METODOLOGIA PARA DESENVOLVIMENTO ASSISTIDO POR MODEL CHECKING DE SISTEMAS DE CONTROLE E MONITORAÇÃO DE NAVIOS

FELIPE DA S. LÁZARO[*], MAX H. DE QUEIROZ[†], JEAN-MARIE FARINES[†]

[*] *Instituto de Pesquisas da Marinha – IPqM, Marinha do Brasil – MB, 21931-090, Rio de Janeiro, RJ, Brasil*

[†] *Departamento de Automação e Sistemas – DAS, Universidade Federal de Santa Catarina – UFSC 88040-900, Florianópolis, SC, Brasil*

Emails: felipe.engenharia@gmail.com, {max.queiroz, j.m.farine}@ufsc.br

Abstract— The Brazilian Navy develops the Control and Monitoring System for several ships in its fleet. This system is responsible for controlling the ship's main and auxiliary engines and is essential for its operation and safety. This work proposes a project development methodology for the Brazilian Navy Control and Monitoring System, using formal verification by model checking. The methodology establishes specification documents and translation methods for the language interpreted by the formal verification tool. Thus, making it possible to certify in an exhaustive and automatic way that all the essential safe properties to the project are included in the code specification document for the Programmable Logic Controller (PLC), generating cost reduction in error correction, and increasing reliability and availability of the ship. Because it is automatic, the method eliminates the need for designers to have formal verification skills. A typical Brazilian Navy's case study was used and confirmed the feasibility and efficacy of the proposed methodology.

Keywords— Model Checking, FIACRE, Formal Verification, Ships, Binary Logic Diagrams

Resumo— A Marinha do Brasil (MB) desenvolve o Sistema de Controle e Monitoração (SCM) para diversos navios de sua esquadra. Este sistema é responsável pelo controle das máquinas principais e auxiliares do navio, sendo essencial para sua operação e segurança. Neste trabalho é proposta uma metodologia de desenvolvimento de projetos para o SCM da MB, utilizando verificação formal por *model checking*. A metodologia estabelece documentos de especificação e métodos de tradução para a linguagem interpretada pela ferramenta de verificação formal. Assim, possibilitando certificar, de forma exaustiva e automática, que todas as propriedades de segurança essenciais ao projeto estão constando no documento de especificação do código para o Controlador Lógico Programável (CLP), gerando redução de custos na correção de erros, e aumentando a confiabilidade e disponibilidade do navio. Por ser automático, o método dispensa a necessidade dos projetistas terem conhecimentos em verificação formal. Um estudo de caso típico da MB foi utilizado e confirmou a viabilidade e eficácia da metodologia proposta.

Palavras-chave— Model Checking, FIACRE, Verificação Formal, Navios, Diagramas de Lógica Binária

1 Introdução

O Sistema de Controle e Monitoração (SCM) desenvolvido pela Marinha do Brasil (MB) é o sistema responsável por comandar e monitorar as máquinas principais do navio e seus sistemas auxiliares, além de adicionar estratégias de segurança de acordo com as especificidades de cada navio da MB. O SCM utiliza Controladores Lógico Programáveis (CLP) para atender à grande complexidade envolvida no projeto e aos diversos requisitos de segurança. Erros na especificação e programação do CLP podem gerar danos catastróficos à tripulação e ao navio, além de poder impedir que o navio efetivamente possa cumprir sua missão, em um resgate ou até a atuação em conflitos.

Segundo Gergely, Coroiu e Ponpetiu-Vladicescu (2011), em projetos envolvendo CLPs, os erros são introduzidos principalmente nas etapas de projeto conceitual e programação, e em sua maioria só são detectados na etapa de testes no sistema. Além disso, o custo de correção de erros aumenta consideravelmente quanto mais tardiamente ocorrer a detecção do erro.

Uma forma de viabilizar a detecção de erros nas etapas iniciais do projeto do SCM pode ser obtida com a validação das descrições das especificações através de verificação por métodos formais já na etapa de projeto conceitual. Métodos formais se baseiam em formalismos matemáticos, garantindo exatidão e reduzindo ambiguidades, erros e inconsistências. A IEC 61508 (2010), que trata de segurança funcional, prevê o uso de métodos formais em sistemas como o SCM.

Model checking é uma técnica automática para verificação e validação formal de sistemas concorrentes de estados finitos (Clarke; Emerson; Sistla, 1986), tem como principais vantagens ser um processo de verificação exaustivo, automático e que apresenta contraexemplo caso a propriedade desejada não estiver satisfeita, facilitando a correção (Moon, 1994).

Na literatura são encontrados diversos trabalhos voltados para a modelagem e verificação formal de programas de CLP (Moon, 1994)(Frey e Litz, 2000) (Mokadem et al., 2010) (Farines et al., 2011), porém poucos são os trabalhos envolvendo verificação da especificação do código para o CLP. Entre eles podem ser citados os trabalhos de Oliveira et al. (2010), onde foi utilizado a ferramenta Uppaal (Bengtsson et al., 1995) para realizar a verificação de Diagramas de

Lógica Binária (DLB) (ISA 5.2, 1992) e programas implementados em diagrama ladder, e Barbosa et al. (2007) que apresentou algoritmos para extração automática de autômatos temporizados a partir de DLBs para também serem verificados pela ferramenta Uppaal.

