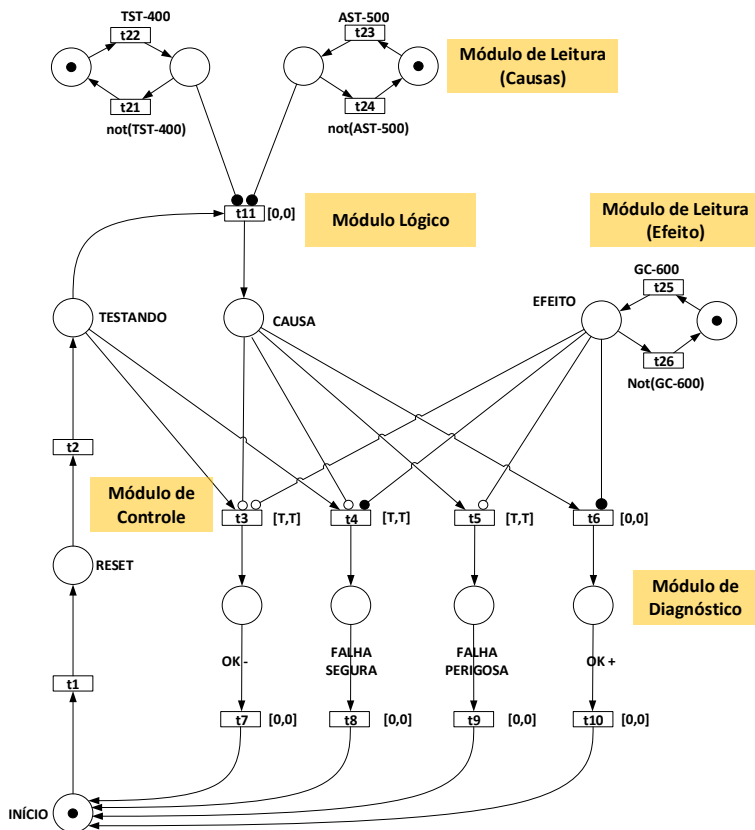


Figura 20 – Exemplo 1 - Oráculo do efeito **GC-600**

Algumas condições são estabelecidas para o disparo das transições *t22*, *t23* e *t25*. Cada disparo ocorrerá somente no caso em que os sinais respectivos de entrada/saída (**TST-400**, **AST-500** e **GC-600**) do CLP estiverem ativos.

Assim que o teste for iniciado e os dois sinais de entrada estiverem ativos, será depositada uma ficha no lugar *CAUSA* com o disparo da transição *t11* identificando que a lógica foi atendida. A partir desse instante inicia-se a contagem do tempo da transição *t5*. Caso o efeito **GC-600** seja ativado dentro desse tempo de tolerância, as fichas dos

lugares CAUSA e EFEITO permitirão o disparo de  $t_6$  e o diagnóstico de conformidade (OK +) será identificado. Caso o efeito **GC-600** não seja ativado dentro do tempo de tolerância, a transição  $t_5$  fará o disparo de uma ficha e o diagnóstico de falha perigosa será identificado.

Os lugares que correspondem aos outros dois diagnósticos somente receberão uma ficha quando a lógica dos sinais de entrada não for atendida. No instante em que o teste é iniciado, inicia-se também a contagem do tempo da transição  $t_3$ . Caso o efeito **GC-600** não seja ativado durante esse tempo de tolerância, uma ficha será disparada ao lugar que indica a conformidade (OK -). Caso contrário, inicia-se a contagem do tempo da transição  $t_4$  que será disparada após esse prazo e indicará uma falha segura da saída sob teste.

A determinação do tempo da transição de  $t_4$  é fundamental não apenas para garantir que o CLP tenha tempo de atualizar o efeito antes do diagnóstico, mas também para garantir que as fichas da seção da lógica sejam processadas antes da leitura do sinal de saída do CLP. Caso esse tempo não seja considerado, um atraso no processamento das fichas da seção lógica, fornecerá um diagnóstico errôneo da situação observada no teste.

### 4.3 EXEMPLO II

Tomando como base a MCE da Figura 19 deseja-se construir um modelo que seja capaz de verificar a ocorrência de falhas do efeito da saída **FC-700** da tabela. A partir das definições feitas nas seções anteriores e utilizando uma estratégia de fusão de lugares de Redes de Petri obtém-se o modelo da Figura 21.

A operação desse modelo segue o mesmo princípio do Exemplo I porém nesse exemplo existem dois módulos adicionais e a função lógica é diferente. No módulo de leitura existe a verificação da votação 2003 entre os sensores **UST-100A**, **UST-100B** e **UST-100C**. Quando a condição verdadeira da votação é atendida, o módulo do timer inicia a contagem do tempo T e o oráculo entra em modo de espera. Esse tempo T é especificado pela matriz causa e efeito. Após o cumprimento desse período de tempo o módulo Timer entrega uma ficha ao módulo lógico. Assim o oráculo analisará a lógica descrita pela matriz e executará o diagnóstico conforme a explicação do exemplo anterior.

