UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA – PPGEEL

HERBERT ALBÉRICO DE SÁ LEITÃO

METODOLOGIA DE PROJETO DE CONTROLE TOLERANTE A FALHAS DE SEDS ADERENTE À IEC 61499

JOINVILLE
2022

HERBERT ALBÉRICO DE SÁ LEITÃO

METODOLOGIA DE PROJETO DE CONTROLE TOLERANTE A FALHAS DE SEDS ADERENTE À IEC 61499

Tese apresentada ao Programa de Pós–Graduação em Engenharia Elétrica do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, como requisito parcial para a obtenção do grau de Doutor em Engenharia Elétrica.

Orientador: Prof. Dr. Roberto Silvio Ubertino Rosso Junior
Coorientador: Prof. Dr. André Bittencourt Leal

JOINVILLE
2022

de Sá Leitão, Herbert Albérico
Metodologia de Projeto de Controle Tolerante a Falhas de SEDs Aderente à IEC 61499 / Herbert Albérico de Sá Leitão. - 2022.
158 p.

Orientador: Prof. Dr. Roberto Silvio Ubertino Rosso Junior.
Coorientador: Prof. Dr André Bittencourt Leal.
Tese (Doutorado) - Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Engenharia Elétrica, Joinville, 2022.

1. Controle tolerante a falhas. 2. Metodologia de controle tolerante a falhas. 3. Diagnose de falhas. 4. Sistemas a eventos discretos. 5. IEC 61499. I. Rosso Junior, Roberto Silvio Ubertino . II. Leal, André Bittencourt . III. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Engenharia Elétrica. IV. Título.

HERBERT ALBÉRICO DE SÁ LEITÃO

METODOLOGIA DE PROJETO DE CONTROLE TOLERANTE A FALHAS DE SEDS ADERENTE À IEC 61499

Tese apresentada ao Programa de Pós–Graduação em Engenharia Elétrica do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, como requisito parcial para a obtenção do grau de Doutor em Engenharia Elétrica.

Orientador: Prof. Dr. Roberto Silvio Ubertino Rosso Junior
Coorientador: Prof. Dr. André Bittencourt Leal

BANCA EXAMINADORA:

Prof. Dr. Roberto Silvio Ubertino Rosso Junior
Universidade do Estado de Santa Catarina

Membros:

Prof. Dr. Carlos Eduardo Pereira
Universidade Federal do Rio Grande do Sul

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

Dr. Leandro Israel Pinto
IOTBRAS L. S. Ltda.

Prof. Dr. Yuri Kazubowski Lopes
Universidade do Estado de Santa Catarina

Joinville, 08 de dezembro de 2022

Dedico este trabalho aos meus familiares e amigos que sempre me apoiaram e me motivaram para o cumprimento dessa difícil jornada.

AGRADECIMENTOS

Agradeço aos meus pais, Jurandir e Maria Lêda, por tudo que sempre fizeram por mim e por me mostrarem a importância do trabalho e do estudo.

Agradeço ao meu orientador, Prof. Roberto Silvio Ubertino Rosso Junior, e ao meu coorientador, Prof. André Bittencourt Leal, por toda dedicação e ensinamentos. Profissionais que respeito, admiro e tenho uma grande gratidão por todo o apoio que me deram.

Por fim, agradeço ao Prof. Douglas Wildgrube, a todos os amigos do PPGEEL (Programa de Pós-Graduação em Engenharia Elétrica), em especial a Leo Schirmer, Renan Sebem, Thalys Rezende e Aureo Dobrikopf, e aos alunos bolsistas que trabalharam comigo. A todos vocês o meu muito obrigado por toda solidariedade, ajuda e presteza.

“Às vezes o nosso avanço começa quando nos recusamos a ficar impressionados com o tamanho do nosso problema.” (Bill Johnson)

RESUMO

Os sistemas industriais modernos são caracterizados por ações complexas que exigem grande integração e disponibilidade. Entretanto, são susceptíveis a falhas que afetam sua operação normal e acarretam em mudanças no seu comportamento. Essas falhas podem ocasionar danos de diversas naturezas ou até mesmo a interrupção completa do funcionamento do sistema. Neste cenário, é imprescindível tornar o sistema tolerante a falhas por meio da detecção, do diagnóstico e do tratamento das falhas, dando origem ao que se denomina de controle tolerante a falhas. Nesta tese, é apresentada uma metodologia de projeto de controle tolerante a falhas que possibilitou a implementação de um sistema de projeto de controle tolerante a falhas capaz de analisar a condição de diagnosticabilidade de falhas, desenvolver uma estrutura de controle tolerante a falhas e converter esta estrutura em lógica de controle. Este sistema de controle tolerante a falhas é um sistema computacional que opera de forma não assistida (sem dependência de ação humana), utilizando métodos formais para sistemas a eventos discretos na análise da diagnosticabilidade das falhas e no desenvolvimento da estrutura de controle tolerante a falhas. Além disso, o sistema converte a estrutura de controle tolerante a falhas desenvolvida em blocos de função da norma IEC 61499, o que facilita sua aplicação em sistemas industriais. O trabalho realizado nesta tese é parte integrante do projeto de uma arquitetura de reconfiguração para tratamento de falhas em sistemas industriais, denominada de *Reconfiguration Architecture for Fault Handling* (RAFAH), que se fundamenta no desenvolvimento de um sistema inteligente capaz de projetar controles tolerante a falhas e de implementar esses controles em sistemas industriais. Um processo de mistura de substâncias químicas é utilizado na tese como estudo de caso com o intuito de avaliar e validar a metodologia de controle tolerante a falhas proposta. Os resultados da avaliação se mostraram consistentes e apresentaram uma estrutura de controle capaz de reagir às falhas e de ter tomadas de decisões que garantem novas condições de controle para o sistema.

Palavras-chave: controle tolerante a falhas; metodologia de projeto de controle tolerante a falhas; diagnose de falhas; sistemas a eventos discretos; IEC 61499.

ABSTRACT

Modern industrial systems have the possibility of operating complex tasks as one of their features, demanding high integration and availability. However, they are susceptible to faults that affect their operation and change their behavior. These faults can result in many types of damages or even the interruption of the system. In this scenario, it is essential to make the system fault tolerant through detection, diagnosis and fault handling, giving rise to what is known as fault-tolerant control. This thesis presents a methodology for fault-tolerant control design as well as a system for fault-tolerant control design. The system for fault-tolerant control design is a computational system in which it is possible to analyze the diagnosability condition of faults, develop fault-tolerant control, and implement this fault-tolerant control in control logic. It performs tasks in an autonomous way, independent of human action, applying formal methods related to discrete event systems in order to analyze the diagnosability of faults and design the fault-tolerant control. Moreover, it is able to implement the designed fault-tolerant control in function blocks based on IEC 61499, which facilitates its application in industrial systems. The work carried out in this thesis is an integral part of the design of a reconfiguration architecture for fault handling in industrial systems, called RAFAH, which is based on the development of an intelligent system capable of designing fault-tolerant controls and implementing these controls in industrial systems. A case study based on a chemical mixing process is presented in order to evaluate and validate the proposed methodology of fault-tolerant control design. The evaluation results were consistent and presented a control model ables to react a faults and take decisions that guarantee continuity of action and flexibility to the operation of the system.

Keywords: fault-tolerant control; methodology of fault-tolerant control design; fault diagnosis; discrete event systems; IEC 61499.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADEF	Autômato Determinístico de Estados Finitos
BFB	*Basic Function Block* (Bloco de Função Básico)
CPME	Conjunto dos Primeiros Maus Estados
CEPC	Conjunto de Estados Passíveis de Controle
CFB	*Composite Function Block* (Bloco de Função Composto)
CTF	Controle Tolerante a Falhas
ECC	*Execution Control Chart* (Gráfico de Controle de Execução)
e.r.a	Em relação a
FAS	*Fault Analysis System* (Sistema de Análise de Falhas)
FB	*Function Block* (Bloco de Função)
FTCSS	*Fault Tolerant Control Synthesis System* (Sistema de Síntese de Controle Tolerante a Falhas)
FMEA	*Failure Mode and Effect Analysis* (Análise de Modos de Falhas e Efeitos)
IEC	*International Electrotechnical Commission*
RAFAH	*Reconfiguration Architecture for Fault Handling* (Arquitetura de Reconfiguração para Manipulação de Falhas)
SED	Sistema a Eventos Discretos
SIFB	*Service Interface Function Block* (Bloco de Função de Interface de Serviço)
SPA	Sistema de Produção Automatizado
TCS	Teoria de Controle Supervisório
XML	*eXtensible Markup Language* (Linguagem de Marcação Extensível)

LISTA DE SÍMBOLOS

$\exists$	existe
$\forall$	para todo
$\wedge$	operador lógico “e”
$\vee$	operador lógico “ou”
:	tal que
$\Rightarrow$	implica
$\mathbb{N}$	conjunto dos números naturais
$\mathbb{N}^*$	conjunto dos números naturais não nulos
ε	palavra vazia
Σ	conjunto ou alfabeto de eventos de um sistema a eventos discretos
Σ^*	fecho de Kleene ou conjunto de todas palavras finitas do alfabeto Σ, incluindo ε
Σ_c	conjunto de eventos controláveis
Σ_{nc}	conjunto de eventos não controláveis
Σ_o	conjunto de eventos observáveis
Σ_{no}	conjunto de eventos não observáveis
L	linguagem gerada por um sistema a eventos discretos
$L(G)$	linguagem gerada por G
$L_m(G)$	linguagem marcada por G
L^*	fecho de Kleene da linguagem L
$\overline{L}$	prefixo-fechamento da linguagem L
L/s	pós-linguagem da linguagem L para uma dada sequência de eventos s
L_aL_b	concatenação das linguagens L_a e L_b
$\|\|t\|\|$	comprimento da palavra t
P	projeção $P : \Sigma_a^* \rightarrow \Sigma_b^*$, sendo que $\Sigma_b \subseteq \Sigma_a$
P^{-1}	projeção inversa $P^{-1} : \Sigma_b^* \rightarrow 2^{\Sigma_a^*}$, sendo que $\Sigma_b \subseteq \Sigma_a$
G	autômato determinístico de estados finitos
G^{nom}	autômato que representa o comportamento nominal de um SED

G_f	autômato que representa o comportamento nominal de um SED com representação da falha f
X	conjunto de estados de um autômato
δ	função de transição de estados em um autômato
Γ	função de eventos ativos no estado de um autômato
x_0	estado inicial de um autômato
X_m	conjunto de estados marcados de um autômato
Ac	componente acessível de um autômato
$CoAc$	componente coacessível de um autômato
$Trim$	componente acessível e coacessível de um autômato
$T^{in}_{x_{A_{N-Y}}}$	conjunto das transições de entrada do estado $x_{A_{N-Y}}$
$\parallel T^{in}_{x_{A_{N-Y}}} \parallel$	número de transições de entrada do estado $x_{A_{N-Y}}$
$G_1 \parallel G_2$	composição síncrona entre os autômatos G_1 e G_2
$\Sigma_1 \setminus \Sigma_2$	complemento relativo de Σ_2 em Σ_1
$C(M,L)$	conjunto de todas as sublinguagens de M que são controláveis em relação a linguagem L
$supC(M,L)$	elemento supremo do conjunto de todas as sublinguagens de M que são controláveis em relação a linguagem L
S	autômato supervisor
S/G	G sob ação de controle de S
$\mathscr{D}$	condição de diagnosticabilidade
R_{N-Y}	autômato rotulador de falha
R_l	autômato rotulador de falha e reconhecedor de palavras proibidas
$Obs(G,\Sigma_o)$	função para cálculo do autômato observador dos eventos de Σ_o em G
Φ	conjunto de palavras proibidas
$\Psi_L(f)$	conjunto de todas as palavras de L que terminam com o evento de falha f
$\mathscr{K}_f$	linguagem proibida
A_{N-Y}	autômato rotulado com rótulos N e Y
A^{ed}_{N-Y}	autômato rotulado com rótulos N e Y e com estados duplicados
D_{N-Y}	diagnosticador obtido pela função $Obs(A_{N-Y},\Sigma_o)$
D^{ed}_{N-Y}	diagnosticador obtido pela função $Obs(A^{ed}_{N-Y},\Sigma_o)$

D	diagnosticador seguro
GD_{scc}	autômato diagnosticador com componentes fortemente conectadas

SUMÁRIO

1 INTRODUÇÃO

Os sistemas industriais modernos, principalmente aqueles que se adequam ao contexto da Indústria 4.0, possuem ação complexa e com grande integração vertical e horizontal de seus setores. Neste cenário, o tratamento de falhas, que porventura possam ocorrer no sistema de produção, se apresenta como uma ação de grande importância para garantia do fluxo e da continuidade de sua operação.

Conforme define Blanke et al. (2016), de uma maneira em geral, uma falha é algo que muda o comportamento de um sistema de tal forma que ele passa a não satisfazer o seu propósito em plenitude. A manifestação da falha acarreta em mudanças na estrutura ou nos parâmetros do sistema e eventualmente o conduz para uma operação degradada ou até mesmo para uma interrupção de seu funcionamento. Com o intuito de evitar perdas na produção e danos a equipamentos, é importante que as falhas sejam detectadas e diagnosticadas e que decisões sejam tomadas a fim de cessar sua propagação e seus efeitos. Estas ações de detecção e diagnóstico de falhas, em conjunto com mudanças na operação do sistema, podem ser executadas pelos controladores com o objetivo de tornar o sistema tolerante a falhas, dando origem ao que se chama de controle tolerante a falhas (CTF).

Segundo Blanke et al. (2016), um CTF tem a habilidade de reagir à existência da falha por meio do ajuste de suas atividades para atender ao comportamento faltoso do sistema. Seu desenvolvimento é efetuado por um processo que consiste de duas etapas: a análise das condições de diagnosticabilidade das falhas no sistema e o projeto de sua estrutura controle.