Este artigo apresenta um método para desenvolvimento de SCM assistido por *model checking* adaptado às práticas da MB, com foco na etapa de projeto conceitual, verificando se a especificação do código satisfaz as propriedades de segurança. Diferentemente dos trabalhos encontrados na literatura sobre *model checking* em projetos envolvendo CLP, o presente trabalho se caracteriza por: utilização de *model checking* em novo domínio de aplicação; sistematização de especificações de segurança para geração automática de propriedades em lógica temporal; sistematização de especificações do código para o CLP para geração automática de modelos formais; aplicação da cadeia de verificação TINA-FIACRE (Berthomieu et al., 2006) para validação de DLBs; estratégia de abstração sistemática para lidar com a complexidade computacional do modelo formal.

A seção 2 apresenta a metodologia atual de desenvolvimento de projetos de SCM da MB, a seção 3 propõe uma nova metodologia assistida por *model checking*, a seção 4 aborda as especificidades da verificação formal utilizada na nova metodologia, incluindo uma aplicação real da MB e estratégias para reduzir o problema de explosão combinacional, e a seção 5 apresenta as conclusões do trabalho.

2 A metodologia de projetos do SCM

A metodologia atualmente utilizada pela MB para projeto do SCM (Figura 1), baseia-se na elaboração do código do CLP a partir da especificação operativa do SCM, onde constam informações em linguagem natural e fluxogramas das funcionalidades desejadas, requisitos operacionais e de segurança, e da própria experiência da equipe especificadora do código do CLP. A especificação do código do CLP é apresentada como um pseudocódigo, documento sem padronização, que é entregue à equipe de programação do CLP para o interpretar e gerar o código para o CLP. Com o código para o CLP elaborado, é realizado o plano de testes, que são os testes de aceitação de fábrica (TAF), testes de aceitação de porto (TAP) e os testes de aceitação de mar ou rio (TAM/TAR). Caso seja encontrado qualquer tipo de inconformidade em quaisquer testes realizados, são necessárias correções no código do CLP. Após a realização de todos os testes, caso não ocorra a persistência de erros, o navio estará apto para navegar com o SCM.

As limitações do método atual estão na falta de um documento adequado para especificar o código para o CLP, a ausência de uma representação exclusiva para as especificações de segurança e a ausência de um método exaustivo, previsto pela IEC 61508 (2010), que garanta que o que foi especificado realmente contém todas os requisitos de segurança desejados.

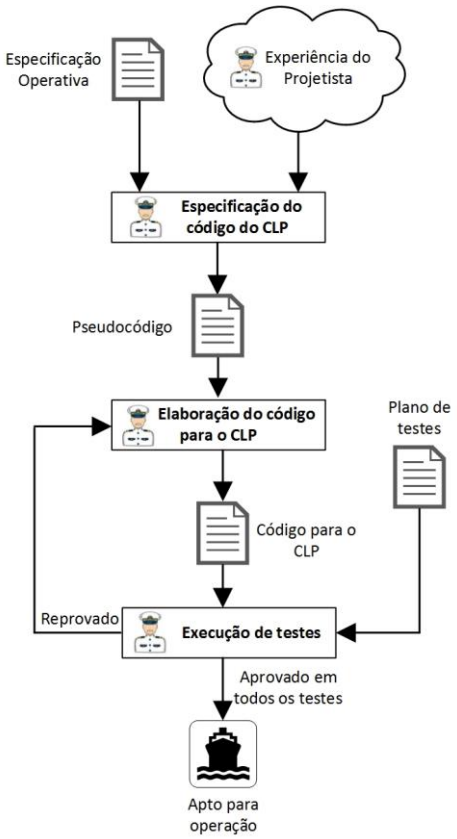

Figura 1. Metodologia atual dos projetos de SCM.

3 Proposta de metodologia para desenvolvimento de SCM assistido por *model checking*

A proposta de metodologia assistida por *model checking* para desenvolvimento de SCM da MB visa suprir as limitações da metodologia atualmente utilizada. Consiste na criação de documentação padronizada para especificação do código para o CLP (seção 3.1) e para as propriedades de segurança do projeto (seção 3.2). A Figura 2 apresenta a nova proposta de metodologia, onde o pseudocódigo atualmente utilizado para especificar o código para o CLP é substituído por um documento padronizado pela ISA 5.2 (1992), que são os DLBs. Os DLBs têm como função fornecer informações para controle, intertravamento para partida, operação, alarme e bloqueio de equipamentos e processos nos vários segmentos industriais, com uniformidade de símbolos e regras para a elaboração de documentos, como a especificação do código para o CLP.

As propriedades de segurança do projeto são apresentadas em uma Matriz de Causa e Efeito (MCE), que é uma forma de representar os requisitos de segurança dos sistemas com detalhes e de forma concisa, através de lógicas simbolizadas em suas linhas e colunas, além de permitir a visualização das

relações entre sistemas e subsistemas. O *model checking* verifica se todas as propriedades de segurança estão contidas no documento de especificação do código para o CLP, e caso o resultado do *model checking* for falso, o erro deve ser identificado e a especificação do código para o CLP deve ser corrigida antes da implementação.