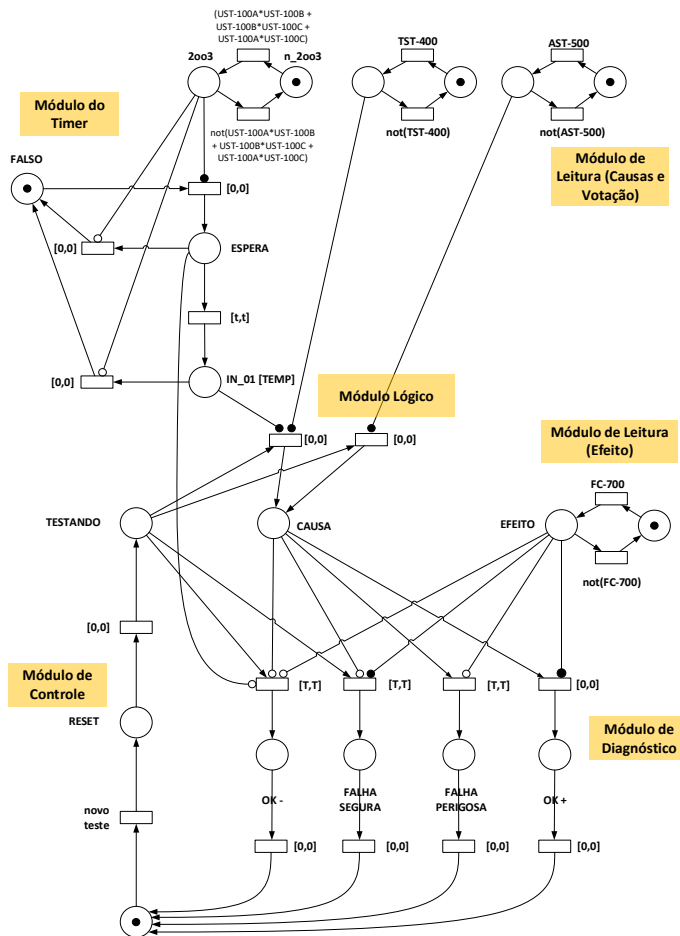


Figura 21 – Oráculo do efeito **FC-700**

#### 4.4 CONCLUSÕES FINAIS DO CAPÍTULO

O diagnóstico da execução dos casos de teste pode ser feito por meio de oráculos. Nesse trabalho, os oráculos são representados por modelos de Redes de Petri que são compostos por módulos com diferentes funcionalidades. Cada modelo possui um módulo que representa a lógica de um determinado efeito e outro módulo capaz de realizar

um diagnóstico se essa lógica é respeitada pela execução da implementação. Nesse capítulo foram tratados dois exemplos para entendimento da construção dos modelos de uma matriz que possui funções lógicas de segurança de um Sistema Instrumentado de Segurança. O próximo capítulo tratará da implementação desse modelo por meio do desenvolvimento e proposição de uma ferramenta computacional de testes.

## 5 VALIDAÇÃO DO MÉTODO

Esse capítulo apresenta uma solução para validar o método proposto nesse trabalho. A implementação conta com duas ferramentas computacionais. Uma foi desenvolvida para automatizar o Teste de Conformidade do CLP e a outra para criação e edição da MCE. Esta última tem a finalidade de exportar os dados da matriz para a ferramenta de teste. A Figura 22 apresenta uma visão geral da solução desenvolvida.

Durante o projeto do SIS, o documento da MCE é preparado por engenheiros do processo e o código Ladder do CLP é desenvolvido por uma equipe de programadores seguindo uma metodologia. Normalmente o documento da MCE é preparado em planilhas eletrônicas, as quais são suscetíveis a erros de preenchimento e ainda não fornecem uma estrutura de dados que favorece a exportação das suas informações. Para isso foi desenvolvida uma ferramenta para criação e edição do documento da Matriz Causa e Efeito obedecendo as diretrizes da Petrobrás.

O editor de MCE permite a criação do documento em pdf, exigido pela norma interna da empresa, e ainda oferece uma opção para a criação de um arquivo XML contendo uma base de dados da especificação. Assim, os dados contidos nesse arquivo serão aproveitados pelo programa responsável pela geração e execução automática do teste durante a etapa de realização desses.

Durante a etapa de Testes, o CLP é colocado em execução e um servidor OPC permite a comunicação com a ferramenta de testes (Testador Automático). Dessa forma é possível automatizar o procedimento de manipulação dos bits de memória do CLP através da ferramenta. Além disso, a ferramenta é capaz de gerar os casos de teste baseados no arquivo XML da especificação, implementar as técnicas de definição de casos de teste e os modelos de observadores de redes de Petri e ainda exibir uma interface ao usuário mostrando o progresso do teste.

### 5.1 EDITOR DE MCE

As ferramentas mais comuns utilizadas na geração de tabelas, como a MCE, oferecem uma alta flexibilidade na edição de dados e não conduzem um projetista a descrever as funções lógicas do SIS seguindo uma norma estabelecida. Em vista desse problema, uma ferramenta de

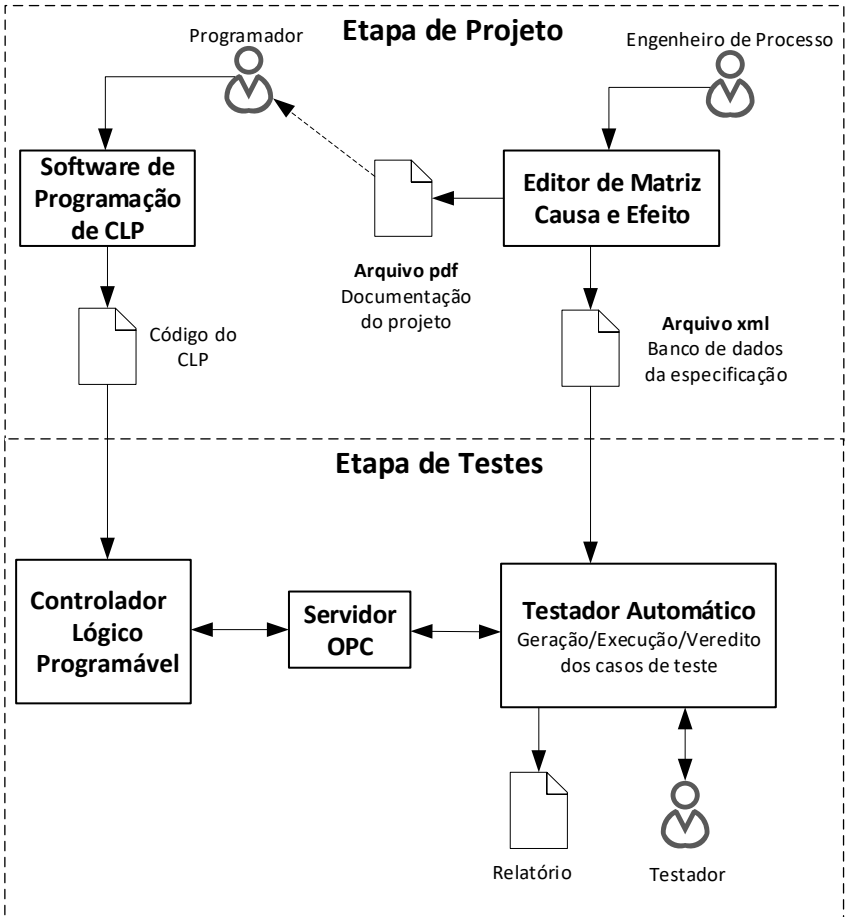


Figura 22 – Solução para a automatização do Teste de Conformidade



edição da MCE foi desenvolvida em Saito (2016) visando oferecer uma interface gráfica que facilite a criação do documento da MCE, padronize os dados inseridos no documento seguindo a norma ET-3000.00-1200-800-PGT-006 (2000) da empresa, gere um arquivo pdf para fins relativos à documentação do projeto, e ainda gere um arquivo no padrão XML estruturado para importação dos dados pela ferramenta de testes. A Figura 23 apresenta uma visão da tela principal dessa ferramenta.