Tendo em vista que muitas falhas podem ser modeladas como eventos discretos, torna-se útil o uso de modelos de sistemas a eventos discretos para diagnose de falhas e para controle tolerante a falhas (SHU; LIN, 2013). Com isso, é possível desenvolver modelos de CTF por meio do uso de métodos formais para sistemas a eventos discretos. Esses modelos de CTF, operando em conjunto com algum sistema de detecção e isolamento de falhas, permitem que ocorram mudanças nas leis de controle do sistema, a fim de garantir requisitos mínimos de desempenho para uma nova dinâmica de operação do sistema sob condição de falha (PAOLI; SARTINI; LAFORTUNE, 2011).

Um fator importante relacionado aos modelos de controle obtidos por métodos formais em sistemas a eventos discretos é a possibilidade de poder interpretá-los em lógica de controle. Essa temática vem sendo explorada em trabalhos que envolvem principalmente a teoria de controle supervisório (TCS) (RAMADGE; WONHAM, 1987) e tem como foco o desenvolvimento de aplicações que sigam padrões vigentes para sistemas industriais, como as normas IEC 61131 (IEC, 2013b) e IEC 61499 (IEC, 2012a) (IEC, 2012b) (IEC, 2013a). Entretanto, conforme descrito em Zaytoon e Riera (2017), a área de pesquisa relacionada à implementação da estrutura de controle supervisório obtida pela aplicação da TCS, ou de suas extensões, na norma IEC 61499 é promissora, mas ainda pouco estudada. Somado a isso, o conceito de blocos de função existente na IEC 61499 permite que o controle seja projetado por meio de funcionalidades

encapsuladas, o que acarreta em benefícios como: redução de complexidade, reutilização, menor tempo de projeto e melhor qualidade do sistema de controle (ZOITL; LEWIS, 2014).

Considerando o potencial de uso de CTF em soluções que proporcionem síntese de controle dentro do perfil exigido pelas indústrias modernas, principalmente no âmbito da Indústria 4.0, este trabalho contempla o desenvolvimento de uma metodologia voltada para a geração automática (não assistida) de sistemas de CTF baseados em métodos formais aplicados a sistemas a eventos discretos e para a síntese, também automática, desses sistemas de CTF em blocos de função da norma IEC 61499.

1.1 MOTIVAÇÕES

Esta tese é parte integrante do desenvolvimento de uma arquitetura de reconfiguração para tratamento de falhas em sistemas industriais, denominada de *Reconfiguration Architecture for Fault Handling* (RAFAH) (LEITÃO et al., 2020), que relaciona conceitos voltados à diagnose de falhas e ao controle de sistemas a eventos discretos, com aplicações voltadas à norma IEC 61499.

As principais motivações da tese são o desenvolvimento de uma metodologia de projeto de CTF, com aplicação em sistemas a eventos discretos, e a criação de um sistema computacional que permita aplicar a metodologia de projeto de CTF e que opere de forma não assistida na análise da diagnosticabilidade de falhas, na modelagem do CTF e em sua implementação em lógica de controle. Entretanto, devido ao amplo contexto de áreas de estudo relacionadas ao desenvolvimento da metodologia, a tese também incorpora uma motivação específica na área de síntese de lógica de controle em blocos de função da IEC 61499.

O projeto de um CTF depende da classe na qual se deseja incorporá-lo. Segundo Lin (1993), os sistemas de CTF podem ser classificados como sendo de abordagem passiva ou ativa. A abordagem passiva tem como base uma única estrutura de controle capaz de manipular as especificações de controle para os modos nominal (pré-falha) e defeituoso (pós-falha) do sistema (FRITZ; ZHANG, 2018). A abordagem ativa se fundamenta em uma arquitetura de controle capaz de mudar os parâmetros ou a estrutura de controle em uso, caso ocorra uma falha no sistema, e geralmente consiste em uma unidade de diagnose (conjunto de diagnosticadores para as falhas) e em um esquema de reconfiguração (FRITZ; ZHANG, 2018). Em sua operação em malha fechada com o sistema e após a ocorrência da falha, o CTF deve: realizar uma ação de parada do sistema antes que ele execute alguma ação proibida; e conduzir o comportamento do sistema para que ele opere de acordo com um novo conjunto de especificações de controle, projetadas para tratar do comportamento do sistema após a falha (WATANABE et al., 2021). Contudo, apesar dessas ações serem comuns aos dois tipos de abordagens, as soluções obtidas pela abordagem passiva podem ser, em alguns casos, mais complexas e restritivas que as soluções obtidas pela abordagem ativa (FRITZ; ZHANG, 2018). Portanto, baseando-se nisso foi definido o uso de CTF de abordagem ativa na metodologia de projeto proposta nesta tese.

Em relação ao CTF incorporado à metodologia de projeto da tese, desenvolveu-se uma estrutura para ele baseada na arquitetura de supervisão tolerante a falhas apresentada por Paoli, Sartini e Lafortune (2008) e (2011). Nesses trabalhos, os autores estabeleceram uma estrutura de controle a partir da análise conjunta, em uma estrutura monolítica, do comportamento geral do sistema e de todas as suas estratégias de controle (supervisores), antes e após a falha. O módulo de chaveamento existente na arquitetura, ao receber a informação do diagnóstico da falha, troca instantaneamente o supervisor do sistema, colocando-o para operar em um estado que esteja em sincronia com o sistema. Apesar da arquitetura proposta pelos autores ter uma contribuição interessante para a área de CTF, visto que apresenta um conjunto de regras que habilitam o uso de diagnosticadores no tratamento de falhas em sistemas tolerantes a falhas, seu estudo tem as seguintes limitações:

- Ficou restrito ao diagnóstico de uma única falha no sistema;
- Não previu no modelo do CTF ações de chaveamento associadas à recuperação da falha;
- Apresentou regras para a modelagem do CTF apenas de forma assistida;
- Não apresentou uma metodologia para a implementação da arquitetura proposta em uma aplicação prática na IEC 61131 ou na IEC 61499.

Estas limitações fazem com que a proposta do CTF possa ser melhorada com o intuito de tornar sua aplicabilidade mais adaptável, flexível e de simples implantação.

Com relação à síntese de lógica de controle, observou-se que o desenvolvimento de sistemas de automação baseados em modelos de controle obtidos por métodos formais em sistemas a eventos discretos é um tema bastante explorado, contudo o foco principal é dado a aplicações voltadas para a Norma IEC 61131. No que se refere à IEC 61499, foram encontrados os trabalhos apresentados por Cengic et al. (2005), Pang e Vyatkin (2008), Basile, Chiacchio e Gerbasio (2012), Dubinin et al. (2019), Prado (2019), Mendonça (2019) e Foyth (2018) voltados a este tema, sendo que os três últimos consistem em trabalhos de conclusão de curso desenvolvidos no âmbito do mesmo grupo de pesquisa no qual esta tese foi desenvolvida (Grupo de pesquisa em Automação de Sistemas e Robótica - GASR). Entretanto, apenas Cengic et al. (2005), Prado (2019), Mendonça (2019) e Foyth (2018) apresentaram metodologias de implementação da estrutura de controle supervisório de sistemas a eventos discretos no ambiente da IEC 61499. Além destes trabalhos, existe também o estudo apresentado por Pinto, Leal e Rosso (2017), baseado na identificação e manutenção do sistema em estados específicos que permitam a execução de sua reconfiguração dinâmica de forma segura. Sendo assim, não foram encontrados trabalhos explorando a síntese de CTF de sistemas a eventos discretos em implementações na IEC 61499.

1.2 OBJETIVOS

O objetivo geral do trabalho é desenvolver uma metodologia de projeto de CTF para aplicações em sistemas a eventos discretos, na qual seja possível: efetuar análise de diagnosticabilidade de falhas, efetuar análise de controlabilidade segura do sistema por meio da diagnose das falhas, projetar um CTF para o sistema e implementar esse CTF em blocos de função da norma IEC 61499.

Para alcançar o objetivo geral proposto, são contemplados os seguintes objetivos específicos:

- Propor uma arquitetura de CTF que seja baseada na arquitetura apresentada por Paoli, Sartini e Lafortune (2011), mas que flexibilize, por meio de regras, a síntese de sua estrutura, composta por supervisores, módulo de chaveamento e diagnosticadores, permitindo que o CTF possa inicializar por qualquer um dos seus supervisores e atuar no tratamento de mais de uma falha no sistema, inclusive com ações de chaveamento associadas à recuperação da falha;
- Propor uma metodologia para a implementação da estrutura do CTF em blocos de função da IEC 61499;
- Desenvolver e implementar os algoritmos do sistema computacional originado da metodologia de projeto de CTF.

1.3 ORGANIZAÇÃO DO TRABALHO

O restante deste trabalho está estruturado como descrito a seguir. No Capítulo 2, é apresentada uma fundamentação teórica de conceitos que auxiliam no entendimento da metodologia de projeto de CTF proposta. Uma revisão bibliográfica referente ao CTF é apresentada no Capítulo 3. O Capítulo 4 contempla: um estudo de caso baseado em um processo de mistura de substâncias químicas; a descrição da metodologia de projeto de CTF proposta; a representação dos algoritmos que compõem a estrutura do sistema computacional originado da metodologia de projeto de CTF; e a implementação e a análise de um projeto de CTF para o estudo de caso apresentado, baseado-se na metodologia proposta. E finalmente, as conclusões obtidas e as perspectivas de trabalhos futuros são descritas no Capítulo 5.

2 FUNDAMENTAÇÃO TEÓRICA

A fim de separar por tema e organizar melhor os conceitos relacionados ao trabalho, este capítulo está dividido em duas seções. Na primeira seção, o foco é direcionado à introdução dos principais conceitos relacionados aos Sistemas a Eventos Discretos (SEDs), à Teoria de Controle Supervisório (TCS) e à diagnose de falhas em SEDs. Estes conceitos são importantes para a tese pois constituem a base para a modelagem do comportamento dos sistemas, o desenvolvimento de modelos de controle (supervisores) voltados para estes sistemas e a análise do comportamento destes sistemas perante a ocorrência de falhas. Por fim, na segunda seção é feita uma introdução da norma IEC 61499 com o intuito de apresentar, de uma forma geral, suas principais propriedades e de detalhar alguns aspectos importantes que têm relação direta com a metodologia de implementação de CTF proposta na tese.

2.1 SISTEMAS A EVENTOS DISCRETOS

Um SED pode ser definido como sendo um sistema formado por um espaço de estados discreto cuja a dinâmica de evolução se dá pela ocorrência de eventos de natureza discreta, normalmente atemporais. Embora nem todos os sistemas tenham estados de natureza discreta, qualquer sistema pode ser representado por um SED, considerando um certo grau de abstração (CASSANDRAS; LAFORTUNE, 2009). No cenário de um sistema de produção automatizado (SPA), as condições de "máquina parada" e de "máquina em funcionamento" e os comandos para "ligar a máquina" e para "desligar a máquina" são, respectivamente, exemplos de estados e de eventos em SEDs.

2.1.1 Linguagens

No contexto de SEDs, uma linguagem L é definida como sendo um conjunto de palavras de tamanho finito que são formadas pela concatenação de eventos pertencentes a um alfabeto Σ, sendo Σ considerado finito. Por exemplo, se considerarmos o alfabeto $\Sigma = \{a, b, c\}$, a palavra abc é formada pela concatenação dos eventos a, b e c e a linguagem $L_1 = \{aba, abb, abc\}$, sobre Σ, representa todas as possíveis palavras formadas pela concatenação de 3 eventos e iniciadas pela sequência de eventos ab. Assim, uma linguagem é um subconjunto do conjunto de todas as possíveis palavras de comprimento finito formadas com os elementos de Σ. A esse conjunto dá-se o nome de fecho de Kleene de Σ, que é denotado por Σ^*. Formalmente, o fecho de Kleene de Σ é definido como $\Sigma^* = \{\varepsilon\} \cup \Sigma \cup \Sigma\Sigma \cup \cdots$, sendo que ε é chamada de palavra vazia. Analogamente, o fecho de Kleene de L é definido por $L^* := \{\varepsilon\} \cup L \cup LL \cup \cdots$, em que L_aL_b denota a operação de concatenação das linguagens L_a e L_b, definida como $L_aL_b := \{s \in \Sigma^* : (s = s_as_b), (s_a \in L_a) \text{ e } (s_b \in L_b)\}$. Seja $s = tuv$ uma palavra pertencente à linguagem L, considera-se que t, u e v são sequências de eventos que representam o prefixo, a subpalavra e o sufixo de s, respectivamente. O prefixo-fechamento da linguagem L é a linguagem $\overline{L} := \{s \in \Sigma^* : (\exists t \in \Sigma^*)[st \in L]\}$. Portanto,

o prefixo-fechamento $\overline{L}$ é formado por todos os prefixos de todas as palavras da linguagem L. L é denominada prefixo-fechada quando $L = \overline{L}$. A pós-linguagem de L para uma dada sequência de eventos $s \in L$ é definida como $L/s := \{t \in \Sigma^* : st \in L\}$.