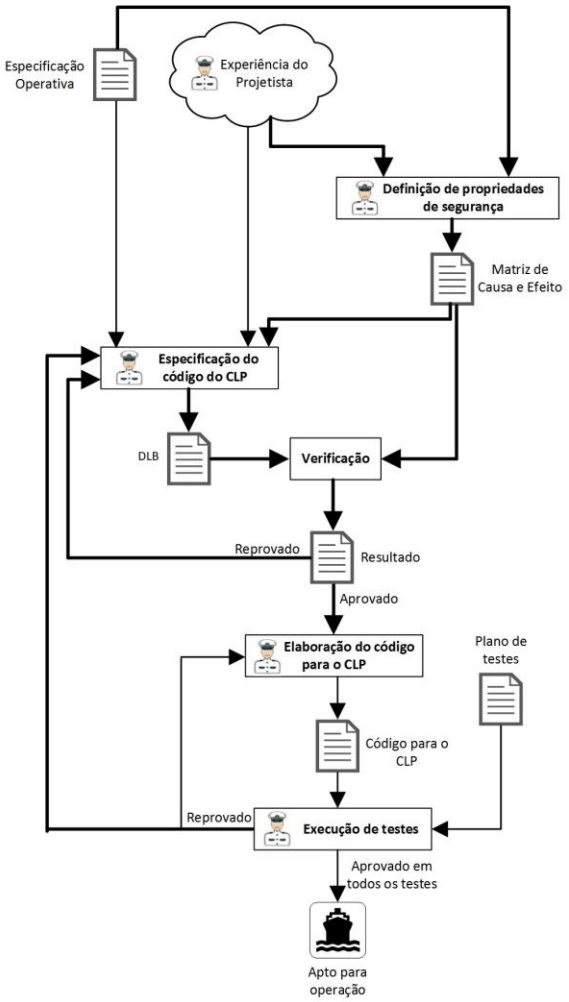

Figura 2. Proposta de metodologia para os projetos de SCM, incluindo *model checking*.

3.1 Especificação do código do CLP

A especificação do código do CLP apresentada na forma de DLB facilita o entendimento e a comunicação entre os programadores e projetistas, pois utilizam elementos lógicos básicos (lógicas *and*, *or*, *set-reset*, *not* e *timers*).

Um exemplo típico de DLB encontrado nos projetos de SCM da MB está apresentado na Figura 3, que é o procedimento para parada em emergência do motor principal de um navio. O DLB apresenta que a demanda forçada (DEMANDA_FORCADA_MCP) no Motor de Combustão Principal (MCP) permanecerá ativa se ao menos uma vez, simultaneamente, o botão de parada de emergência for pressionado, a Engrenagem Reversora (ER) não estiver desacoplada (ER_DESACOPLADA) e se a demanda estiver acima de 600rpm (DEMANDA>600) e será desativada se o MCP estiver na condição parado (MCP_OFF). Além disso, permanecerá em modo parando (MCP_STOPPING) se ao menos uma vez, simultaneamente, o botão de parada de emergência (PRDA_EMERG) for pressionado, a ER estiver desacoplada e se for forçada uma demanda acima de 600rpm por no mínimo 3 segundos, e será desativado se o MCP encontrar-se na condição parado. Apresenta também, que o MCP estará parado se a condição parando estiver ativa e a rotação chegar a zero (ROTACAO_MCP_=0).

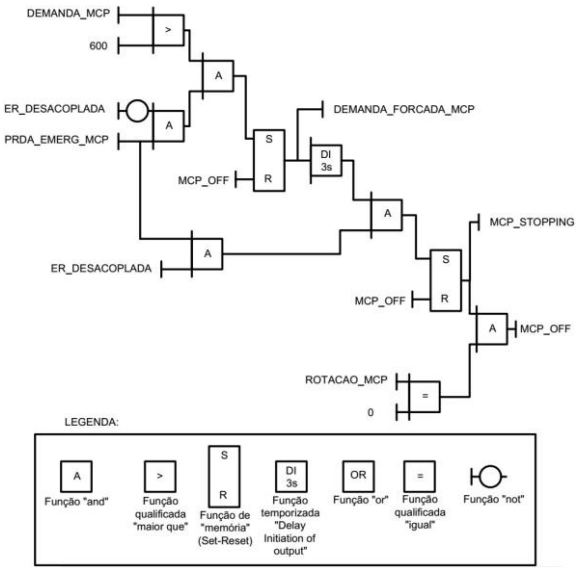

Figura 3. DLB da parada de emergência do MCP.

3.2 Representação de propriedades de segurança

As propriedades de segurança representadas na forma de MCE também utilizam elementos lógicos básicos, o que facilita sua interpretação por parte dos projetistas e programadores. A MCE é uma forma de representação matricial com as causas (entradas) nas linhas, os efeitos (saídas) nas colunas e símbolos padronizados nas entradas da matriz para especificar a lógica que relaciona as causas aos efeitos. A simbologia proproposta para ser utilizada na MCE da MB, que é uma extensão da utilizada pela Petrobras (Petrobras, 2005) e que também representa suas especificações de segurança na forma de MCE, é a seguinte:

X : Função *or*;
NX : Função *not or*;
A*i* : Função *and*;
NA*i* : Função *not and*;
} $i \in \mathbb{N}$
T*n* : Função de atraso temporal; *n* = tempo
\+ : Função de ativação do efeito; e
\- : Função de desativação do efeito.