## 5.2 FERRAMENTA AUTOMÁTICA DE TESTES

A ferramenta automática de testes é responsável pela preparação e execução automática do teste. Três etapas bem definidas determinam a operação do programa.

A primeira delas se refere a preparação do teste onde os dados de entrada são importados pelo programa e as seqüências de sinais de entrada são geradas e executadas como comandos ao CLP. Ainda nessa etapa, são gerados os oráculos formados por modelos de Redes de Petri e que farão a avaliação do comportamento da implementação durante a próxima etapa.

A segunda etapa se refere a execução do teste. Nesta a seqüência

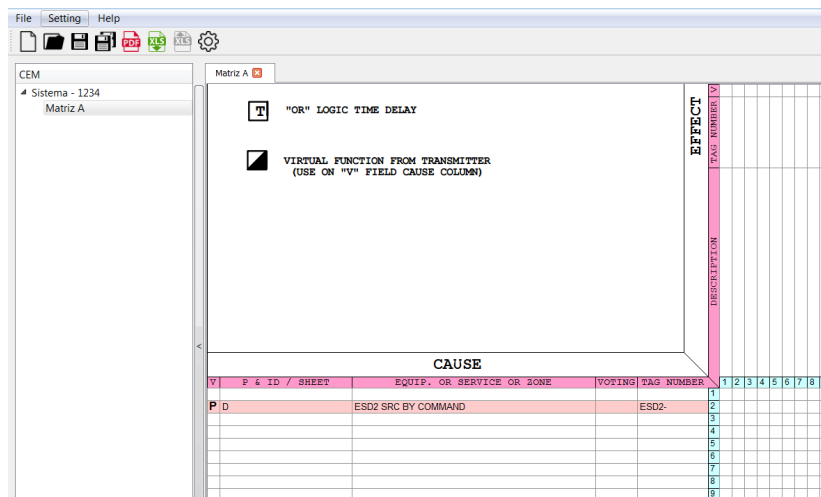


Figura 23 – Tela do Editor de Matriz Causa e Efeito

gerada pelo passo anterior é então executada para a leitura das informações do CLP pela ferramenta. Os valores de entrada/saída são então interpretados pelo programa e traduzidos como comandos para as transições de leitura dos modelos de Redes de Petri.

A terceira etapa refere-se a apresentação dos resultados do teste. A estrutura que compõe a ferramenta foi subdividida em diferentes módulos responsáveis pelas tarefas desempenhadas durante as três etapas citadas:

- **Entrada de Dados:** permite a importação dos dados de entrada da MCE e ainda é responsável pelo processamento desses para utilização dos módulos seguintes;
- **Gerador de Casos de Teste:** gera uma lista de casos de testes utilizando uma estratégia definida pelo usuário ou testador.
- **Gerador de Observadores:** realiza a interpretação dos dados da especificação representando-os como observadores baseados em Redes de Petri;
- **Comunicação OPC:** cria um cliente OPC que permite a comunicação OPC entre a ferramenta e o servidor OPC. O servidor por sua vez comunica-se diretamente com o CLP.
- **Executor dos modelos:** interpreta a leitura de operação do comportamento do sistema e executa o modelo para obtenção do diagnóstico
- **Interface com o Usuário:** refere-se a interface com o testador humano onde é possível selecionar o arquivo XML de entrada, configurar a comunicação com o CLP, executar o teste automático e ainda analisar o progresso do teste.
- **Geração do resultado:** emite um relatório com os resultados do diagnóstico de cada oráculo.

### 5.2.1 Entrada de Dados

O objetivo desse módulo é permitir a importação dos dados da especificação contidos na MCE. O editor de MCE oferece a opção da criação de um arquivo XML o qual favorece a interpretação dos dados pelo módulo da ferramenta de teste. Além das informações relativas à relação direta entre as causas e os efeitos, o documento ainda fornece

uma listagem com as TAGs das entradas e saídas correspondentes. As TAGs da MCE são então associadas às TAGs do servidor OPC que por sua vez, correspondem as variáveis de entrada/saída do CLP.

### 5.2.2 Gerador de Casos de Teste

A partir das entradas já lidas, o módulo pode preparar a sequência de casos de teste seguindo a estratégia de seleção CEG-BOR e CEG-BOR-MI apresentada no capítulo 3. Nessa etapa são atribuídos os valores que serão enviados às entradas bem como o tempo que essas permanecerão ativados durante a execução. Esse tempo é importante pois o teste de algumas funcionalidades só é conclusivo ao final desse.

### 5.2.3 Geração de Observadores

A partir dos dados da especificação são também gerados os modelos de observadores baseados em Redes de Petri. O módulo é capaz de criar várias instâncias de observadores, cada um adaptado para um efeito (coluna) e suas respectivas relações com as causas (linhas).

A biblioteca SNAKES (POMMERAU, 2008) suporta a criação dos modelos de observadores. Essa biblioteca específica do Python disponibiliza funções para a construção e execução de modelos em Redes de Petri. Outra característica importante da biblioteca SNAKES é seu sistema de *plugins* que permite a extensão das ferramentas já existentes. Com isso foi possível adicionar o suporte à Redes de Petri temporizadas, fundamental para o projeto.

Também foi adicionado um *plugin* para a ilustração das Redes de Petri criadas, o qual gera um arquivo de imagem com os lugares, fichas e transições. Desse modo é possível verificar e comparar o modelo criado pela ferramenta automática com o modelo esperado.

### 5.2.4 Interação da Ferramenta com o SUT via OPC

Neste trabalho foi utilizado o protocolo OPC-DA (*data access*) para comunicação entre a ferramenta de testes e o CLP. Esse protocolo industrial permite a comunicação com diversos tipos de fabricantes e 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 su-

pervisórios e, além da ampla aceitação por parte da indústria, é amplamente suportado pela comunidade de desenvolvimento de software através de diferentes bibliotecas de acesso.