Considerando os alfabetos Σ_a e Σ_b, sendo $\Sigma_b \subset \Sigma_a$, a projeção $P : \Sigma_a^* \to \Sigma_b^*$, é definida como:

$$P(\varepsilon) := \varepsilon$$

$$P(e) := \begin{cases} e, & \text{se } e \in \Sigma_b \\ \varepsilon, & \text{se } e \in \Sigma_a \setminus \Sigma_b \end{cases}.$$

$$P(se) := P(s)P(e), \text{ para } s \in \Sigma_a^* \text{ e } e \in \Sigma_a$$

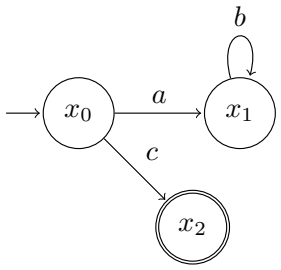
Sendo $\Sigma_a \setminus \Sigma_b = \{\sigma \in \Sigma_a | \sigma \notin \Sigma_b\}$. De forma análoga, a projeção inversa $P^{-1} : \Sigma_b^* \to 2^{\Sigma_a^*}$ de uma palavra $t \in \Sigma_b^*$ é definida como sendo $P^{-1}(t) := \{s \in \Sigma_a^* : P(s) = t\}$. Os conceitos de projeção e de projeção inversa de palavras são extensíveis às linguagens, portanto, dada uma linguagem $L \subseteq \Sigma_a^*$, sua projeção $P : \Sigma_a^* \to \Sigma_b^*$, é definida como $P(L) := \{t \in \Sigma_b^* : (\exists s \in L)[P(s) = t]\}$, e dada uma linguagem $L_b \subseteq \Sigma_b^*$, sua projeção inversa $P^{-1} : \Sigma_b^* \to 2^{\Sigma_a^*}$, é definida como sendo $P^{-1}(L_b) := \{s \in \Sigma_a^* : (\exists t \in L_b)[P(s) = t]\}$.

2.1.2 Autômato Determinístico de Estados Finitos

Um autômato determinístico de estados finitos (ADEF) G é representado pela sêxtupla $G = (X, \Sigma, \delta, \Gamma, x_0, X_m)$, em que: X é o conjunto finito de estados do autômato G; Σ é o conjunto de eventos associados à G; $\delta : X \times \Sigma \to X$ é a função de transição (geralmente parcial em seu domínio); $\Gamma : X \to 2^{\Sigma}$ é a função de eventos ativos; x_0 é o estado inicial do autômato; e $X_m \subseteq X$ é o conjunto de estados marcados. Na função de transição δ, $\delta(x_a, e) = x_b$ significa que existe uma transição do estado x_a para o estado x_b rotulada pelo evento e e na função de eventos ativos Γ, $\Gamma(x)$ representa o conjunto de todos os eventos e ativos no estado x, ou seja, $\delta(x, e)$ está definida em x, o que pode ser denotado por $\delta(x, e)!$. A função de transição δ pode ser recursivamente extendida do domínio $X \times \Sigma$ para o domínio $X \times \Sigma^*$ da seguinte forma: $\delta(x, \varepsilon) := x$, $\delta(x, se) := \delta(\delta(x, s), e)$, para $s \in \Sigma^*$ e $e \in \Sigma$. Portanto, considerando-se o ADEF G e a extensão de domínio de sua função de transição δ, define-se $L(G) := \{s \in \Sigma^* : \delta(x_0, s)!\}$ como sendo a linguagem gerada por G e $L_m(G) := \{s \in L(G) : \delta(x_0, s) \in X_m\}$ como sendo a linguagem marcada em G, formada por sequências de eventos que levam a estados marcados, estados nos quais se completam tarefas no sistema. A Figura 1 mostra um exemplo de ADEF, denominado de autômato G_a, que é formado pelos conjuntos de estados $X = \{x_0, x_1, x_2\}$ e $X_m = \{x_2\}$ e pelo conjunto de eventos $\Sigma = \{a, b, c\}$. A linguagem gerada e a linguagem marcada por G_a são, respectivamente, $L(G_a) = \{\varepsilon, c, a, ab, abb, abbb, \ldots\}$ e $L_m(G_a) = \{c\}$.

A noção de bloqueio em um autômato está associada à impossibilidade de completar uma tarefa (alcançar um estado marcado) a partir de um dado estado. Diz-se que G é não bloqueante

Figura 1 – Autômato determinístico de estados finitos G_a. Os círculos representam os estados do autômato, sendo os estados com círculos duplos denominados de estados marcados. A seta sem origem aponta para o estado inicial e as demais setas indicam as transições entre estados.



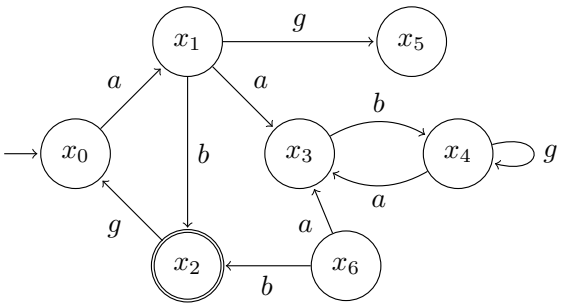
Fonte: Elaborada pelo autor (2022).

se $\overline{L_m(G)} = L(G)$, isto é, se qualquer sequência de eventos gerada for um prefixo de uma tarefa completa.

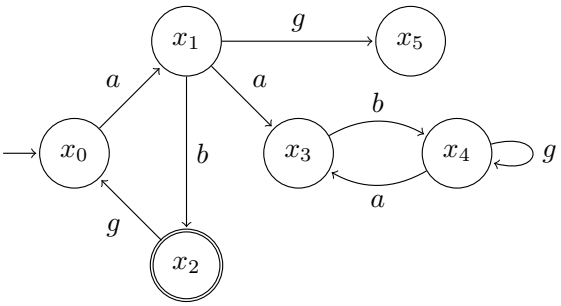
Acessibilidade e coacessibilidade são duas propriedades que podem ser atribuídas a um ADEF G. Um estado $x \in X$ é dito acessível se existe ao menos uma palavra que, a partir do estado inicial, leva a esse estado. A componente acessível de um autômato G é definida por $Ac(G) \coloneqq (X_{Ac}, \Sigma, \delta_{Ac}, \Gamma_{Ac}, x_0, X_{Ac_m})$, sendo $X_{Ac} = \{x \in X : (\exists s \in \Sigma^*)[\delta(x_0, s) = x]\}$, $X_{Ac_m} = X_m \cap X_{Ac}$ e $\delta_{Ac} = \delta : X_{Ac} \times \Sigma \to X_{Ac}$. Quando $G = Ac(G)$, G é dito ser acessível. Um estado $x \in X$ é dito ser coacessível se existe uma sequência de eventos a partir de x que leve o autômato a um estado marcado $x' \in X_m$. A componente coacessível de um autômato G é representada por $CoAc(G) \coloneqq (X_{CoAc}, \Sigma, \delta_{CoAc}, \Gamma_{CoAc}, x_{CoAc_0}, X_m)$, sendo $X_{CoAc} = \{x \in X : (\exists s \in \Sigma^*)[\delta(x, s) \in X_m]\}$, $x_{CoAc_0} = x_0$ (se $x_0 \in X_{CoAc}$) ou x_{CoAc_0} não definido e $\delta_{CoAc} = \delta : X_{CoAc} \times \Sigma \to X_{CoAc}$. Quando $G = CoAc(G)$ ele é dito ser um autômato coacessível. Note que a coacessibilidade está intimamente relacionada ao conceito de bloqueio e que um autômato é bloqueante se existem estados acessíveis que não são coacessíveis. Um autômato é dito ser *trim* quando é acessível e coacessível, portanto $Trim(G) \coloneqq CoAc[Ac(G)] = Ac[CoAc(G)]$ (CASSANDRAS; LAFORTUNE, 2009). A Figura 2 mostra as partes acessível e coacessível de um autômato G_b, formadas, respectivamente, pelos estados que são alcançados a partir do estado inicial x_0 e pelos estados que alcançam o estado marcado x_2.

Sejam os ADEFs $G_1 = (X_1, \Sigma_1, \delta_1, \Gamma_1, x_{1_0}, X_{1_m})$ e $G_2 = (X_2, \Sigma_2, \delta_2, \Gamma_2, x_{2_0}, X_{2_m})$, a composição síncrona entre G_1 e G_2 é o autômato $G_1 \parallel G_2 \coloneqq Ac(X_1 \times X_2, \Sigma_1 \cup \Sigma_2, \delta_{1||2}, \Gamma_{1||2}, (x_{1_0}, x_{2_0}),$

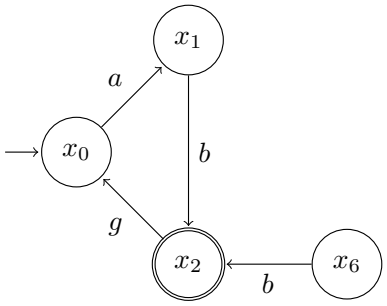
Figura 2 – Exemplo de acessibilidade e coacessibilidade de um autômato.



(a) Autômato G_b.



(b) Componente acessível do autômato G_b.



(c) Componente coacessível do autômato G_b.

Fonte: Adaptada de (CASSANDRAS; LAFORTUNE, 2009).

$(X_{1_m} \times X_{2_m}))$, sendo $\Gamma_{1||2}(x_1,x_2) = [\Gamma_1(x_1) \cap \Gamma_2(x_2)] \cup [\Gamma_1(x_1) \setminus \Sigma_2] \cup [\Gamma_2(x_2) \setminus \Sigma_1]$ e

$$\delta_{1||2}((x_1,x_2),e) = \begin{cases} (\delta_1(x_1,e),\delta_2(x_2,e)), & \text{se } e \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ (\delta_1(x_1,e),x_2), & \text{se } e \in \Gamma_1(x_1) \setminus \Sigma_2 \\ (x_1,\delta_2(x_2,e)), & \text{se } e \in \Gamma_2(x_2) \setminus \Sigma_1 \\ \text{não definido}, & \text{caso contrário} \end{cases} . \quad (1)$$

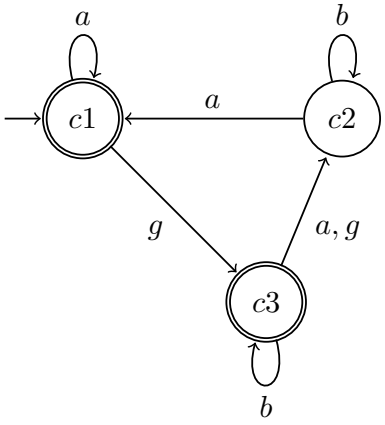
A Figura 3c apresenta um exemplo de operação de composição síncrona entre os autômatos G_c (Figura 3a) e G_d (Figura 3b), sendo $\Sigma_c = \{a,b,g\}$ o alfabeto de G_c e $\Sigma_d = \{a,b\}$ o alfabeto de G_d.

São consideradas propriedades da composição síncrona a comutatividade, $G_1 \parallel G_2 = G_2 \parallel G_1$, e a associatividade $(G_1 \parallel G_2) \parallel G_3 = G_1 \parallel (G_2 \parallel G_3)$, sendo esta extensível para um número maior de autômatos (CASSANDRAS; LAFORTUNE, 2009).

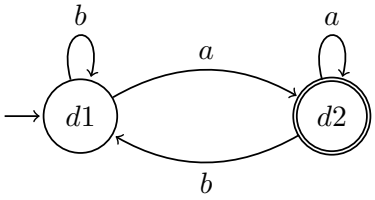
2.1.3 Teoria de Controle Supervisório

A TCS estabelece um método formal para cálculo de supervisores ótimos para SEDs, que devem operar em malha fechada com a planta, conforme mostra a Figura 4, impondo restrições

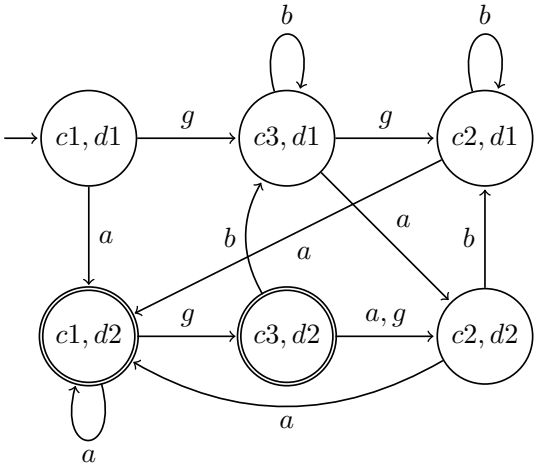
Figura 3 – Exemplo de operação de composição síncrona entre autômatos.



(a) Autômato G_c.



(b) Autômato G_d.



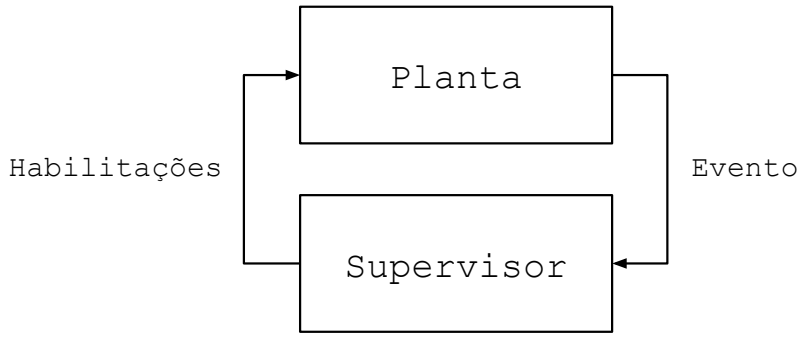
(c) Composição síncrona $G_c \parallel G_d$.

Fonte: Adaptada de (CASSANDRAS; LAFORTUNE, 2009).

de controle à sua dinâmica de operação. A planta, representada pelo autômato G, é composta por um alfabeto de eventos Σ, formado pela união disjunta $\Sigma = \Sigma_{nc} \dot{\cup} \Sigma_c$. Σ_{nc} representa o conjunto de eventos não controláveis de G, eventos que não podem ter a ocorrência diretamente restringida pela ação de controle, e Σ_c representa o conjunto de eventos controláveis de G, eventos que podem ser habilitados ou desabilitados por um agente externo por meio de restrições de controle.