A coluna AI apresenta os sinais analógicos existentes e a coluna DI segmenta os sinais discretos. A coluna *voting* é o campo para explicitar caso de votações, a coluna *TAG* identifica a *tag* de cada sinal, e o campo *NOTES* apresenta eventuais relações não representáveis na MCE ou de complexidade alta.

A MCE (Tabela 1) apresenta as propriedades de segurança para o procedimento de parada de emergência do motor principal de um navio.

Tabela 1. MCE das especificações de segurança para o procedimento de parada em emergência.

			TAG					
AI	**DI**	**VOTING**	**TAG**		**DEMANDA_FORCADA_MCP**	**MCP_STOPPING**	**MCP_OFF**	**NOTES**
ER	**DESACOPLADA**			**1**	NA1+	A1+		
	PRDA_EMERG_MCP			**2**	A1+	A1+		
DEMANDA_MCP	**>600**			**3**	A1+			
ROTACAO_MCP	**=0**			**4**			A1	
	MCP_OFF			**5**	X-	X-		
	DEMANDA_FORCADA_MCP			**6**		T3+		
	MCP_STOPPING			**7**			A1	

A MCE (Tabela 1) apresenta que o efeito DEMANDA_FORCADA_MCP) deve ser ativado sempre que, simultaneamente, a ER estiver acoplada, o botão de parada for pressionado e a demanda do MCP estiver acima de 600rpm. Deve ser desativado quando MCP_OFF estiver ativo. O efeito MCP_STOPPING deve ser ativado sempre que, simultaneamente, a ER estiver desacoplada, o botão de parada de emergência for pressionado e a demanda estiver em modo forçado por, no mínimo, três segundos. Deve ser desativado quando MCP_OFF estiver ativo. O efeito MCP_OFF deve permanecer ativo enquanto a rotação do MCP estiver igual a zero e o motor estiver em modo parando.

4 Verificação formal

A verificação por *model checking* consiste na comparação de um modelo matemático da especificação do código para o CLP escrito em DLB com fórmulas de lógica temporal que representem as propriedades de segurança apresentadas em MCE. Então, alguns procedimentos são necessários para viabilizar o *model checking*, como apresentar as especificações de segurança e do código para o CLP em uma linguagem interpretável por uma ferramenta verificadora.

A linguagem FIACRE (Berthomieu et al., 2008) foi projetada para ser uma linguagem intermediária formal de alto nível entre as linguagens de modelagem de alto nível e as ferramentas de verificação. Por ser uma linguagem de alto nível e existirem ferramentas de tradução automática para linguagem formal, FIACRE foi a linguagem intermediária escolhida para ser utilizada na metodologia.

FIACRE é construída em torno das noções de processos e componentes. Processos correspondem a máquinas de estados, em que as transições entre os estados definem o comportamento do processo. Componentes descrevem a composição dos processos de forma hierárquica, e é definido como uma composição paralela de componentes e/ou processos que se comunicam através de portas e variáveis compartilhadas.

Por ser capaz de interpretar a linguagem formal gerada pelas ferramentas de tradução automática de FIACRE para linguagem formal, a ferramenta verificadora TINA/SELT (Berthomieu et al., 2006) foi escolhida para ser utilizada na metodologia. A ferramenta é um *State/Event* LTL *model checker* e permite analisar modelos temporais.

A Figura 4 apresenta as etapas de tradução tanto dos DLBs quanto da MCE. As propriedades extraídas da MCE são representadas em fórmulas em Lógica Temporal Linear (LTL) (Pnueli, 1977) e declaradas em FIACRE.

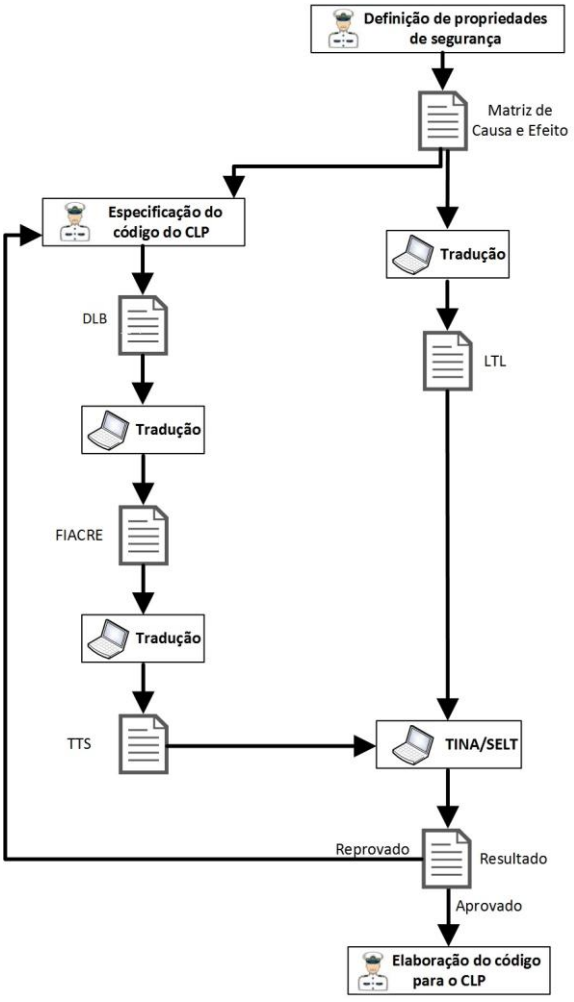

Figura 4. Cadeia de verificação proposta no método.