A ferramenta pode ainda estabelecer comunicação OPC com um simulador de CLP, eximindo a necessidade do hardware e permitindo o teste da lógica em laboratório ou ambiente de testes.

### 5.2.5 Execução dos Observadores

Cada oráculo criado pela ferramenta é alocado em uma *Thread* individual, para que todos executem paralelamente durante a análise dos valores das variáveis de entrada/saída do CLP. Durante a execução do teste, as entradas e saídas do SUT são lidas através da comunicação OPC. Cada mudança no valor de entrada ou saída é identificada pelo modelo e a transição correspondente dessa ação é disparada.

### 5.2.6 Interface com o Usuário

A interface visual do usuário busca facilitar a configuração, execução e análise dos resultados do teste pelo testador ou avaliador humano. Algumas funcionalidades estão disponíveis ao usuário:

- Carregar o arquivo xml de entrada
- Configurar a comunicação OPC e associar as variáveis da especificação com as TAGS do servidor
- Definir a técnica de definição de casos de teste
- Executar o teste da matriz
- Acompanhar o progresso do teste
- Gerar relatório de Falhas

Além de gerar casos de teste automaticamente com as técnicas apresentadas nesse trabalho (Exemplo: CEG-BOR, Randômico) é possível definir manualmente casos de teste onde são selecionadas as causas a serem forçadas. A Figura 24 mostra a janela principal da ferramenta de testes

Usualmente o projeto do SIS de uma planta possui mais de uma folha de MCE. A ferramenta desenvolvida permite a importação de todas as matrizes do projeto. A janela principal mostra um resumo do

teste de cada uma das matrizes apresentando ao usuário o número de inconformidades encontradas. No exemplo da Figura 24 foram carregadas oito matrizes de um subsistema de um caso real.

Existe ainda uma tela de visualização da execução do teste da MCE. A medida que os casos de teste são executados são apresentados os resultados de ativação das causas/efeitos na tela. No final do teste pode ser visualizado o diagnóstico de inconformidades de cada um dos casos de teste, como mostrado na Figura 25.

A Figura 26 apresenta a tela de configuração da comunicação da ferramenta com o CLP. Essa comunicação é feita via OPC e cada causa ou efeito carregada na ferramenta deve ser associada à uma variável OPC do servidor. Ao terminar a configuração é então permitida a execução do teste.

### 5.2.7 Geração do Relatório

Foi implementado ainda uma funcionalidade para geração de um relatório de inconformidades contendo informações sobre todas as falhas (*Safe Failure* e *Dangerous Failure* encontradas durante a execução do teste. Esse relatório pode servir como documentação do TAF (Teste de Aceitação de Fábrica) e atestar a conformidade ou inconformidade do sistema projetado.

## 5.3 ESTUDO DE CASO E RESULTADOS

O SIS da plataforma de produção de petróleo offshore P-51 da Petrobras serve como estudo de caso para aplicação do método proposto nesse trabalho. O sistema é composto por CLPs responsáveis por funções de segurança específicas as quais são divididas por subsistemas: *electrical subsystem* (ELE), *shutdown subsystem* (ESD), *fire and gas subsystem* (F&G), *control subsystem* (CON), *turret subsystem* (TUR), and *vessel subsystem* (VES). A lógica de programação de cada subsistema deve respeitar a documentação de especificação de segurança. A documentação do SIS foi concedida pela Petrobras, essa contém 130 matrizes Causa e Efeito e ainda diretrizes que descrevem requisitos para a criação da estrutura básica do programa Ladder.

Como já descrito anteriormente, a formatação das informações inseridas no documento da MCE não obedece um padrão específico mesmo com a existência de uma diretriz para a elaboração desse. As-

PLC Tester

Arquivo Ajuda

— 0 X

**Matrix Cause & Effect**

Production Wells - 1210	Analise	OK+	OK-	SF	DF
<input type="radio"/> 1210-12	<input checked="" type="radio"/> Não Testado	0	0	0	0
<input type="radio"/> 1210-13	<input type="radio"/> Não Testado	0	0	0	0
<input type="radio"/> 1210-14	<input type="radio"/> Não Testado	0	0	0	0
<input type="radio"/> 1210-15	<input type="radio"/> Não Testado	0	0	0	0
<input type="radio"/> 1210-16	<input type="radio"/> Não Testado	0	0	0	0
<input type="radio"/> 1210-17	<input type="radio"/> Não Testado	0	0	0	0
<input type="radio"/> 1210-18	<input type="radio"/> Não Testado	0	0	0	0
<input type="radio"/> 1210-19	<input type="radio"/> Não Testado	0	0	0	0
<input type="radio"/> 1210-20	<input type="radio"/> Não Testado	0	0	0	0
<b>Fire and Gas - 5530</b>					
<input type="radio"/> 5530-81	<input type="radio"/> Não Testado	0	0	0	0
<input type="radio"/> 5530-82	<input type="radio"/> Não Testado	0	0	0	0
<input type="radio"/> 5530-83	<input type="radio"/> Não Testado	0	0	0	0
<input type="radio"/> 5530-84A	<input type="radio"/> Não Testado	0	0	0	0
<input type="radio"/> 5530-85	<input type="radio"/> Não Testado	0	0	0	0
<input type="radio"/> 5530-86	<input type="radio"/> Não Testado	0	0	0	0
<input type="radio"/> 5530-88	<input type="radio"/> Não Testado	0	0	0	0
<input type="radio"/> 5530-92	<input type="radio"/> Não Testado	0	0	0	0
<b>Electric Power System - 5140</b>					
<input type="radio"/> 5140-46	<input type="radio"/> Não Testado	0	0	0	0
<input type="radio"/> 5140-121	<input type="radio"/> Não Testado	0	0	0	0
<input type="radio"/> 1223-27	<input checked="" type="radio"/> Não Testado	0	0	0	0
<b>Oil Exportation Pipeline Departures - 1223</b>					

**Controle Projeto**

Analise de Resultados do Projeto

OK+  NOQF

OK-  NOQF

Matrizes Conformes: 0 Matrizes Não Conformes: 0

Matrizes Incompletas: 0 Matrizes Não Testadas: 20

Subsistema: Fire and Gas

№ total de testes: 167

MCE	Método	Testes	Status
1	CEG-BOR-MI	24	Pronto
2	5520-82 CEG-BOR-MI	24	Pronto
3	5520-83 CEG-BOR-MI	24	Pronto
4	5520-84A CEG-BOR-MI	18	Pronto
5	5520-85 CEG-BOR-MI	26	Pronto
6	5520-86 CEG-BOR-MI	17	Pronto
7	5520-88 CEG-BOR-MI	6	Pronto
8	5520-92 CEG-BOR-MI	28	Pronto