O modelo da planta é comumente formado por diversos subsistemas independentes, os quais podem ser modelados isoladamente pelos autômatos $G_1, \ldots, G_n$, de modo que o modelo para o comportamento global da planta pode ser obtido pela composição síncrona dos modelos desses subsistemas que o compõem, isto é, $G = G_1 \parallel \ldots \parallel G_n$, sendo n o número de subsistemas. A sequência de eventos fisicamente possíveis de serem gerados na planta e a sequência de eventos que a fazem completar tarefas representam, respectivamente, a linguagem gerada $L(G)$ e

Figura 4 – Diagrama de blocos com representação de controle supervisório em malha fechada com a planta.



Fonte: Elaborada pelo autor (2022).

a linguagem marcada $L_m(G)$. Podemos dizer que G modela um "comportamento não controlado" da planta e, dessa forma, necessita que este comportamento sofra restrições para um subconjunto de $L(G)$ por meio de ações de controle provenientes de um supervisor S. Formalmente, o supervisor S é dado pela função $S : L(G) \to 2^{\Sigma}$. Assim, para cada palavra $s \in L(G)$ gerada pela planta sob ação de controle de S, $S(s) \cap \Gamma(\delta(x_0, s))$ é o conjunto de eventos habilitados que G pode executar no estado atual. Diz-se que um supervisor é admissível se para todo $s \in L(G)$, $\Sigma_{uc} \cap \Gamma(\delta(x_0, s)) \subseteq S(s)$, o que significa que um supervisor admissível não desabilita eventos não controláveis que podem acontecer num dado estado da planta. Dada uma planta G e um supervisor admissível (daqui para frente chamado apenas de supervisor) S, denota-se por S/G o SED que representa a planta G sob a ação de controle de S. A linguagem gerada e a linguagem marcada por S/G são definidas como segue.

Definição 2.1.1 (Linguagens Gerada e Marcada por S/G (CASSANDRAS; LAFORTUNE, 2009))**.** A linguagem gerada por S/G é definida recursivamente como segue:

LG.(1) $\varepsilon \in L(S/G)$
LG.(2) $[(s \in L(S/G)) \,\mathrm{e}\, (s\sigma \in L(G)) \,\mathrm{e}\, (\sigma \in S(s))] \Leftrightarrow [s\sigma \in L(S/G)]$.

A linguagem marcada por S/G é definida como

$$L_m(S/G) \coloneqq L(S/G) \cap L_m(G).$$

Diz-se que o supervisor é não bloqueante para G se garantir o não bloqueio do sistema em malha fechada, isto é:

$$L(S/G) = \overline{L_m(S/G)}.$$

Teorema 2.1.1 (Teorema da Controlabilidade (CASSANDRAS; LAFORTUNE, 2009))**.** Considere o autômato G em que $\Sigma_{nc} \subseteq \Sigma$. Seja $K \subseteq L(G)$, em que $K \neq \emptyset$. Existe supervisor S, tal que $L(S/G) = \overline{K}$, se e somente se

$$\overline{K}\Sigma_{nc} \cap L(G) \subseteq \overline{K}.$$

Esta condição em K é chamada de condição de controlabilidade.

A partir do Teorema 2.1.1, sabe-se que se a condição de controlabilidade é satisfeita, então o supervisor S, definido por

$$S(s) = [\Sigma_{nc} \cap \Gamma(\delta(x_0, s))] \cup \{\sigma \in \Sigma_c : s\sigma \in \overline{K}\},$$

resulta em $L(S/G) = \overline{K}$. Esta propriedade garante que é possível obter um supervisor S que, apenas pela desabilitação de eventos controláveis em G, garante um comportamento K em malha fechada.

Suponha que K não seja controlável em relação à $L(G)$, isto é, $\overline{K}\Sigma_{nc} \cap L(G) \nsubseteq \overline{K}$. Nesse caso, deve-se encontrar uma sublinguagem de K que seja controlável em relação à $L(G)$. A classe de sublinguagens de K que são controláveis em relação à $L(G)$ é denotada por $C(K, L(G)) := \{K' \subseteq K : K' \text{é controlável e.r.a. } L(G)\}$. Esse conjunto é fechado para a operação de união e, portanto, existe um elemento supremo, chamado de máxima sublinguagem de K que é controlável e.r.a. $L(G)$, que é dado por

$$SupC(K, L(G)) = \bigcup_{K' \in C(K, L(G))} K'.$$

2.1.4 Diagnose de Falhas

A detecção e o isolamento de falhas são importantes ações relacionadas aos sistemas industriais modernos, principalmente naqueles que possuem grande complexidade. Isso se deve ao fato desses sistemas possuirem requisitos de desempenho e de confiabilidade cada vez mais rigorosos (SAMPATH et al., 1995). Uma falha pode ser considerada como sendo uma mudança repentina no comportamento de um sistema com consequências potencialmente negativas e que afetam o seu desempenho em relação à execução de tarefas predefinidas (MOOR, 2015).

Anteriormente foi descrito que o alfabeto de eventos Σ da planta G é formado por uma união disjunta entre um conjunto de eventos controláveis e um conjunto de eventos não controláveis. Este mesmo alfabeto também pode sofrer uma classificação em função da observabilidade de seus eventos, podendo, portanto, ser representado por uma união disjunta entre um conjunto de eventos observáveis Σ_o e um conjunto de eventos não observáveis Σ_{no}, $\Sigma = \Sigma_o \dot{\cup} \Sigma_{no}$, no sistema. Nesta tese, segue-se o mesmo critério para análise de falhas adotado por Paoli, Sartini e Lafortune (2011), considerando que as falhas são não controláveis e não observáveis, ou seja, uma falha $f \in \Sigma_{nc} \cap \Sigma_{no}$. É importante destacar também que o estudo em questão se restringe aos tipos de falhas permanentes.

A diagnose de uma falha em um sistema é um processo composto por duas etapas. Na primeira etapa, tem-se a análise da diagnosticabilidade da falha, ação efetuada em modo *offline* na qual se busca observar se a falha é diagnosticável. Na segunda etapa, tem-se a diagnose da

falha propriamente dita, ação que ocorre em modo *online* com o sistema em malha fechada e que se caracteriza pela detecção da falha após sua ocorrência no sistema.

Considerando-se que existe um conjunto de falhas Σ_f no sistema, segundo Sampath et al. (1995), Σ_f pode ser particionado em $\prod_f = \{\Sigma_{f_1}, \Sigma_{f_2}, \ldots, \Sigma_{f_n}\}$, sendo $\Sigma_{f_1}, \Sigma_{f_2}, \ldots, \Sigma_{f_n}$ subconjuntos de Σ_f. Por questão de simplificação, será considerado nas definições e teoremas a seguir que $\prod_f = \{\Sigma_f\}$, em que $\Sigma_f = \{f\}$.

O conceito de linguagem diagnosticável é formalizado na Definição 2.1.2, na qual se denota por $\Psi_L(f)$ o conjunto formado por todas as palavras da linguagem L que terminam com o evento de falha f, $\Psi_L(f) = \{sf \in L : s \in \Sigma^*, f \in \Sigma_{no}\}$, e por $\|t\|$ o comprimento da subpalavra t.

Definição 2.1.2 (Linguagem Diagnosticável (SAMPATH et al., 1995))**.** Uma linguagem L, prefixo-fechada, viva [1] e que não contém ciclos de eventos não observáveis [2], é dita diagnosticável em relação a projeção $P : \Sigma \to \Sigma_o$, e ao conjunto de falhas Σ_f se a seguinte condição for verdadeira:

$$(\exists n \in \mathbb{N})(\forall s \in \Psi_L(f))(\forall t \in L/s)(\|t\| \geq n \Rightarrow \mathscr{D}),$$

sendo a condição de diagnosticabilidade $\mathscr{D}$ expressa como:

$$\forall w \in P^{-1}[P(st)] \cap L \Rightarrow f \in w.$$

A Definição 2.1.2 estabelece que para uma linguagem ser diagnosticável é necessário que seja possível detectar a falha f a partir de um certo número finito n de eventos após sua ocorrência.

Neste trabalho, utiliza-se a abordagem de diagnose proposta por Sampath et al. (1995 e 1996), na qual um autômato diagnosticador é usado tanto para a análise *offline* da diagnosticabilidade quanto para realizar a diagnose *online* de falhas. O processo para a obtenção deste autômato é dividido em duas partes. A primeira parte consiste na geração do autômato rotulado $A_{N-Y} = (X_a, \Sigma, \delta_a, \Gamma_a, x_{a_0})$ por meio de uma operação de rotulagem processada pela composição síncrona $A_{N-Y} = G \parallel R_{N-Y}$. O autômato R_{N-Y}, mostrado na Figura 5b, é denominado de autômato rotulador de falha (autômato rotulador $N-Y$). Os rótulos dos estados de R_{N-Y} estampam N (*No*), estado antes da falha, e Y (*Yes*), estado após a falha. Na segunda parte do processo se tem a obtenção do diagnosticador. Portanto, como $\Sigma = \Sigma_{no} \dot{\cup} \Sigma_o$, o comportamento observável de um ADEF pode ser descrito por um autômato chamado de diagnosticador $D_{N-Y} = (X_d, \Sigma_d, \delta_d, \Gamma_d, x_{d_0})$, cujo conjunto de eventos $\Sigma_d = \Sigma_o$. No diagnosticador, os estados são representados por todos os estados em que A_{N-Y} pode estar após a observação de uma sequência de eventos observáveis. Sendo assim, $x_d \in X_d$ é representado por um rótulo composto $\{x_1 l_1, \ldots, x_n l_n\}$, $n = 1, 2, \ldots$, em que $x_1, \ldots, x_n \in X$ é uma parte do rótulo herdada de G e

[1] Considera-se que a linguagem é viva quando $\Gamma(x_i) \neq \emptyset$ para todo $x_i \in X$.

[2] Dada a palavra st, com $s \in \Sigma^*$ e $t \in \Sigma^*_{no} - \{\varepsilon\}$, considera-se a existência de um ciclo de eventos não observáveis quando $\delta(x_0, s) = x_i$ e $\delta(x_0, st) = x_i$.

$l_1,\ldots,l_n \in \{N,Y\}$ é uma parte herdada de R_{N-Y}. Considera-se x_d um estado normal (estado antes da ocorrência da falha) se seu rótulo possuir apenas "N" na parte herdada de R_{N-Y}, considera-se x_d um estado certo de falha (estado após a ocorrência da falha) se seu rótulo possuir apenas "Y" na parte herdada de R_{N-Y} e considera-se x_d um estado incerto (estado em que não se tem certeza se está antes ou após a falha) quando existir tanto "N" quanto "Y" na parte herdada de R_{N-Y}. A Figura 5 mostra um exemplo de obtenção do diagnosticador D_{N-Y} para a falha f ($\Sigma_{no} = \{f\}$), no autômato G_e (Figura 5a), por meio da aplicação da função $Obs(A_{N-Y},\Sigma_o)$, sendo A_{N-Y} o autômato rotulado apresentado na Figura 5c. No diagnosticador, o estado $\{1N\}$ é considerado um estado normal, estado antes da ocorrência da falha f, os estados $\{6Y\}$ e $\{7Y\}$ são considerados estados certos de falha, estados após a ocorrência da falha f, e os estados $\{2N,4Y\}$ e $\{3N,5Y\}$ são considerados estados incertos, estados nos quais não se tem a certeza se o SED modelado por G_e se encontra em um estado anterior ou posterior a falha f.

Em se tratando das condições para a diagnose de falhas em SEDs, o conceito de ciclos indeterminados tem grande importância, pois a existência destes tipos de ciclos pode inviabilizar a condição de diagnosticabilidade da falha. A princípio, para que as condições de diagnosticabilidade fossem verificadas em um autômato G, Sampath et al. (1995) assumiram que não existia ciclos de eventos não observáveis no autômato. Posteriormente, Carvalho, Basilio e Moreira (2012) apresentaram a Definição 2.1.4 que descreve o que são ciclos escondidos e ciclos indeterminados escondidos em um diagnosticador D_{N-Y}.

Definição 2.1.3 (Ciclos Indeterminados Observáveis ((SAMPATH et al., 1995; CARVALHO; BASILIO; MOREIRA, 2012))). Um conjunto de estados incertos $\{x_{d_1},x_{d_2},\ldots,x_{d_p}\} \subset X_d$ forma um ciclo indeterminado observável se as seguintes condições são satisfeitas:

CO.(1) $x_{d_1},x_{d_2},\ldots,x_{d_p}$ formam um ciclo em D_{N-Y};

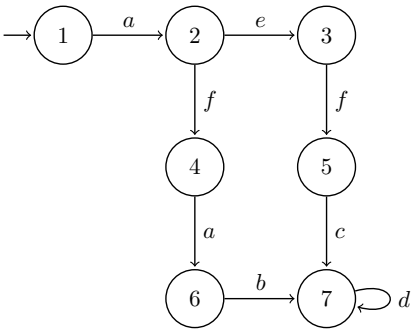
CO.(2) $\exists x_{q,k_q}Y$, $\tilde{x}_{q,r_q}N \in x_{d_q}$, x_{q,k_q} não necessariamente distinto de $\tilde{x}_{q,r_q}$, $q = 1,\ldots,p$, $k_q = 1,\ldots,m_q$ e $r_q = 1,\ldots,\tilde{m}_q$, de tal forma que as sequências de estados $\{x_{q,k_q}\}$ e $\{\tilde{x}_{q,r_q}\}$ formam ciclos em G;

CO.(3) Existem duas sequências $s = s_1\sigma_1\ldots s_p\sigma_p$ e $\tilde{s} = \tilde{s}_1\sigma_1\ldots\tilde{s}_p\sigma_p$, em que $s_i,\tilde{s}_i \in \Sigma_{no}^*$, $\sigma_i \in \Sigma_o$, $i = 1,\ldots,p$, sendo então $P(s) = P(\tilde{s}) \neq \varepsilon$, e $s_q = \sigma_{q,1}\ldots\sigma_{q,m_q-1}$, se $m_q > 1$, ou $s_q = \varepsilon$, se $m_q = 1$, satisfazendo para $q = 1,\ldots,p$, $\delta(x_{q,k_q},\sigma_{q,k_q}) = x_{q,k_q+1}$, $k_q = 1,\ldots,m_q-1$, $\delta(x_{q,m_q},\sigma_q) = x_{q+1,1}$, se $q \neq p$, ou $\delta(x_{p,m_p},\sigma_p) = x_{1,1}$, se $q = p$, e similarmente para $\tilde{s}$.