As lógicas temporais especificam propriedades em verificação de modelos, pois são capazes de expressar relações de ordem, sem a noção explícita de tempo. A lógica temporal usa proposições atômicas para fazer afirmações sobre os estados, as quais, em um dado estado, possuem um valor verdadeiro bem definido. A ferramenta TINA utiliza um subconjunto de LTL para verificação formal.

Após a tradução dos DLBs para FIACRE, o código é compilado pela ferramenta FRAC (Berthomieu et al., 2006), onde o modelo em FIACRE é transformado em um Sistema de Transição Temporizado (TTS). Posteriormente, a ferramenta verificadora TINA/SELT confronta as propriedades de segurança formuladas em LTL com a especificação do código do CLP em TTS. Em caso de resultado falso o SELT fornece um contraexemplo, que na metodologia proposta é apresentado na forma de diagrama de sinais, facilitando a identificação e posterior correção do erro. Em caso de resultado verdadeiro, os DLBs podem seguir para a etapa seguinte de elaboração do programa do CLP.

4.1 Tradução de DLBs para FIACRE

Diferente da linguagem FBD (*Function Block Diagrams*), que é uma linguagem de programação de CLPs, em que seu ciclo de execução é o próprio *scan cycle* do CLP, o DLB não é uma linguagem de programação. O fluxo de leitura (*flow of intelligence*) dos DLBs é normatizado pela ISA 5.2 (1992), que estabelece que deve ocorrer ordenamente da esquerda para a direita, e de cima para baixo. As operações devem ser calculadas sequencialmente porque o DLB especifica a lógica a ser implementada em CLPs, que funcionam em *scan cycles* para simular o comportamento concorrente de circuitos digitais através de um único processador computacional.

Para representar o comportamento especificado pela ISA 5.2, o fluxo de leitura é dividido em duas etapas: atualização das entradas e atualização das saídas. Primeiro, as variáveis de entrada são atualizadas. Posteriormente ocorre a etapa de atualização dos valores de saída, calculados pela aplicação das operações lógicas especificadas no DLB sobre os valores de entrada da etapa anterior, priorizando a atualização no sentido de cima para baixo. Feito isso, o fluxo de leitura pode recomeçar, caracterizando assim, um ciclo de leitura.

Para representar o padrão do ciclo de leitura em FIACRE são criados quatro estados (*UPDATE_INPUTS*, *UPDATES_OUTPUTS*, *PROBE* e *WAITING*). Para garantir que os valores nas saídas estejam sempre relacionadas com os valores das entradas ao final de cada ciclo, todas as transições entre estados são definidas como instantâneas (comando *wait*[0,0] em FIACRE), com exceção da transição de reinício de ciclo. O estado *PROBE* é o estado em que será realizado o *model checking* e que é posicionado após o estado de atualização do efeito que se deseja verificar, garantindo que todas as variáveis relacionadas ao efeito estão atualizadas no estado. O estado *WAITING* representa o reinício do ciclo de leitura dos DLBs, sendo sempre posicionado ao final do código. Como a norma não estabelece um padrão para o reinício do ciclo de leitura dos DLBs, a transição entre o estado *WAITING* e *UPDATE_INPUTS* foi modelada sem restrição temporal, de forma que a verificação formal considere todos os intervalos de tempo possíveis de reinício do ciclo de leitura. Assim, as propriedades validadas para o modelo independem do *scan cycle* do CLP escolhido para implementação da lógica especificada pelo DLB.

Seguindo a metodologia proposta, um trecho do código FIACRE da tradução do DLB da Figura 3, com o estado *PROBE* posicionado para verificar o efeito DEMANDA_FORCADA_MCP, está apresentado a seguir:

```
Process diagram
...
init to UPDATE_INPUTS
    from UPDATE_INPUTS
       ...
       to UPDATE_OUTPUTS0
     from UPDATE_OUTPUTS0
        DEMANDA_FORCADA_MCP:=((DEMANDA_FORCAD
        A_MCP or (DEMANDA_MCP_MAIOR_600 and
        not(ER_DESACOPLADA) and
        PRDA_EMERG_MCP)) and not(MCP_OFF));
        portDI0_IN! ((DEMANDA_FORCADA_MCP or
       to PROBE
     from PROBE
       wait[0,0];
       to UPDATE_OUTPUTS1
     from UPDATE_OUTPUTS1
       ...
       to WAITING
     from WAITING
       to UPDATE_INPUTS
```

Os operadores "!" e "?" (envio e recebimento) correspondem a comunicação síncrona entre processos e/ou componentes através das portas de comunicação, e pela concepção do FIACRE as transições com comunicação são tratadas como instantâneas.

4.2 Tradução da MCE para LTL

A MCE representando especificações de segurança consiste de ações (efeito) que devem ser tomadas quando determinadas condições (causa) são satisfeitas. Por exemplo, a abertura de uma válvula de segurança caso dois sensores de pressão alta estiverem ativos.