Figura 24 – Tela Principal

PLC Tester

Arquivo Ajuda

Teste Automático Teste Manual

Análise de Resultados

OK+  OK-  NOKSF  NOKDF

OK  OK-  NOKDF  NOKDF

Casos de Teste em Conformidade: 12

Casos de Teste em Não Conformidade: 0

Casos de Teste Inconclusivos: 0

Controle de Testes

CEC-ROR-MI

12 testes

Importar Salvar

Teste	OK+	OK-	NOKSF	NOKDF
Teste 4	3	0	0	0
Teste 5	2	1	0	0
Teste 6	2	1	0	0
Teste 7	2	1	0	0
Teste 8	3	0	0	0
Teste 9	1	2	0	0
Teste 10	3	0	0	0
Teste 11	0	3	0	0
Teste 12	1	2	0	0

Parar Escalar

Log de Falhas

Tempo de Teste 00:00:49

OPC conectado ao subsistema Subistema Exemplo.  
Resposta: Matr: 000-1

Finalizado

"OR" LOGIC

"NOR" LOGIC (INVERTED OR)

"AND" LOGIC

"OR" LOGIC TIME DELAY

VIRTUAL FUNCTION FROM TRANSMITTER (USE ON "V" FIELD CAUSE COLUMN)

Legenda:

- Sem Memória
- OK+ (Com Causa e Com Efeito)
- OK- (Sem Causa e Sem Efeito)
- Falha Segura (Sem Causa e Com Efeito)
- Falha Perigosa (Com Causa e Sem Efeito)
- Causa Forçada Diretamente
- Causa Forçada Indiretamente

CAUSE	DESCRIPTION	TAG NUMBER	V	P & ID
E 4: ID / SHEET	EMERG ALARM	ES0-101	1	1
P	FIRE DETECTED	FD-101	1	2
P	FIRE CONFIRMED	FC-102	1	3
P				4
P				5
P				6
P				7
P				8
P				9
P				10
P				11
P				12

Figura 25 – Tela de Execução do teste da Matriz Causa e Efeito

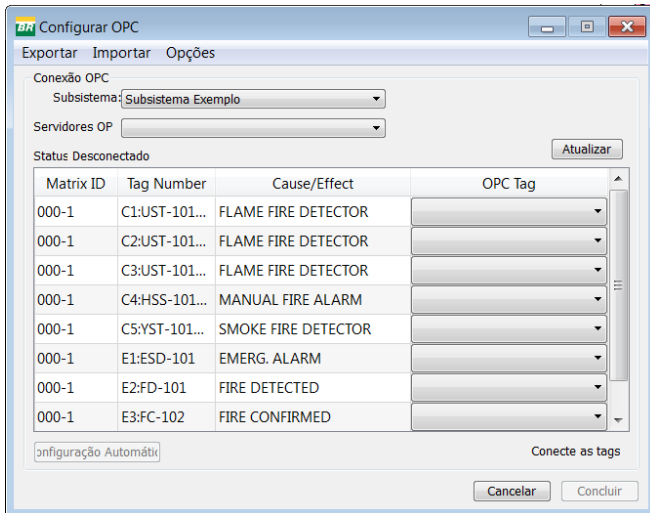


Figura 26 – Tela de configuração da comunicação

sim a ferramenta de testes desenvolvida nesse trabalho permite a preparação de uma versão *as-built* da documentação já existente. A ideia é avaliar de forma automática o comportamento da lógica de intertravamento do CLP com as funções de segurança descritas nas matrizes.

Dois exemplos são escolhidos para a execução de um teste automático utilizando a ferramenta desenvolvida. O primeiro deles envolve o teste de uma implementação didática que segue como especificação a MCE da Figura 27. Já o segundo exemplo envolve o teste automático do programa de CLP do subsistema de Fogo e Gás da plataforma da Petrobras buscando avaliar a conformidade com a Matriz Causa e Efeito.

### 5.3.1 Exemplo didático

No capítulo 3 foi utilizada a Matriz Causa e Efeito da Figura 27 para gerar os casos de teste do exemplo. Agora pretende-se executar esses casos e verificar o comportamento do CLP com a introdução de alguns erros no seu código Ladder

Para facilitar a explicação, primeiramente foi desenvolvido um código sem erros em conformidade com as lógicas do exemplo da matriz



EXEMPLO DE MCE			EFEITO			
			Equipamento	Emerg. Alarm	Fire Detected	Fire Confirmed
CAUSA			TAG	ESD-101	FD-101	FC-102
Equipamento	Votação	TAG				
Manual Fire Alarm		HSS-101100	X	A1		
Smoke Fire Detector		YST-101200	X	X	T10	
Flame Fire Detector	1oo3	UST-101001		A1		
Flame Fire Detector	2oo3	UST-101002			X	
Flame Fire Detector		UST-101003				

Figura 27 – Matriz Causa e Efeito Exemplo

em estudo, como mostra a Figura 28. Já a Figura 29 representa um código com alguns erros de programação. As falhas ocasionadas por esses erros devem ser detectadas durante a execução da ferramenta de testes.