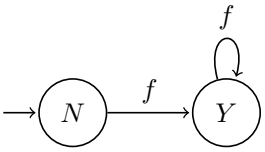
Em resumo, para que um ciclo de estados incertos no diagnosticador D_{N-Y} possa ser considerado um ciclo indeterminado observável, é necessário que: (I) exista um ciclo em D_{N-Y}; (II) o ciclo deve ser composto apenas por estados incertos nos quais a parte do rótulo herdada de G deve formar ciclos em G antes e após a falha.

Definição 2.1.4. (Ciclos escondidos e ciclos indeterminados escondidos (CARVALHO; BASILIO; MOREIRA, 2012)) Seja $x_d = \{x_1l_1,\ldots,x_nl_n\}$ um estado de D_{N-Y}. Existe um ciclo

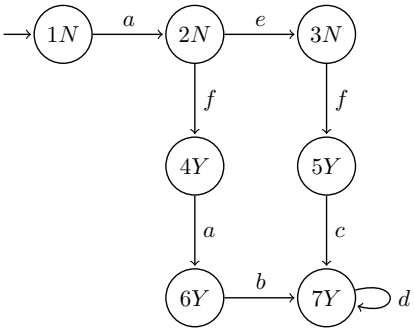
Figura 5 – Obtenção do diagnosticador D_{N-Y} por meio do rotulador R_{N-Y}, com $\Sigma_{no} = \{f\}$.



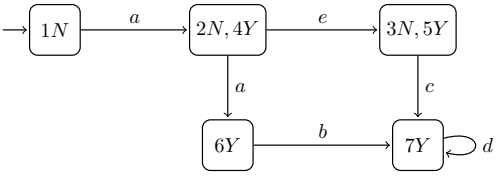
(a) Autômato G_e, sendo $L(G_e) = \overline{a(fab+efc)d^*}$.



(b) Autômato rotulador $N-Y$ (R_{N-Y}).



(c) Autômato rotulado $A_{N-Y} = G_e \parallel R_{N-Y}$.



(d) Diagnosticador $D_{N-Y} = Obs(A_{N-Y}, \Sigma_o)$.

Fonte: Adaptada de (WATANABE, 2019).

escondido em x_d se para algum conjunto de índices $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$, as seguites condições forem satisfeitas:

CI.(1) $x_{i_1}, \ldots, x_{i_k}$ formam um ciclo em G;

CI.(2) $\exists\{\sigma_{i_1}, \ldots, \sigma_{i_k}\} \subseteq \Sigma_{no}$, em que $\sigma_{i_1}, \ldots, \sigma_{i_k}$ são eventos tais que $\delta(x_{i_j}, \sigma_{i_j}) = x_{i_{j+1}}$, sendo $j = 1, \ldots, k-1$, e $\delta(x_{i_k}, \sigma_{i_k}) = x_{i_1}$.

Se x_d é um estado incerto de D_{N-Y} e, além das condições **CI.(1)** e **CI.(2)**, a seguite condição também é satisfeita,

CI.(3) $l_{i_j} = Y, j = 1, \ldots, k$,

então x_d tem um ciclo indeterminado escondido.

Com a definição de ciclos escondidos e ciclos indeterminados escondidos, foi possível explicitar a influência dos ciclos de eventos não observáveis na diagnose de falhas, por meio das seguintes conclusões: (I) ciclos de eventos não observáveis em G que geram ciclos escondidos em estados normais em D_{N-Y} não comprometem a diagnosticabilidade da falha, já que a operação do sistema ainda não foi afetada pela falha; (II) ciclos de eventos não observáveis

em G que geram ciclos escondidos em estados certos de falha em D_{N-Y} não comprometem a diagnosticabilidade da falha, pois como o estado é certo de falha, a diagnose da falha já foi efetuada; e (III) ciclos de eventos não observáveis em G que geram ciclos escondidos em estados incertos de D_{N-Y}, denominados na Definição 2.1.4 como ciclos indeterminados escondidos, comprometem a diagnosticabilidade da falha, pois como estes ciclos de eventos não observáveis ocorrem após a falha, o sistema pode ficar "preso"nestes ciclos e não evoluir para um estado em D_{N-Y} que exista a certeza da falha e, consequentemente, sua diagnose. Baseando-se nestas conclusões e nas conclusões obtidas com a Definição 2.1.3, é possível formular as condições para diagnosticabilidade conforme mostra o Teorema 2.1.2 a seguir.

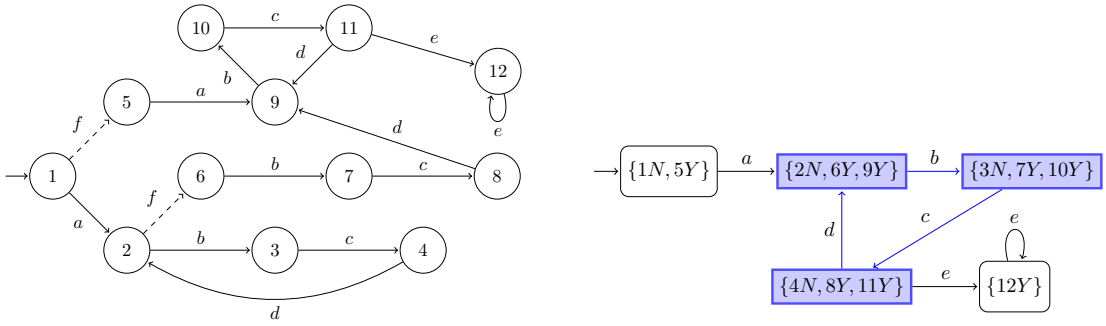
Teorema 2.1.2 (Condições para Diagnosticabilidade (SAMPATH et al., 1995; CARVALHO; BASILIO; MOREIRA, 2012))**.** Uma linguagem L, prefixo-fechada e gerada por um autômato G é diagnosticável em relação à projeção $P : \Sigma^* \to \Sigma_o^*$ e ao conjunto de falhas Σ_f se, e somente se, o seu diagnosticador D_{N-Y} não contém ciclos indeterminados observáveis e nem ciclos indeterminados escondidos.

Portanto, para que seja possível efetuar a análise da diagnosticabilidade de uma falha, é necessário que não existam ciclos indeterminados observáveis nem ciclos indeterminados escondidos no autômato diagnosticador. Exemplos desses tipos de ciclos são mostrados nas Figuras 6 e 7. Podemos ver na Figura 6b que o diagnosticador $D_{N-Y,1}$ possui um ciclo de estados incertos formado por $\{2N, 6Y, 9Y\} \xrightarrow{b} \{3N, 7Y, 10Y\} \xrightarrow{c} \{4N, 8Y, 11Y\} \xrightarrow{d} \{2N, 6Y, 9Y\}$. Este ciclo tem, em parte de sua composição, as sequências de G_1 (Figura 6a): $2 \xrightarrow{b} 3 \xrightarrow{c} 4 \xrightarrow{d} 2$ (antes da ocorrência da falha f) e $9 \xrightarrow{b} 10 \xrightarrow{c} 11 \xrightarrow{d} 9$ (após a ocorrência da falha f). Estas sequências caracterizam o que chamamos de ciclo indeterminado observável, sendo, especificamente a sequência após a ocorrência de f, a sequência que compromete a diagnosticabilidade, já que o sistema pode executar esta sequência indefinidamente e não evoluir para o estado 12, estado que permitiria a diagnose da falha por parte de $D_{N-Y,1}$.

Considere agora um SED modelado pelo autômato G_2 mostrado na Figura 7b. Se a mesma análise for feita sobre o diagnosticador $D_{N-Y,2}$, como não existe ciclo de estados incertos neste diagnosticador, conclui-se que não existe ciclo indeterminado obsevável. Entretanto, como pode ser observado na Figura 7a, a sequência $5 \xrightarrow{g} 6 \xrightarrow{h} 5$ forma em G_2 um ciclo composto apenas por eventos não observáveis. Vemos que este ciclo ocorre após a falha f e que não apresenta nenhuma informação que o represente em $D_{N-Y,2}$, sendo, portanto, chamado de ciclo indeterminado escondido (CARVALHO; BASILIO; MOREIRA, 2012). Já que existe a possibilidade desta sequência ser executada indefinidamente em G_2, o sistema pode não evoluir para o estado 7, o que compromete a diagnose de f por parte de $D_{N-Y,2}$.

A seguir, são apresentados dois conceitos importantes para o controle tolerante a falhas: o de diagnosticabilidade segura e o de controlabilidade segura pela diagnose. A diagnosticabilidade segura consiste na capacidade de detectar a ocorrência da falha antes que o sistema execute uma sequência de eventos considerados proibidos após a falha. Já a controlabilidade segura pela

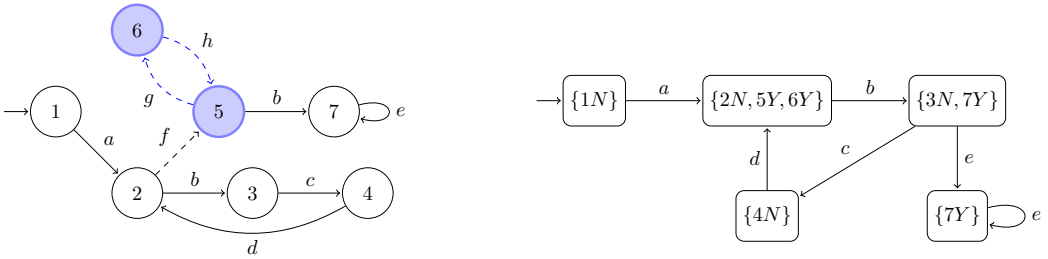
Figura 6 – Exemplo de ciclo indeterminado observável.



(a) Autômato G_1. (b) Diagnosticador $D_{N-Y,1}$ da falha f em G_1.

Fonte: Elaborada pelo autor (2022).

Figura 7 – Exemplo de ciclo indeterminado escondido.



(a) Autômato G_2. (b) Diagnosticador $D_{N-Y,2}$ da falha f em G_2.

Fonte: Elaborada pelo autor (2022).

diagnose representa a capacidade de, a partir da diagnose segura da falha, impedir a ocorrência de eventos ou de sequências de eventos indesejados após a falha.

Considerando que as sequências de eventos proibidos após a falha formam conjuntos de palavras proibidas $\Phi_i (i = 1, \ldots, m)$, sendo $\Phi_i = \{\xi \in \Sigma^* : \xi$ é uma sequência de eventos proibida após a falha$\}$. Todas as possíveis sequências de eventos após uma falha f que possuem uma sequência de eventos de Φ_i formam a linguagem proibida $\mathscr{K}_f$, $\mathscr{K}_f = \{w \in L/s : [s \in \Psi_L(f)][\exists \xi \in \Phi_i$ tal que ξ é uma subpalavra de $w\}$.

As noções fundamentais para que uma linguagem seja considerada diagnosticável segura e controlável segura pela diagnose são apresentadas nas Definições 2.1.5 e 2.1.6 respectivamente.

Definição 2.1.5 (Diagnosticabilidade Segura (PAOLI; LAFORTUNE, 2005))**.** Uma linguagem L, prefixo-fechada e viva, é dita ser diagnosticável segura em relação à projeção $P : \Sigma^* \to \Sigma_o^*$, ao evento de falha f e à linguagem proibida $\mathscr{K}_f$, se as seguintes condições forem verdadeiras:

DS.(1) L é diagnosticável em relação à $P : \Sigma^* \to \Sigma_o^*$ e ao evento de falha f;

DS.(2) $(\forall s \in \Psi_L(f))(\forall t \in L/s)$, considerando que t_c seja o prefixo mais curto de t tal que a diagnose seja atendida, então $\bar{t}_c \cap \mathscr{K}_f = \emptyset$.

Definição 2.1.6 (Linguagem Controlável Segura pela Diagnose ((PAOLI; LAFORTUNE, 2005), (WATANABE, 2019))). Uma linguagem L, prefixo-fechada e viva, é dita ser controlável segura pela diagnose em relação à projeção $P : \Sigma^* \to \Sigma_o^*$, ao evento de falha f e à linguagem proibida $\mathscr{K}_f$, se a seguinte condição for verdadeira:

CS.(1) A linguagem L é diagnosticável segura em relação à projeção $P : \Sigma^* \to \Sigma_o^*$, ao evento de falha f e à linguagem proibida $\mathscr{K}_f$;

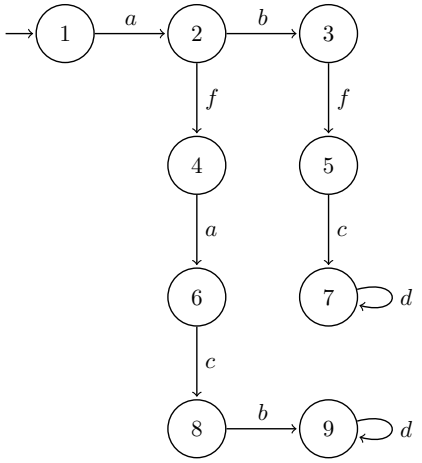
CS.(2) Sendo $s \in L$, tal que o evento de falha $f \in s$ e $s = v\lambda$, $s \in \Sigma^*$ e $\lambda \in \Sigma_o$. Supondo que a condição de diagnosticabilidade segura é atendida pela palavra s mas não é atendida pela palavra v. Então, $(\forall t \in L/s)$, tal que $t = u\xi$, com $\xi \in \Phi$, $\exists \sigma \in \Sigma_c$ tal que $\sigma \in t$.