Dois tipos de falhas (IEC 61511-1, 2016) podem ser definidas para cada efeito para fins de organização das propriedades de segurança. Uma falha segura (FS) ocorre quando existe o efeito, mas não existe a causa. Por exemplo, a válvula de segurança abriria sem que os dois sensores de pressão alta estivessem ativos. Uma falha perigosa (FP) ocorre quando existe a causa, mas não existe efeito. Por exemplo, a válvula não seria aberta mesmo com os sensores de pressão alta ativos.

As propriedades verificadas com *model checking* são as propriedades livre de falha segura (LFS) e livre de falha perigosa (LFP) (Lázaro, 2018) (Reis, 2018). A verificação de propriedades LFS e LFP é efetuada em um estado específico (*PROBE*) do ciclo de leitura. Além disso, as propriedades são verificadas de forma individual para cada efeito da MCE. As fórmulas baseadas na sintaxe LTL das propriedades LFS e LFP estão apresentadas em (1) e (2). Na propriedade LFS se verifica se sempre não haverá, em um estado específico (*PROBE*), o efeito ativo e a causa desativada. Na propriedade LFP se verifica se sempre não haverá, em um estado específico (*PROBE*), o efeito desativado e a causa ativada.

$$LFS = []\neg(probe \wedge \neg causa \wedge efeito) \quad (1)$$

$$LFP = []\neg(probe \wedge causa \wedge \neg efeito) \quad (2)$$

Por vezes, causas podem ser formadas por lógicas que utilizam a simbologia de ativação e desativação do efeito. Nesse caso, a definição do valor atualizado da causa necessita de um passo intermediário para sua definição (3), pois estão relacionadas com o valor do efeito no ciclo de leitura anterior. Sendo assim, uma causa formada por lógicas de ativação e desativação é definida da seguinte forma:

$$causa = \left\{ \begin{array}{c} \left(efeito' \vee (\text{"}lógicas\ que\ ativam\ o\ efeito\text{"})\right) \\ \wedge \\ \neg(\text{"}lógicas\ que\ desativam\ o\ efeito\text{"}) \end{array} \right\} \quad (3)$$

Onde *efeito'* é o valor do efeito no ciclo anterior.

Exemplificando para o primeiro efeito (DEMANDA_FORCADA_MCP) da MCE da Tabela 1, a representação em LTL é a seguinte (4):

```
causa =((DEMANDA_FORCADA_MCP ∨
(¬(ER_DESACOPLADA) ∧ PRDA_EMERG_MCP ∧
DEMANDA_MCP_MAIOR_600)) ∧ ¬(MCP_OFF))
```
(4)

4.3 Redução por cone de influência

Com o objetivo de diminuir o espaço de estados e minimizar o problema de explosão combinacional, pode-se utilizar o método de redução por cone de influência (Clarke; Grumberg; Peled, 1999) (Darvas et al., 2014), que propõe realizar a verificação das propriedades de cada efeito da MCE utilizando somente as entradas que realmente influenciam no efeito desejado e descartando as variáveis que não influenciam. Tal estratégia possibilita redução no número de estados do sistema/subsistema cujo efeito se deseja verificar, minimizando o problema de explosão combinacional.

A Figura 5 apresenta o cone de influência da MCE da Tabela 1. Verifica-se que todas as variáveis estão relacionadas entre si, não podendo descartar nenhuma entrada, pois todas estão relacionadas ao efeito desejado.

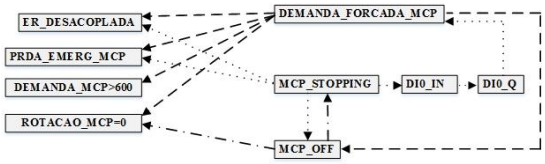

Figura 5. Cone de influência da MCE da Tabela 1.

4.4 Contraexemplo

A interpretação correta do contraexemplo fornecido é muito importante para agilizar possíveis correções no documento de especificação para o código do CLP, resultando nas vantagens de se identificar erros já na etapa de projeto conceitual (Gergely; Coroiu; Ponpetiu-Vladicescu, 2011).

Realizando *model checking* para o exemplo (Figura 3 e Tabela 1), obtêm-se os resultados apresentados na Tabela 2, todos com tempo de execução insignificantes, onde verifica-se a existência de uma falha perigosa na saída MCP_STOPPING e garante-se a ausência de falhas de segurança na especificação da lógica de CLP para as demais saídas.

Tabela 2. Resultados do estudo de caso.

Efeitos	Estados	Transições	FS/FP/OK
DEMANDA_FORCADA_MCP	**534**	**1948**	**0/0/2**
MCP_STOPPING	**544**	**2466**	**0/1/1**
MCP_OFF	**566**	**2748**	**0/0/2**
Legenda: FS = Falha Segura FP = Falha Perigosa OK = Propriedades aprovadas			

Como o contraexemplo gerado pelo SELT é apresentado de forma complexa para os programadores de CLPs, ele é convertido para uma forma compreensível em diagrama de sinais (Marques; de Queiroz; Farines, 2016). A Figura 6 apresenta o contraexemplo do SELT na forma de diagrama de sinais, possibilitando identificar que no oitavo ciclo existe a causa, mas não existe o efeito, configurando uma falha perigosa, sendo assim, o DLB está reprovado e deve ser corrigido para que não se tenha uma especificação para o código do CLP incorreta. Analisando a MCE da Tabela 1, verifica-se que ela determina que o efeito MCP_STOPPING deve estar ativo quando ER_DESACPL e PRDA_EMERG estiverem ativas ou a saída do temporizador estiver ativa, portanto, pelas condições no oitavo ciclo da Figura 6, o efeito deveria estar ativo, concluindo a existência de um erro na elaboração do DLB. No DLB da Figura 3 identifica-se que o erro foi gerado por uma lógica erradamente representada por uma lógica *and* após o temporizador DI, quando deveria ser *or*, pois no DLB o efeito é ativado quando ER_DESACPL, PRDA_EMERG e a saída do temporizador estiverem ativas simultaneamente.