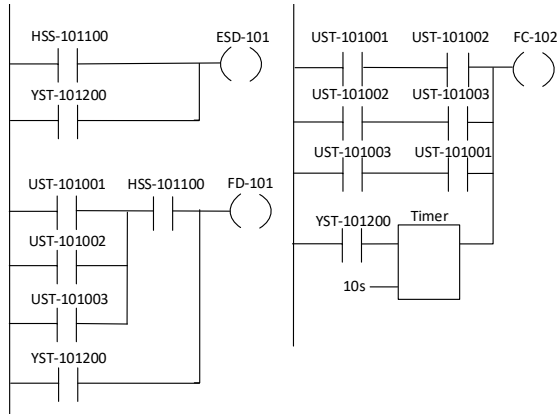


Figura 28 – Código Ladder em conformidade com a especificação

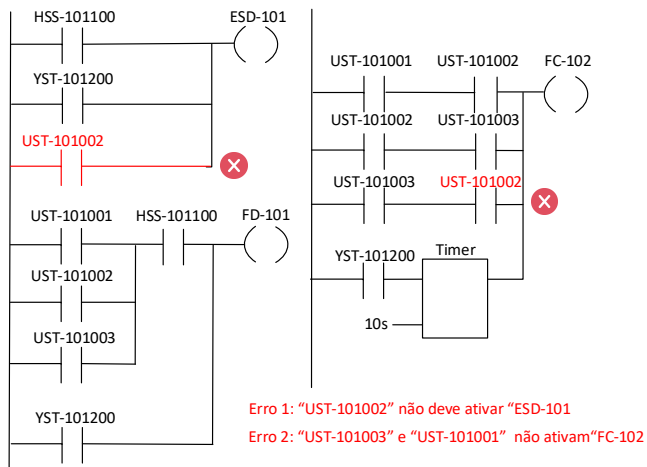


Figura 29 – Código Ladder com inconformidades

Ao executar o teste, quatro casos de teste diagnosticaram falhas perigosas e seguras, conforme a tela de apresentação de falhas da Figura 30. Três dos casos diagnosticaram Falhas Seguras (provenientes do erro 1 do código Ladder) e um caso de teste diagnosticou uma falha perigosa (proveniente do erro 2 do código Ladder).

Além disso um relatório de teste foi preparado descrevendo com detalhes o diagnóstico realizado pela ferramenta de testes. Na Figura 31 é apresentado o relatório de inconformidades gerado pela ferramenta.

### 5.3.2 Aplicação Real e Resultados

O teste na aplicação real envolve a avaliação da conformidade do comportamento da lógica de intertravamento do subsistema de Fogo e Gás a partir dos dados da especificação. A Figura 32 mostra uma giga de testes e um CLP com a lógica do SIS carregada. Ambos são utilizados para testes e simulações em laboratório do sistema de controle que opera na planta.

A ferramenta de testes desenvolvida nesse trabalho pode comunicar-se diretamente com o CLP via OPC, substituindo o uso da giga de testes. Foram avaliadas cinco matrizes de Causa e Efeito do subsistema de Fogo e Gás pelo método proposto nesse trabalho e pelo método exaus-



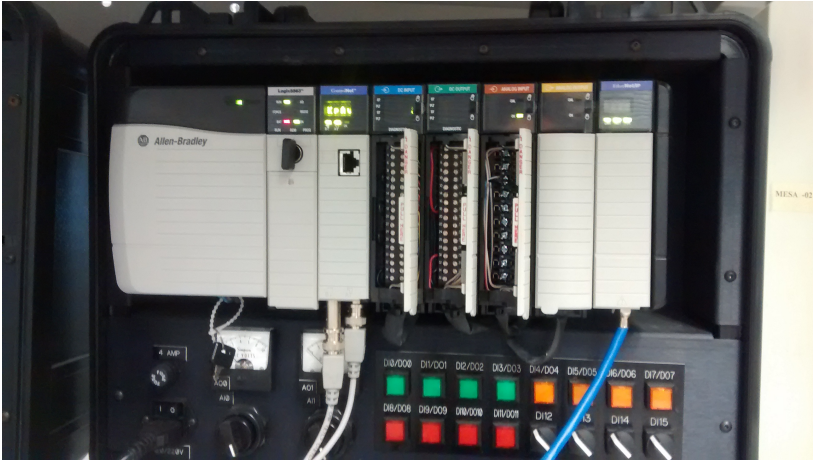


Figura 32 – CLP e giga de testes

tivo. Os resultados do teste são apresentados nas tabelas 7 e 8. As relações das MCEs analisadas envolvem votações de grupo de sensores, lógicas booleanas e *delays* (funções com tempo de acionamento do efeito). O experimento foi executado usando uma CPU com processamento *Dual Core*, 2.79 GHz.

A partir da Tabela é possível observar que o número de diagnósticos de cada matriz é igual ao produto do número de casos de teste com o número de efeitos.

Para a MCE da Zona A, por exemplo, a ferramenta executou 24 casos de teste e avaliou 4 efeitos, totalizando 96 diagnósticos (74

MCE do F&G	Causas/Efeitos	Estratégia Proposta		
		Casos de Teste	Tempo de Execução	OK+/OK-/SF/DF
Zone A	8/4	24	66s	74/22/0/0
Zone B	11/9	28	92s	123/129/0/0
Zone C	14/9	33	98s	140/152/5/0
Zone D	24/2	40	104s	51/29/0/0

Tabela 7 – Resultados do Teste Proposto da MCE do subsistema de Fogo e Gás

MCE do F&G	Causas/Efeitos	Teste Exaustivo		
		Casos de Teste	Tempo de Execução	OK+/OK-/SF/DF
Zona A	8/4	256	523s	942/82/0/0
Zona B	11/9	2048	1h	11008/7424/0/0
Zona C	14/9	16384	>2h	-
Zona D	24/2	16777216	>>2h	-

Tabela 8 – Resultados do Teste Exaustivo da MCE do subsistema de Fogo e Gás

OK+, 22 OK-, 0 SF and 0 DF) em 66s de execução. Já para o teste exaustivo, onde foram testadas todas as combinações de valores entre as entradas, foram executados 256 casos de teste e apresentados 1024 diagnósticos (942 OK+, 82 OK-, 0 SF and 0 DF) em 523s.

Pode-se observar que o número de casos de teste e o tempo de execução do teste cresce linearmente com o número de causas através do método proposto nesse trabalho, diferentemente do teste exaustivo que apresenta o crescimento exponencial. Durante todo o experimento, foi possível diagnosticar algumas inconformidades entre o projeto original e a implementação. Duas inconformidades ficaram evidentes a partir do diagnóstico de cinco (5) falhas seguras encontradas em uma das zonas. A Matriz da Figura 33 serve como base para explicação das inconformidades encontradas.

O primeiro problema está relacionado com a representação da função de segurança de detecção de fogo na matriz causa e efeito. O efeito **FD-100** é representado também como uma causa, e não existe uma identificação (que pode ser feita por um X) representando essa correlação. Assim, quando é ativado o sinal de causa, ativa-se automaticamente o efeito, mesmo que não tenha uma relação definida na matriz. Assim o oráculo entende como um sinal falso positivo.