A Definição 2.1.6 diz que para uma linguagem L possa ser considerada controlável segura, é necessário que exista, após a falha f e antes de alcançar uma subpalavra proibida $\xi \in \Phi$, um evento $\lambda \in \Sigma_o$ que assegure a diagnosticabilidade segura da falha e um evento $\sigma \in \Sigma_c$ na subpalavra u que permita que o sistema possa ser desabilitado com segurança.

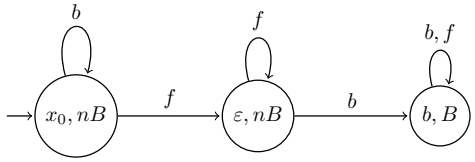
A análise da diagnose segura e da controlabilidade segura pela diagnose de falhas em autômatos depende de um sistema de rotulagem diferente do que foi apresentado anteriormente e que permita a modelagem de um diagnosticador que é denominado de diagnosticador seguro $D = (X_D, \Sigma_o, \delta_D, \Gamma_D, x_{D_0})$. Para sua obtenção, é necessário que exista uma rotulagem em G com marcações que classifiquem seus estados em maus estados (B, *Bad*) e não maus estados (nB, *not Bad*). Na análise, são considerados maus estados aqueles estados que são alcançados após a ocorrência de uma sequência de eventos pertencente a um conjunto de sequências proibidas Φ, que, por sua vez, deve ocorrer após a falha. A Figura 8b mostra um exemplo de um reconhecedor, autômato que fica responsável por gerar estas marcações, que considera f como evento de falha e $\Phi = \{b\}$ como conjunto de sequências proibidas. Conforme é apresentado por Paoli e Lafortune (2005), para a obtenção do diagnosticador seguro, é necessário que as informações provenientes do reconhecedor sejam complementadas com as informações do rotulador $N - Y$, sendo o mapeamento dos rótulos N e Y no reconhecedor uma forma de realizar esta ação, conforme mostra a Figura 8c.

A Figura 8 mostra um exemplo de obtenção do diagnosticador seguro D (Figura 8d) para a falha f ($\Sigma_{no} = \{f\}$) no autômato G_f (Figura 8a). A análise da diagnosticabilidade da falha em D indica que f é diagnosticável segura, visto que é possível diagnosticar a falha (estado $\{6YnB\}$) antes do sistema alcançar o mau estado $\{9YB\}$. Além disso, se $c \in \Sigma_c$, pode-se observar que a linguagem é controlável segura pela diagnose, uma vez que ao desabilitar o evento c, após o sistema alcançar o estado $\{6YnB\}$, isto é, após a diagnose, o mau estado $\{9YB\}$ não será alcançado.

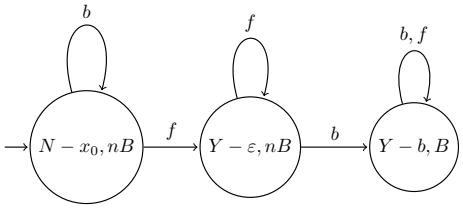
Figura 8 – Obtenção do diagnosticador seguro D.



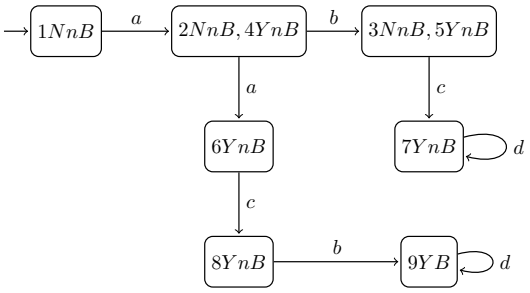
(a) Autômato G_f, sendo $\Sigma_{no} = \{f\}$ e $\Sigma_c = \{c\}$.



(b) Reconhecedor de sequência proibida $\Phi = \{b\}$.



(c) Reconhecedor de sequência proibida $\Phi = \{b\}$ com mapeamento de rótulos N e Y.



(d) Diagnosticador D.

Fonte: Adaptada de (WATANABE, 2019).

2.2 NORMA IEC 61499

A IEC 61499 (2012a, 2012b e 2013a) é uma norma voltada à definição de padrões para sistemas de controle e medição de processos industriais, com foco no desenvolvimento de sistemas de controle distribuídos. Seus conceitos buscam estabelecer um ambiente aberto, orientado a componentes e com estrutura de desenvolvimento independente de plataforma, proporcionando uma melhor (re)usabilidade, (re)configurabilidade, interoperabilidade, portabilidade e distribuição do *software* de controle (PANG et al., 2014). Na IEC 61499, o sistema de controle é projetado por uma rede de blocos de função interconectados, que opera em um ambiente de execução orientado a eventos. Todos os elementos que compõem um projeto desenvolvido por meio da Norma IEC 61449, incluindo o sistema e os blocos de função, são definidos na norma com representações em arquivos com linguagem de marcação do tipo *eXtensible Markup Language* (XML). A definição de tipo de documento, *Document Type Definition* (DTD), de cada um desses elementos é apresentada na Parte 2 (Requisitos da Ferramenta de Software) da norma

((IEC, 2012b)).

Oito níveis de abstração são adotados como modelos de referência na Norma IEC 61499:

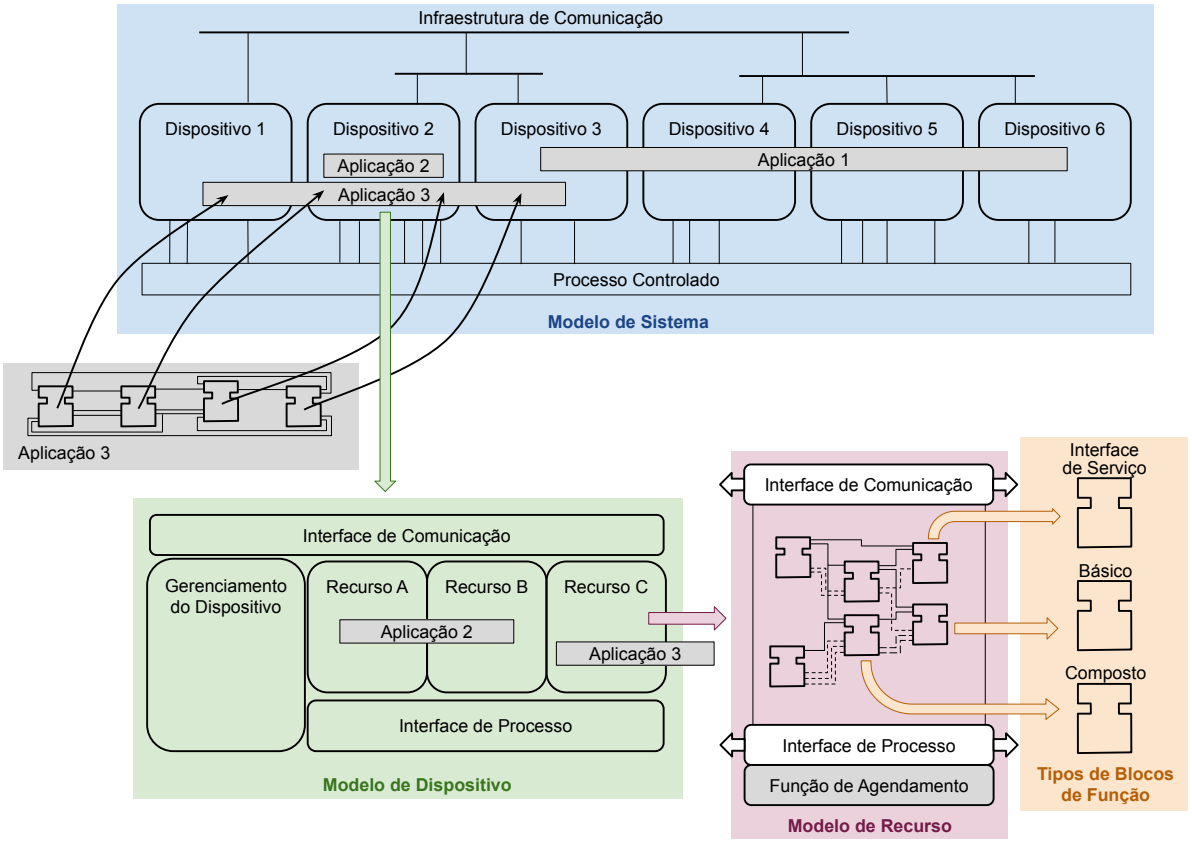
- Modelo de sistema;
- Modelo de aplicação;
- Modelo de distribuição;
- Modelo de dispositivo;
- Modelo de recurso;
- Modelo de bloco de função;
- Modelo de gerenciamento;
- Modelo de estado operacional.

Um exemplo de alguns desses modelos e de como eles se relacionam na estrutura definida pela IEC 61499 pode ser vista na Figura 9, mostrando: o modelo de sistema, composto pelos dispositivos de controle e pelas redes de comunicação que o interligam; o modelo de dispositivo, com os recursos que possui e com as aplicações (sistemas de automação) nas quais atua; e o modelo de recurso, detalhando a parte da rede de blocos existente na aplicação e que se encontra mapeada em um recurso do dispositivo.

O modelo de sistema é composto por uma representação do sistema em seu nível físico. Por se tratar de uma norma voltada à sistemas distribuídos, o modelo do sistema possibilita a interconexão de um conjunto de dispositivos por várias redes de comunicação, propiciando um ambiente operacional para vários tipos de aplicações.

O modelo de aplicação consiste na interconexão de uma rede de blocos de função por meio de eventos, permitindo fluxo de dados entre eles. Portanto, uma aplicação é formada por instâncias de blocos de função e definições de interconexões que, em alguns casos, incluem várias instâncias de blocos de determinados tipos de bloco de função. Em termos práticos, uma aplicação é um conjunto completo de blocos de função e interconexões que visa a resolução de um problema específico de controle ou automação (ZOITL; LEWIS, 2014). A IEC 61499 permite que uma aplicação possa ser distribuída em diferentes recursos, que podem coexistir ou não no mesmo dispositivo definido no modelo do sistema. A Figura 10 mostra um exemplo de rede de blocos de função de uma aplicação distribuída em um recurso. A aplicação representa um sistema de controle de temperatura para ambientes. Seu funcionamento inicia com o disparo do evento *COLD* no bloco de função *START*, que é um bloco de função padrão da IEC 61499. Por meio do evento *COLD*, são inicializados os blocos *OBTER_TEMPERATURA* e *ENVIAR_COMANDO*, que devem habilitar conexões em rede, e o bloco *REFRIGERACAO*, que inicialmente deve fazer a leitura dos níves máximo e mínimo de temperatura a serem utilizados no sistema

Figura 9 – Exemplo de visão geral dos modelos que compõem a IEC 61499.



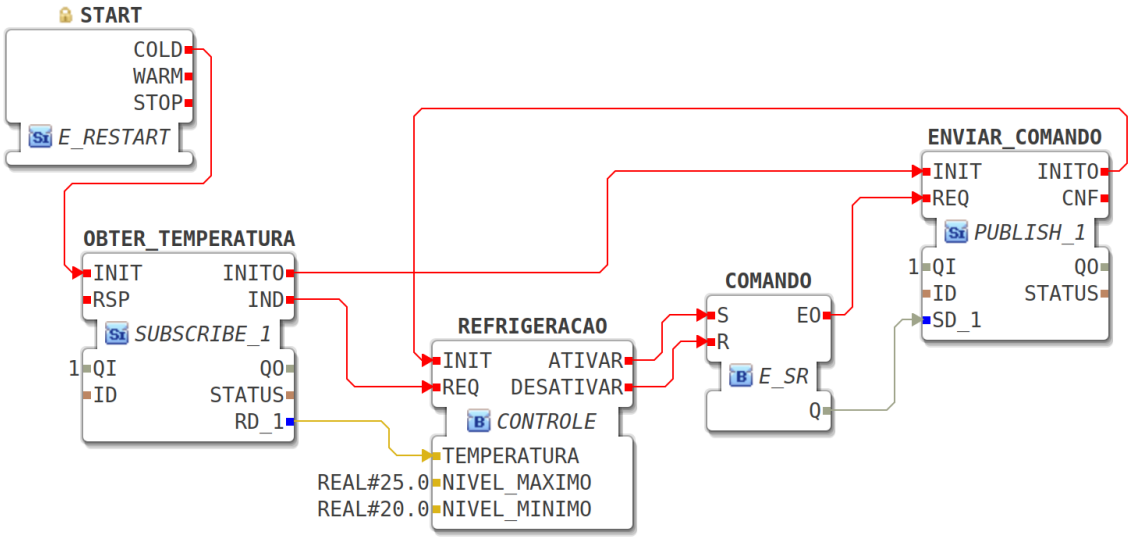
Fonte: Adaptada de Zoitl (2008).

de controle de refrigeração do ambiente. Após esta sequência de inicialização, assim que o bloco *OBTER_TEMPERATURA* receber uma nova informação a respeito da temperatura no ambiente, irá encaminhar esta informação para o bloco *REFRIGERACAO*. Caso exista necessidade de executar algum comando para ativar ou desativar o sistema de refrigeração, o bloco *REFRIGERACAO* irá disparar um evento que será transformado em comando binário (*ATIVAR* = *true* e *DESATIVAR* = *false*) e que será na sequência publicado para o sistema de refrigeração.