O DLB corrigido foi submetido a *model checking*, confirmando que o novo DLB proposto está livre de falhas, atendendo às especificações de segurança e, portanto, está aprovado para ser o documento de especificação para o código do CLP.

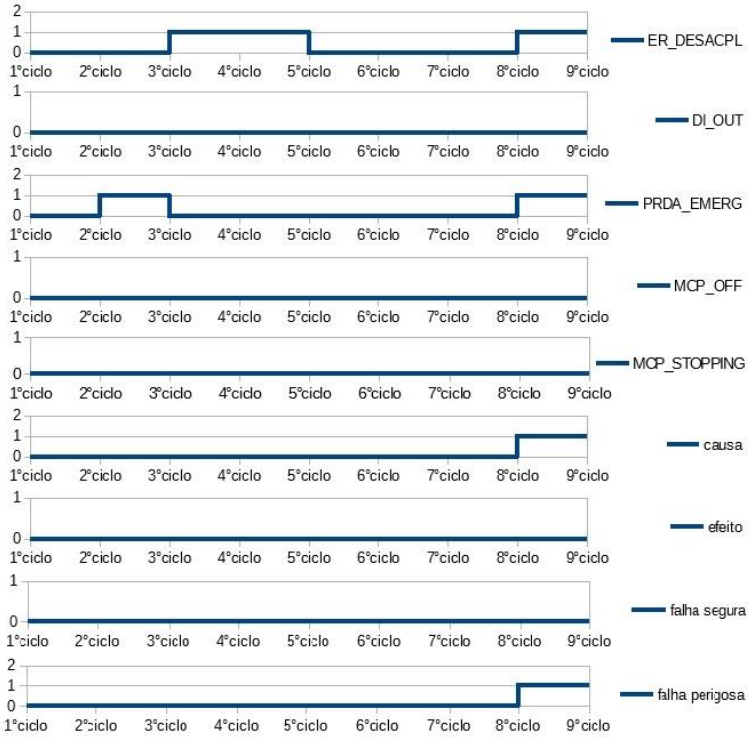

Figura 6. Apresentação do contraexemplo.

4.5 Redução por decomposição

Em alguns casos a estratégia de redução por cone de influência não é suficiente para reduzir o espaço de estados gerados e impedir explosão combinacional. Então, uma estratégia proposta é a redução por decomposição (Lázaro, 2018) (Reis, 2018), que se baseia na decomposição das propriedades a serem verificadas. A estratégia de redução por decomposição parte do princípio de que as lógicas de uma causa podem ser expressadas, de forma generalizada, por "*n*" combinações de lógicas *and* e "*m*" combinações de lógicas *or* (5). Pode-se utilizar (5) em (1), e (5) em (2), e após manipulações algébricas utilizando as leis de Morgan e a regra de equivalência para LTL, obtêm-se (6) e (7) respectivamente.

$$causa = \left(A_1 \wedge \ldots \wedge A_n\right) \vee \left(O_1 \vee \ldots \vee O_m\right) \qquad (5)$$
$$n, m \in \mathbb{N}$$

$$LFS = \left\{ \begin{array}{c} [] \left(\neg Efeito* \vee O_1 \vee O_2 \vee \ldots \vee O_m \vee A_1 \right) \\ \wedge \\ [] \left(\neg Efeito* \vee O_1 \vee O_2 \vee \ldots \vee O_m \vee A_2 \right) \\ \wedge \ldots \\ [] \left(\neg Efeito* \vee O_1 \vee O_2 \vee \ldots \vee O_m \vee A_n \right) \end{array} \right\} \qquad (6)$$

$$LFP = \left\{ \begin{array}{c} [] \left(Efeito* \vee \neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_n \right) \\ \wedge \\ [] \left(Efeito* \vee \neg O_1 \right) \\ \wedge \ldots \\ [] \left(Efeito* \vee \neg O_m \right) \end{array} \right\} \qquad (7)$$

Onde: $Efeito* = Efeito \wedge probe$

Sendo assim, a redução por decomposição possibilita a verificação das propriedades em subconjuntos, onde cada subconjunto consiste das lógicas precedidas pelo operador LTL "sempre" ([]) nas equações (6) e (7), reduzindo o problema de explosão combinacional pela diminuição do espaço de estados gerados no *model checking*, devido ao descarte das entradas não relacionadas a cada subconjunto.

As reduções por cone de influência e por decomposição são utilizadas sistematicamente e automaticamente a cada verificação, mesmo nos casos que não provoquem redução no espaço de estados gerados.