A segunda inconformidade encontrada está relacionada com a votação do grupo de sensores. Durante o teste, o efeito **FC-100** sempre foi ativado quando um dos detectores de fumaça ficou ativo(1003), porém na especificação esse deveria ser ativado somente com a ativação de dois detectores. Dessa forma, caso um dos sensores apresentasse um problema o alarme de fogo confirmado seria soado, ocasionando uma parada da planta e início de procedimentos de emergência. Assim fica evidente a importância do teste do SIS e diagnóstico das falhas seguras e perigosas da execução da lógica de segurança.

CAUSA			EFEITO			
			Equipamento	TAG	FD-100	FC-100
Equipamento	Votação	TAG	Fire Detected	Fire Confirmed		
Smoke Detector A	1003	UST-500A	X			
Smoke Detector B	2003	UST-500B		X		
Smoke Detector C		UST-500C				
...	...	...				
Heat Fire Detector		TST-400	X	X		
Fire Detected		FC-100		X		
...	...	...				

Figura 33 – Causas/Efeitos da MCE com falhas seguras

## 5.4 CONCLUSÕES DO CAPÍTULO

A implementação do método exigiu o desenvolvimento de uma ferramenta computacional de testes para aplicação em um caso real da indústria. Foi demonstrado que a realização de um teste automatizado considerando como entrada a especificação lógica de segurança, ajuda a localizar as falhas provenientes de erros no código Ladder na implementação. A técnica de geração de casos de teste e os modelos de oráculos oferecem uma boa cobertura do diagnóstico das falhas existentes no comportamento de execução do dispositivo e favorecem a identificação dos erros que causaram essas falhas.

## 6 CONCLUSÃO

Nessa dissertação foi apresentado um método para preparar e executar um Teste automatizado do Controlador Lógico Programável do Sistema Instrumentado de Segurança de uma plataforma de petróleo.

As principais etapas do método compreendem a geração de casos de teste e de oráculos para diagnóstico do teste. Essas etapas tomam como base o documento da especificação, conhecido como Matriz de Causa e Efeito. Assim foi definida uma estratégia para seleção de casos de teste que utiliza uma abordagem caixa-preta e que abrange diferentes situações para o teste das lógicas do SIS.

Ainda, essas lógicas podem ser bem representadas por modelos de Redes de Petri, que favorecem a representação de estados (ex: causa ativa, efeito ativo) e eventos (ex: iniciar teste, identificação de falha perigosa), favorecendo assim o diagnóstico de falhas ou de atendimento da especificação.

Uma ferramenta computacional foi desenvolvida para servir como alternativa ou solução para automatização do teste. Essa implementa o método proposto e permite avaliar o atendimento do comportamento do dispositivo sob teste com as funções de segurança descritas no documento da especificação do sistema. Ainda, é possível por meio dessa avaliação, criar uma versão *as-built* da documentação da especificação quando o sistema já está operando.

São vistos como benefícios da utilização do método e da ferramenta desenvolvida: o aumento da confiabilidade de execução do dispositivo da lógica de controle, o melhor aproveitamento do tempo e recursos durante o TAF (Teste de Aceitação de Fábrica) e a padronização da documentação seguindo a norma interna da empresa.

A utilização do protocolo de comunicação OPC favoreceu a aplicabilidade da ferramenta para diversos fornecedores de CLP. Foi possível testar a lógica de segurança da plataforma de Petróleo da P-51 e ainda iniciaram-se testes da lógica de segurança de um floteador de outra plataforma de Petróleo. Esse segundo que possuía um CLP de um fabricante.

O trabalho apresentado em Prati (2014) serviu como ponto de partida para o desenvolvimento do método de testes e das etapas de geração de casos de teste e diagnóstico de inconformidades. As principais contribuições desse novo trabalho abrangem a proposição de uma nova estratégia para geração de casos de teste, a proposição de modelos de oráculos que permitem o diagnóstico de falhas seguras e perigosas (sinais falsos positivos e sinais falsos negativos) e ainda a validação do

método em um caso real da indústria.

O desenvolvimento da pesquisa desse trabalho gerou ainda publicações em Veiga et al. (2017a) e Veiga et al. (2017b).

## 6.1 LIMITAÇÕES E TRABALHOS FUTUROS

A abordagem caixa-preta não explora a estrutura interna do código para geração do teste e não avalia o histórico de falhas que já foram detectadas. Isso limita a cobertura de casos de teste, ou seja, os casos gerados são todos baseados no conhecimento da especificação.

Em vista dessa limitação, o testador automático possui uma funcionalidade que oferece uma opção para ativar entradas ou causas da matriz. A definição dessas entradas fica a critério do testador humano. Também foi oferecida a possibilidade de importação de um arquivo contendo uma lista de casos de teste que pode ser gerado por um outro programa que utiliza um algoritmo caixa-branca.

A primeira dificuldade encontrada durante a execução dos testes do CLP, refere-se a associação da TAG do sinal da causa ou efeito presente na matriz com o endereço de memória do CLP. Caso essa informação não esteja a disposição, o processo de automatização do teste é dificultado. Nesse caso é necessário varrer ou exportar uma lista de endereços de memória do CLP e associar esses endereços as causas e efeitos da MCE. Outra dificuldade refere-se com a presença de variáveis de reset, reconhecimento de alarmes na lógica do sistema e essas informações ficam implícitas na MCE.

A implementação prática dos testes está limitada à execução de um teste de uma matriz contendo 50 causas e 50 efeitos. Ainda não foram realizados testes considerando um caderno que possui diversas matrizes interrelacionadas. Essa análise é importante quando existem relações entre matrizes, onde um sinal de causa ou efeito de uma Matriz A está relacionado com um sinal de causa ou efeito de uma Matriz B.

O método ainda pode ser aplicado em outras indústrias que possuem sistemas críticos responsáveis pelas funções lógicas de segurança da planta.