O modelo de dispositivo descreve a estrutura geral de qualquer dispositivo de controle físico que suporte a execução de uma rede de blocos de função. O principal objetivo do dispositivo é fornecer uma infraestrutura para sustentar um ou mais recursos (ZOITL; LEWIS, 2014).

No modelo de recursos, os recursos fornecem facilidades e serviços para apoiar a execução de redes de blocos funcionais alocados a eles. Para isso, o recurso deve garantir que os eventos sejam usados para agendar a funcionalidade apropriada, encapsulada dentro dos blocos de função na prioridade e no tempo corretos (ZOITL; LEWIS, 2014). O recurso também é responsável: pelo mapeamento de dados e fluxo de eventos que passam dos blocos de função, no recurso local, para os blocos de função nos recursos remotos, por meio das interfaces de comunicação do dispositivo; e pelo mapeamento das solicitações de leitura e escrita para as entradas e saídas

Figura 10 – Exemplo de rede de blocos de função para controle de temperatura.



Fonte: Elaborada pelo autor (2022).

das interfaces de processo do dispositivo.

No modelo de gerenciamento, tem-se na IEC 61499 a definição de uma forma especial de aplicação denominada de aplicação de gerenciamento. Por possuir privilégios especiais no sistema, este tipo de aplicação gerencia o ciclo de vida de aplicações e de redes de blocos em recursos (ZOITL; LEWIS, 2014). Sua funcionalidade permite, por exemplo, a criação ou a remoção de instâncias de blocos de função em um recurso, a criação ou a remoção de conexões de eventos e variáveis entre instâncias de blocos de função e a inicialização da execução de instâncias de blocos de função em uma aplicação. Conforme descreve Pinto (2019), a aplicação de gerenciamento geralmente se encontra disponível de uma forma não volátil no dispositivo, permitindo que os dispositivos carreguem as aplicações quando ligados.

O modelo de estado operacional presente na IEC 61499 está atrelado ao conceito de ciclo de vida operacional dos dispositivos, recursos e blocos de função. O conceito de ciclo de vida permite que existam diferentes fases de operação para estas entidades, possibilitando que mudanças, atualizações e manutenções tenham o mínimo de impacto possível no sistema. Dessa forma, é proposto na IEC 61499 que este ciclo de vida operacional tenha um modelo que defina claramente para todas as entidades gerenciadas quando elas estão habilitadas para manipulação de eventos, processamento de dados e geração de dados e eventos de saída (ZOITL; LEWIS, 2014).

O componente fundamental da IEC 61499 é o bloco de função, *Function Block* (FB). O FB é um elemento de software autocontido que fornece suas funções por meio de uma interface definida (ZOITL, 2008). A execução de um FB depende da ocorrência de um evento de entrada, desencadeando a aquisição de dados de entrada (quando há), o processamento interno de informações e a geração de eventos em conjunto com a atualização de dados na saída (quando há). Os eventos podem ser associados aos dados por meio de um qualificador denominado "WITH".

Na representação gráfica, isso é mostrado por meio de um pequeno conector quadrado que liga o evento aos dados associados (Figura 11). Os tipos de dados utilizados nos blocos de função são os tipos de dados definidos na IEC 61131-3.

Os blocos de função encapsulam o que se chama de funcionalidade. O tipo e a forma da funcionalidade encapsulada no bloco de função dependem do tipo do bloco. Como a IEC 61499 não permite variáveis globais, a funcionalidade encapsulada só tem acesso às variáveis de entrada, de saída e internas que são retidas entre as invocações do bloco de função.

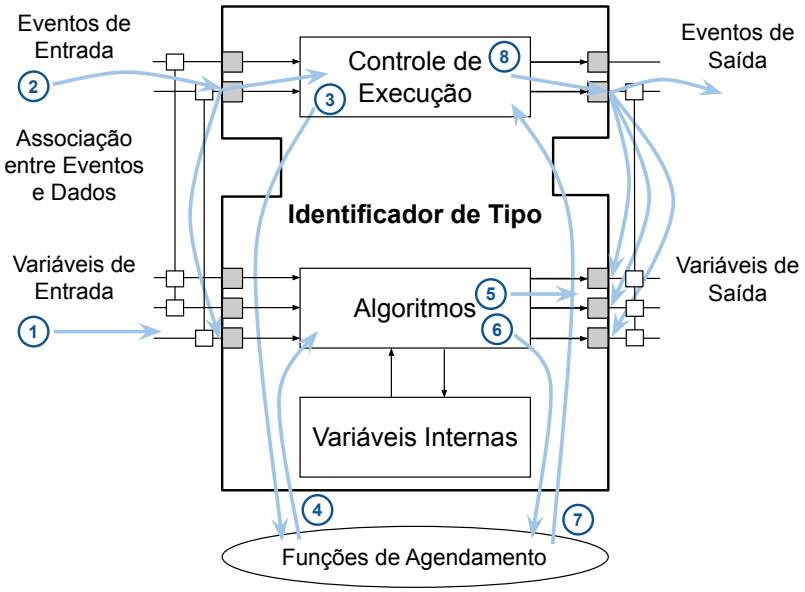
Os blocos de função contêm todos os algoritmos e valores de inicialização para definir seu comportamento de forma plena. A execução de um bloco de função é algorítmica, portanto termina em um tempo finito e não possui elementos de bloqueio ou parada. A IEC 61499 define dois tipos diferentes de blocos de função, o bloco de função básico e o composto, e especifica um terceiro tipo de bloco denominado de interface de serviço.

O bloco de função básico, *Basic Function Block* (BFB), tem seu comportamento definido pela ação de um diagrama de controle de execução, denominado *Execution Control Chart* (ECC), e por algoritmos que são chamados em resposta a eventos de entrada. O ECC consiste em uma máquina de estados formada por estados, transições e ações de execução de algoritmos e de disparo de eventos. Qualquer linguagem pode ser utilizada nos algoritmos presentes no BFB, inclusive as linguagens definidas na IEC 61131-3 (IEC, 2013b). A Figura 11 exemplifica um BFB mostrando: os elementos que compõem sua interface externa, os elementos que compõem sua estrutura interna e o fluxo de ações (etapas numeradas) que ocorrem em sua execução. Na execução do BFB, as variáveis de entrada, que são atualizadas na etapa 1, não sofrem mudanças após a amostragem que ocorre na etapa 2. Isso garante valores de dados estáveis e determinísticos durante a execução da funcionalidade encapsulada. Dependendo de qual evento de entrada aciona o bloco e do estado interno de seu controle de execução, as etapas 3 a 8 podem ser executadas várias vezes. Em tais casos, vários eventos de saída podem ser disparados em resposta à ocorrência de um evento de entrada.

O bloco de função composto, *Composite Function Block* (CFB), é utilizado para encapsular um conjunto de blocos. Seu comportamento interno é definido por uma rede de instâncias de blocos de função. A definição, portanto, inclui conexões de dados e eventos que precisam existir entre as instâncias internas dos blocos de função. A principal função do CFB é a de organizar, em níveis hierárquicos, a estrutura do controle em desenvolvimento.

O bloco de função de interface de serviço, *Service Interface Function Block* (SIFB), é um bloco específico para comunicação, possibilitando, por exemplo, leitura ou escrita de informações nas interfaces de entrada e de saída do controlador ou a troca de mensagens via rede, por meio de protocolos de redes de comunicação industriais. Os blocos de interface de serviço se dividem em dois grupos de blocos: os blocos requisitores e os blocos respondedores. Os blocos requisitores são acionados pela aplicação, portanto precisam da ativação de um de seus eventos de entrada para que seja processado. Os blocos respondedores são acionados pelo recurso ou pelo *hardware*, tendo a ocorrência de eventos de saída sempre que houver a chegada

Figura 11 – Estrutura do bloco de função básico.



Fonte: Adaptada de Cengic e Akesson (2010).

de uma informação. A representação da interface do SIFB segue a mesma estrutura do BFB e do CFB, porém a especificação de sua operação interna está fora do escopo da norma IEC 61499.

Para obter mais informações referentes aos outros modelos presentes na Norma IEC 61499 e apresentados na Figura 9, assim como conhecer outros elementos que ela disponibiliza, como as subaplicações e as interfaces adaptadoras, recomenda-se a leitura do livro de Zoitl e Lewis (2014).

2.2.1 Ambientes de Desenvolvimento de Projetos e de Execução

Existem alguns ambientes gráficos que permitem o desenvolvimento de projetos baseados na IEC 61499. Alguns exemplos desses ambientes são o 4diac IDE (4DIAC, 2022), o FBDK (HOLOBLOC, 2008), o ISaGRAF, o NxtStudio e o GASR-FBE (UDESC, 2014). O 4diac IDE e o FBDK são os ambientes mais usuais e explorados em projetos de pesquisa científica. O ISaGRAF e o NxtStudio são ambientes comerciais pagos e desenvolvidos para soluções que incorporem produtos (*hardware* e *software*) dos seus fabricantes. O GASR-FBE é um ambiente gráfico desenvolvido na Udesc que foi utilizado em alguns projetos de pesquisa científica, entretanto, não vem sendo utilizado em novos projetos por necessitar de uma atualização para atender aos requisitos da nova versão da IEC 61499.

O 4diac IDE é um ambiente de desenvolvimento de código aberto que passa por constantes atualizações e desenvolvimento. Seu ambiente gráfico possibilita o uso de um grande número de recursos definidos na IEC 61499. Além disso, possui ferramentas que possibilitam o teste de blocos e o monitoramento de sistemas em execução.

O FBDK é um ambiente de desenvolvimento que também possui um grande número

de recursos definidos na IEC 61499. Ele é disponibilizado juntamente com uma biblioteca de modelos de blocos que abrangem diversas funcionalidades. Apesar de possibilitar a compilação e o teste de blocos em desenvolvimento de forma rápida e fácil, seu ambiente gráfico é pouco interativo.

Com relação aos ambientes de execução, pode-se destacar o FORTE, o FBRT e o ICARU-FB. O FORTE é um ambiente de execução implementado em C++ e disponibilizado em conjunto com o 4diac IDE. Ele suporta reconfiguração dinâmica e pode operar em controladores lógicos programáveis da família PFC 200 da Wago ou como *softPLC* em sistemas operacionais Linux, Windows ou Mac e em microcontroladores Raspberry Pi ou Beaglebone. Uma de suas grandes vantagens é a possibilidade de uso de protocolos de redes industriais como, por exemplo, Modbus, OPC UA e OPC DA. O FBRT é um abiente de execução implementado em Java e que pode operar como *softPLC* em sistemas operacionais Linux ou Windows e em microcontroladores Raspberry Pi. A comunicação em rede feita pelo FBRT permite o uso apenas do protocolo FBDK/ip, portanto, o FBRT não opera com protocolos de redes industriais em seu ambiente de execução. O ICARU-FB (PINTO et al., 2016) é um ambiente de execução multiplataforma desenvolvido para operar em ambientes computacionais providos de poucos recursos, como a placa de prototipagem Arduino. Seu uso tem foco em dispositivos de campo com baixa necessidade de processamento, como atuadores e sensores.

Na tese, os projetos foram desenvolvidos no 4diac IDE e os testes de simulação em máquinas FORTE. As escolhas destes ambientes de desenvolvimento e de execução foram motivadas pela possibilidade de monitoramento do sistema no ambiente gráfico e pela possibilidade de uso do protocolo OPC UA.

2.3 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo foi apresentada uma fundamentação teórica baseada em diversos conceitos relacionados ao desenvolvimento da metodologia de CTF proposta na tese. A primeira seção foi destinada a apresentação de conceitos relacionados a SEDs que são importantes para o entendimento da revisão bibliográfica relacionada à CTF, apresentada no Capítulo 3, e para o desenvolvimento da metodologia de projeto de CTF proposta na tese, descrita no Capítulo 4. Já na segunda seção foi feita uma introdução à norma IEC 61499, que se baseia no desenvolvimento de projetos de sistemas de controle e automação distribuídos por meio de redes de blocos de função. O BFB é um dos principais componentes presentes na IEC 61499 e teve uma descrição mais detalhada, tanto de sua estrutura quanto de seu modo de execução. O BFB possui em sua estrutura um diagrama de controle de execução que possui uma dinâmica que se assemelha ao comportamento dos autômatos estudados na Seção 2.1. Baseando-se neste fato, é visto no Capítulo 4 que a metodologia de projeto de CTF proposta executa a implementação do CTF em BFBs.

3 REVISÃO BIBLIOGRÁFICA

O desenvolvimento da arquitetura RAFAH se fundamenta em um estudo multidisciplinar, envolvendo conceitos baseados na diagnose de falhas em SEDs, no CTF de SEDs e no desenvolvimento de projetos de automação e controle por meio da norma IEC 61499. Contudo, seu foco principal é o desenvolvimento do CTF, especialmente em sua arquitetura. Sendo assim, este capítulo apresenta uma revisão bibliográfica que aborda, em um primeiro momento, as classificações existentes de CTF de SEDs, apresentando conceitos e noções fundamentais, e, depois, enfatiza trabalhos que propõem arquiteturas de CTF.

A revisão bilbiográfica foi baseada inicialmente em uma busca de trabalhos realizada nas bases de dados do *IEEE Xplore*, da *Science Direct* e do *Mendeley*. Nesta etapa inicial da busca, foram utilizadas as palavras-chave *"Fault-Tolerant Control"* e *"Discrete Event Systems"*. Os trabalhos obtidos foram analisados, sendo escolhidos aqueles que tratavam especificamente de CTF em SEDs com representações em autômatos. Posteriormente, a revisão bibliográfica foi complementada com trabalhos previamente conhecidos e com uma busca recursiva de trabalhos citados. Como resultado final, foram obtidos os trabalhos tratados nas Seções 3.1 e 3.2 a seguir.