Outra aplicação real da MB foi submetida à metodologia, com seis DLBs e uma MCE de dimensão 43x20, gerando aproximadamente 500 mil estados e 1 milhão de transições para se garantir a ausência de falhas de segurança nos DLBs. Para esse caso de aplicação, as duas estratégias de redução foram essenciais para contornar o problema de explosão combinacional.

5 Conclusão

A padronização dos métodos de especificação tanto do código do CLP quanto das propriedades de segurança facilita a compreensão das especificações por todas as equipes envolvidas no projeto de SCM da MB, e a verificação por *model checking* possibilita a antecipação na detecção de erros já na etapa de especificação do projeto, gerando redução de custos e aumentando a confiabilidade do projeto. O estudo de caso confirmou que o método proposto é eficiente e viável de ser implementado nos projetos de SCM da MB. Novas técnicas de minimização do problema de explosão combinacional ainda são necessárias e serão motivo de estudos futuros.

Agradecimentos

Os autores são gratos à Marinha do Brasil e à Universidade Federal de Santa Catarina.

Referências Bibliográficas

Barbosa, L. P., Gorgônio, K., Lima, A. M. N., Perkusich, A., and Da Silva, L. D. (2007). On the automatic generation of timed automata models from isa 5.2 diagrams. *Emerging Technologies and Factory Automation. ETFA. IEEE Conference on*, IEEE, pp. 406-412.

Bengtsson, J., Larsen, K., Larsson, F., Pettersson, P., and Yi, W. (1995). UPPAAL - a tool suite for automatic verification of real-time systems. *International Hybrid Systems Workshop*. Springer, p. 232-243.

Berthomieu, B., and Vernadat, F. (2006). Time petri nets analysis with tina. *Quantitative Evaluation of Systems, 2006. QEST 2006. Third International Conference on*, pp. 123-124.

Berthomieu, B., Bodeveix, J.-P., Farail, P., Filali, M., Garavel, H., Gaufillet, P.,Lang, and Vernadat, F. (2008). Fiacre: an intermediate language for model verification in the topcased environment. *ERTS 2008*.

Clarke, E. M., Emerson, E. A., and Sistla, A. P. (1986). Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, **8**(2): 244-263.

Clarke, E. M., Grumberg, O., and Peled, D. (1999). *Model checking*. MIT press.

Darvas, D., Adiego, B. F., Vörös, A., Bartha, T., Viñuela, E. B., and Suárez, V. M. G. (2014). Formal verification of complex properties on PLC programs. *International Conference on Formal Techniques for Distributed Objects, Components, and Systems*, Springer, pp. 284-299.

Farines, J. M., de Queiroz, M. H., da Rocha, V. G., Carpes, A. M. M., Vernadat, F., and Crégut, X. (2011). A model-driven engineering approach to formal verification of PLC programs. *Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference*, IEEE, pp. 1-8.

Frey, G., and Litz, L. (2000). Formal methods in PLC programming. *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*. IEEE, 2000. Vol. 4. p. 2431-2436.

Gergely, E. I., Coroiu, L., and Popentiu-Vladicescu, F. (2011). Methods for Validation of PLC Systems. *Journal of Computer Science and Control Systems*, **4**(1): 47.

ISA 5.2, (1992). Binary Logic Diagrams for Process Operations – 1976 (R1992) edition, International Society of Automation.

IEC 61508, (2010). Functional safety of electrical/electronic/programmable electronic safety related systems. *International Electrotechnical Commission.*

IEC 61511-1, (2016). Functional safety - Safety instrumented systems for the process industry sector - Part 1: Framework, definitions, system, hardware and application programming requirements. *International Electrotechnical Commission.*

Lázaro, F. 2018, "Metodologia para desenvolvimento de Sistemas de Controle e Monitoração de navios assistido por model checking", Dissertação de Mestrado, Universidade Federal de Santa Catarina, Florianópolis, SC,

Marques, L. G. P. C., de Queiroz, M. H., and Farines, J. M. (2016). Improving a design methodology of synthesizable VHDL with formal verification. In Circuits & Systems (LASCAS), *IEEE 7th Latin American Symposium*, IEEE, pp.51-54.

Mokadem, H. B., Berard, B., Gourcuff, V., De Smet, O., and Roussel, J. M. (2010). Verification of a timed multitask system with UPPAAL. *IEEE Transactions on Automation Science and Engineering*, IEEE, **7**(4): 921-932.

Moon, I. (1994). Modeling programmable logic controllers for logic verification. *IEEE Control Systems*, IEEE, 14(2), 53-59.

Oliveira, K., da Silva, L. D., Perkusich, A., Lima, A. M. N., and Gorgônio, K. (2010). Automatic timed automata extraction from ladder programs for model-based analysis of control systems. *Industrial Electronics (ISIE), 2010 IEEE International Symposium*, IEEE, pp. 90-95.

Petrobras. (2005) Especificação Técnica para implementação da lógica de intertravamento e controle - ET-3000.00-5500-800-PCI-002REVB.

Pnueli, A. (1977). The temporal logic of programs. *In Foundations of Computer Science, 1977., 18th Annual Symposium on*, IEEE, pp. 46-57.

Reis, L. 2018, "Verificação formal de Sistemas Instrumentados de Segurança na indústria de petróleo e gás natural", Dissertação de Mestrado, Universidade Federal de Santa Catarina, Florianópolis, SC,