## REFERÊNCIAS

- ADRION, W. R.; BRANSTAD, M. a.; CHERNIAVSKY, J. C. Validation, Verification, and Testing of Computer Software. **ACM Computing Surveys**, v. 14, p. 159–192, 1982.
- BERTHOMIEU, B.; DIAZ, M. Modeling and verification of time dependent systems using time Petri nets. **IEEE transactions on software**, 1991.
- DIAS, L.; PERKUSICH, A. On the automatic generation of timed automata models from function block diagrams for safety instrumented systems. **IEEE Transactions on Industrial Informatics**, 2008.
- DURAN, J. W.; NTAFOSS, S. C. An Evaluation of Random Testing. **IEEE Transactions on Software Engineering**, 1984.
- ELMENDORF, W. R. Automated Design of Program Test Libraries. **IBM Technical Report**, 1970.
- ET-3000.00-1200-800-PGT-006. Project Guidelines for the Confection of Cause and Effect Matrixes and Logic Diagrams. 2000.
- FREY, G.; LITZ, L. Formal methods in PLC programming. **2000 IEEE International Conference on Systems, Man and Cybernetics**, 2000.
- GRUHN, P.; CHEDDIE, H. L. Safety instrumented systems - design, analysis, and justification. ISA, 2006.
- HAMLET, D. Software Process, and Software Testing. **Academic Press, Advances in Computers**, 1994.
- HOWDEN, W. Functional Program Testing. **IEEE Transactions on Software Engineering**, SE-6, p. 162–169, 1980.
- IEC61511. Functional Safety: Safety Instrumented Systems for the Process Industry Sector. 2003.
- J. BELINFANTE, A. G. Automatic Testing with Formal Methods. **Centre for Telematics and Information Technology University of Twente**, 1999.

JORGENSEN, P. Software Testing: A Craftsman's Approach. 2002.

LEE, D.; YANNAKAKIS, M. Principles and methods of testing finite state machines-a survey. **Proceedings of the IEEE**, 1996.

MALEKZADEH, M.; AINON, R. N. An automatic test case generator for testing safety-critical software systems. **2010 The 2nd International Conference on Computer and Automation Engineering, ICCAE 2010**, v. 1, p. 163–167, 2010.

MEINKE, K.; SINDHU, M. A. LBTest: A learning-based testing tool for reactive systems. **Proceedings - IEEE 6th International Conference on Software Testing, Verification and Validation, ICST 2013**, 2013.

MOGYORODI, G. E.; MATH, B. Requirements-Based Testing - Cause-Effect Graphing Principal Consultant Software Testing Services. 2010.

MURATA, T. Petri nets: Properties, analysis and applications. **Proceedings of the IEEE**, v. 77, n. 4, p. 541–580, Apr 1989. ISSN 0018-9219.

MYERS, G. J.; THOMAS, T. M.; SANDLER, C. The Art of Software Testing 3rd Edition. v. 1, p. 255, 2011.

N-1883. Petrobras Internal Standard 1883, Project Instrumentation Presentation. 2002.

N-2595. Petrobras Internal Standard 2595, Specification for Project and Maintenance of Safety Instrumented Systems in Industrial Unities. 2012.

NIDHRA, S. Black Box and White Box Testing Techniques - A Literature Review. **International Journal of Embedded Systems and Applications**, v. 2, 2012.

OLIVEIRA, K. de V. Geração automática de testes de conformidade para programas de controladores lógico programáveis. **Dissertação submetida a Coordenação de Pós-Graduação em Informática da Universidade Federal de Campina Grande**, 2009.

PARADKAR, A.; TAI, K. C. Test generation for boolean expressions. **Software Reliability Engineering. Proceedings. Sixth International Symposium on. IEEE**, 1995.

PARADKAR, A.; TAI, K. C.; VOUK, M. A. Specification-based testing using cause-effect graphs. **Annals of Software Engineering**, v. 4, p. 133–157, 1997.

PEIXOTO, J. R. Gungnir - ferramenta para geração e execução automática de testes de conformidade utilizando Autômatos Temporizados . **Dissertação submetida ao curso de Pós-Graduação em Modelagem Computacional de Conhecimento do Instituto de Computação da Universidade Federal de Alagoas**, 2010.

POMMEREAU, F. Quickly Prototyping Petri Nets Tools with {SNAKES}. **Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops**, 2008.

PRATI, T.; FARINES, J. M.; QUEIROZ, M. H. Automatic test of safety specifications for plc programs in the oil and gas industry. **IFAC**, 2015.

PRATI, T. J. Desenvolvimento de uma metodologia para realização de testes em programas de clp na indústria de petróleo e gás. **Dissertação de Mestrado em Engenharia de Automação e Sistemas - PPGEAS**, 2014.

PROVOST, J.; ROUSSEL, J. M.; FAURE, J. M. Translating Grafcet specifications into Mealy machines for conformance test purposes. **Elsevier**, 2011.

PROVOST, J.; ROUSSEL, J.-M.; FAURE, J.-M. Generation of single input change test sequences for conformance test of programmable logic controllers. **IEEE Transactions on Industrial Informatics**, 2014.

SAITO, G. Desenvolvimento de ferramenta para especificação dos intertravamentos de segurança para clp na indústria de petróleo e gás. **Trabalho de Conclusão de Curso - Graduação em Engenharia de Automação e Sistemas - Universidade Federal de Santa Catarina**, 2016.

SELIC, B. A view on adoption of model-based methods in practice. **Software and Systems Modeling**, 2012.

SINDHU, M. A. Algorithms and Tools for Learning-based Testing of Reactive Systems. **Doctoral Thesis**.

SKOGDALEN, J. E.; SMOGELI, O. Looking forward-reliability of safety critical control systems on offshore drilling vessels. **Working Paper of Deepwater Horizon Study Group**, January 2011.

SRIVASTAVA, P. R.; PATEL, P.; CHATROLA, S. Cause effect graph to decision table generation. **ACM SIGSOFT Software Engineering Notes**, 2009.

UTTING, M.; LEGEARD, B. Practicalmodel-based testing: A tools approach. Morgan Kaufmann Publishers Inc., 2006.

VEIGA, H. et al. Automatic Conformance Testing of Safety Instrumented Systems for Offshore Oil Platforms. **Critical Systems: Formal Methods and Automated Verification: Joint 22nd International Workshop on Formal Methods for Industrial Critical Systems and 17th International Workshop on Automated Verification of Critical Systems, FMICS-AVoCS 2017, Turin, Italy**, Springer International Publishing, p. 51–65, 2017.

VEIGA, H. et al. Teste Automatico de Especificacoes de Seguranca em Matriz de Causa e Efeito para Controladores Logicos Programaveis de Plataformas Offshore. **5 Congresso de Instrumentacao, Controle e Automacao da Petrobras**, 2017.

WU, S.; MALAIYA, Y.; JAYASUMANA, A. Antirandom Testing: Beyond Random Testing. **Technical Report**, 1998.

## **ANEXO A – Template da MCE utilizado pela Petrobras**