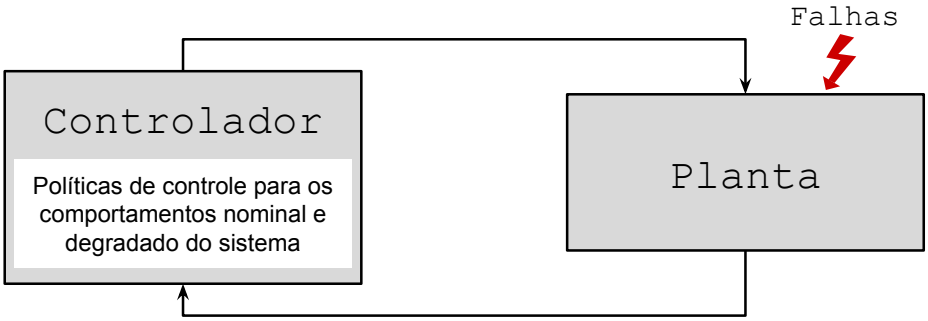
3.1 CONTROLE TOLERANTE A FALHAS DE SISTEMAS A EVENTOS DISCRETOS

Um sistema de CTF é uma estrutura em malha fechada fundamentalmente composta pela planta e pelo controlador, entretanto distingue-se dos sistemas de controle tradicionais por apresentar no controlador estratégias de controle específicas para diferentes condições físicas da planta. Portanto, além de contemplar uma estratégia de controle para o comportamento nominal (normal) da planta, também possui estratégia de controle para atender seu comportamento degradado, representado por operação pós-falha. Conforme descrito por Lin (1993), os sistemas de CTF podem ser classificados como sendo de abordagem passiva ou ativa. Além dessas formas de classificação, existem outras formas que podem ser adotadas, baseadas no modelo utilizado na representação do sistema a eventos discretos (SED), autômatos ou redes de Petri, na condição de recuperação da falha e na perda de observabilidade. Para mais detalhes a respeito do assunto, recomenda-se a leitura das revisões realizadas por Moor (2016), Fritz e Zhang (2018) e Karimadini, Karimoddini e Homaifar (2018).

Os sistemas de CTF com abordagem passiva têm como característica a síntese do que chamamos de estrutura de controle robusta, uma estrutura de controle geral, capaz de conduzir a operação do sistema em malha fechada, antes ou após a ocorrência da falha (Figura 12). Um fato relevante nesse tipo de abordagem é que a estratégia de controle, projetada *offline* a partir de um modelo que contempla todas as possíveis formas de operação do sistema, pode operar em modo *online* sem necessitar da informação de ocorrência da falha (PAOLI; SARTINI; LAFORTUNE, 2011). Entretanto, essa forma de desenvolvimento de CTF, na qual se tem um único controlador com todas as possíveis políticas de controle que podem ser aplicadas ao sistema, geralmente produz uma estrutura de controle maior. Exemplos de estratégias de CTF com abordagem passiva,

apresentados recentemente na literatura, são a arquitetura de controle hierárquica, proposta por Moor e Schmidt (2015), a análise de controlabilidade para diferentes cenários de falha de um sistema tolerante a falhas, realizada por Radel, Mulahuwaish e Leduc (2015), a proposição de um *framework* de síntese de CTF que maximiza a operação do sistema na ausência de falha e minimiza sua operação após a ocorrência de falha, apresentada por Wen, Kumar e Huang (2013), a síntese de supervisores tolerante a falhas por meio do cálculo de sublinguagens supremas do sistema, apresentada por Sülek e Schmidt (2013) e Acar e Schmidt (2015), e os modelos com acomodação de falhas, descritos por Wittmann, Richter e Moor (2012).

Figura 12 – Arquitetura de controle tolerante a falhas de abordagem passiva.

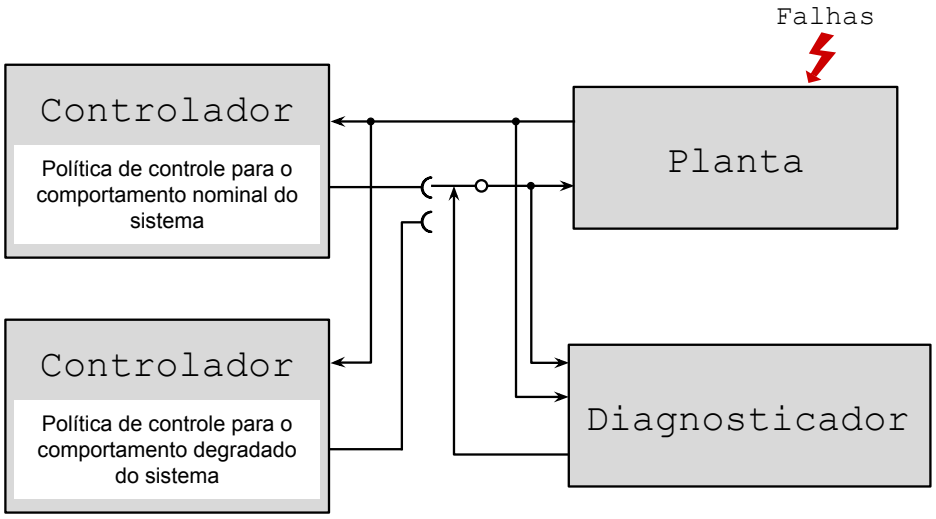


Fonte: Elaborada pelo autor (2022).

Nos sistemas de CTF considerados de abordagem ativa, o controle do sistema em malha fechada é formado por um conjunto de políticas de controle que podem ser habilitadas ou desabilitadas por meio de um chaveamento, acompanhando mudanças de comportamento que são ocasionadas por falhas na planta (Figura 13). Sua estrutura geralmente é composta por diagnosticadores de falhas, por diferentes políticas de controle e por um módulo para coordenar o chaveamento entre essas políticas (FRITZ; ZHANG, 2018). Segundo Moor (2016), esta condição de existência de um controlador que atue especificamente com o comportamento nominal do sistema é um ponto positivo desta abordagem. Além disso, em relação à condução do comportamento pós-falha do sistema a uma nova política de controle, a estrutura de controle produzida na abordagem ativa geralmente é mais flexível que a estrutura desenvolvida pela abordagem passiva (WATANABE et al., 2021). Sendo também denominada de estratégia de controle adaptativa, a elaboração de um CTF pela abordagem ativa primeiramente se baseia, geralmente em modo *offline*, na síntese dos controladores (estratégias de controle) e no projeto dos diagnosticadores e do módulo de chaveamento (componente de reconfiguração do controle), para posteriormente operar com esses elementos em malha fechada com o sistema. Exemplos de CTF com abordagem ativa, apresentados recentemente na literatura, são a representação conceitual de CTF para redes de sensores, apresentada por Darabi, Jafari e Buczak (2003), a arquitetura de CTF composta por um supervisor nominal e por um diagnosticador-controlador, apresentada nos trabalhos de Paoli, Sartini e Lafortune (2008) e Paoli, Sartini e Lafortune (2011),

a proposta de CTF que se baseia no uso de especificações distintas para as etapas anterior e posterior a falha no sistema, presente no trabalho de Kumar e Takai (2012), a abordagem de reconfiguração de controle baseada em autômatos não determinísticos com entradas e saídas que executa procedimentos de replanejamento de trajetória e de adaptação das entradas e saídas, apresentada nos trabalhos de Nke e Lunze (2010), Nke e Lunze (2011) e Nke e Lunze (2012), a proposta de CTF baseada em autômatos determinísticos com entradas e saídas, apresentada por Schmidt e Lunze (2014), os conceitos teóricos para o desenvolvimento de CTF que incorporam a possibilidade de tratar uma ou múltiplas falhas e que podem ser usados tanto em sistemas sob observação total como em sistemas sob observação parcial, definidos por Shu e Lin (2013), a síntese do CTF por meio de algoritmos baseados em modelos, apresentada por Dai, Karimoddini e Lin (2016), a metodologia de CTF baseada em autômatos temporizados, proposta por Niguez, Amari e Faure (2017) e o conceito de controlador-diagnosticador ativo que exerce as ações de diagnóstico da falha e de desabilitação de eventos controláveis, proposto por Moreira e Leal (2020).

Figura 13 – Arquitetura de controle tolerante a falhas de abordagem ativa.



Fonte: Adaptada de Moor (2016).

Denominamos de ocultação de falhas uma subclasse de estratégias de CTF pertencentes ao conjunto de abordagens ativas, na qual um bloco de reconfiguração adapta o controle que atua com o comportamento nominal da planta industrial para atuar com o seu comportamento faltoso, ocultando o efeito da falha (MOOR, 2016). Exemplos de projetos de controle nesse contexto são apresentados por Wittmann, Richter e Moor (2012) e Wittmann, Richter e Moor (2013).

Outros modelos de controles que se assemelham a um CTF de abordagem ativa são os modelos de controles preditivos apresentados por Majdzik et al. (2016) e Majdzik, Witczak e Lipiec (2019). Nos trabalhos, um sistema real de montagem de baterias e um sistema real de esteiras para transporte de embalagens são modelados matematicamente em álgebra MAX-PLUS. O controle preditivo dos sistemas opera por meio de análise *online* dos tempos de execução das

tarefas, atualizando o tempo e o momento de realização das tarefas com o intuito de garantir sincronização e concorrência no sistema.

3.2 TRABALHOS COM PROPOSIÇÕES DE ARQUITETURAS PARA CONTROLE TOLERANTE A FALHAS DE SISTEMAS A EVENTOS DISCRETOS

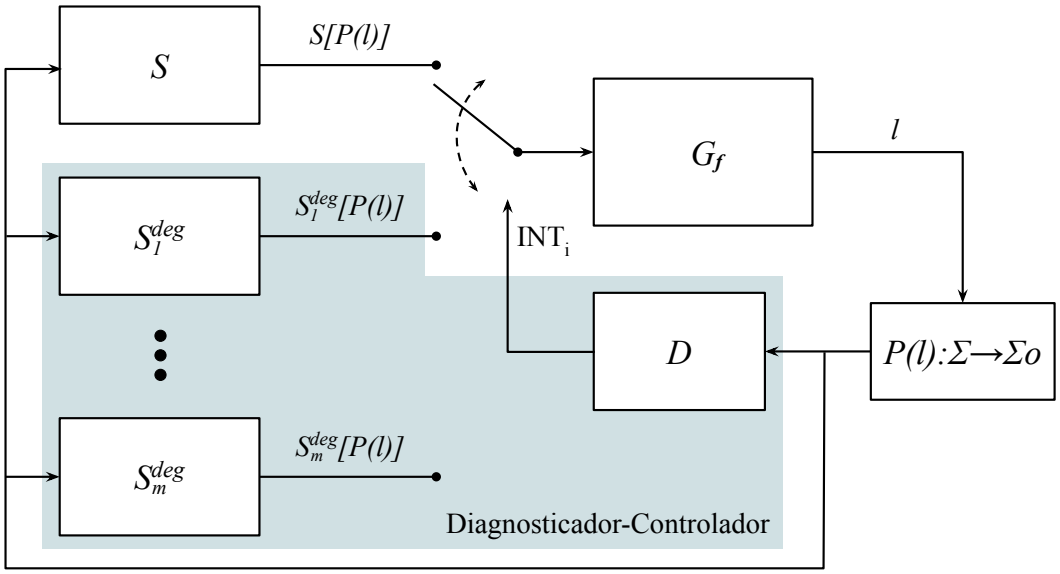
Considerando os trabalhos que foram selecionados na pesquisa realizada, 16 apresentam proposições de arquitetura para CTF de SEDs. Estas arquiteturas propostas, juntamente com a conceituação básica de operação do CTF, são descritas a seguir.

No trabalho de Wen et al. (2008), é apresentada uma proposta de CTF de abordagem passiva que foca na obtenção de um supervisor que seja tolerante a falhas e que possua a capacidade de se recuperar delas. Para o desenvolvimento do supervisor, considera-se uma especificação de controle K^N, para o comportamento nominal da planta (G^{nom}), e uma especificação de controle K, possivelmente mais liberal, para um modelo geral da planta (G_f) que representa tanto o seu comportamento nominal como o seu comportamento com falha. Além de controlar a ação do sistema, o supervisor também tem o objetivo de fazê-lo se recuperar da falha em uma sequência de eventos limitada, garantindo que haja controlabilidade segura no sistema por meio da diagnose da falha. Esta ação de recuperação do sistema permite a obtenção de uma linguagem para o supervisor que, após as ocorrências da falha e de uma sequência de eventos com o sistema operando de forma degradada, o conduz a uma operação normal, ou seja, a apresentar sequências de eventos dentro da região nominal da planta que obedeçam a especificação de controle K^N. Os autores apresentam no trabalho um exemplo de aplicação do controle em um sistema de geração de energia. No exemplo, as falhas são consideradas eventos observáveis e o controle é desenvolvido de forma assistida, dependendo da ação humana para sua modelagem. Além disso, não apresentam uma metodologia para implementação do CTF em lógica de controle.

Paoli, Sartini e Lafortune (2011) propuseram uma arquitetura de supervisão tolerante a falhas em SEDs que se fundamenta na garantia de controlabilidade segura do sistema após a diagnose de uma falha. Isto ocorre quando, após a ocorrência da falha e de sua diagnose, o CTF tem a possibilidade de alterar, por meio de um chaveamento, a ação de controle do sistema, evitando que ele execute sequências de operações que, na ação pós-falha, são consideradas proibidas. Esta arquitetura, apresentada na Figura 14, é um modelo de CTF de abordagem ativa composto por um supervisor S, que representa o comportamento em malha fechada da planta G, e por um diagnosticador-controlador capaz de efetuar a diagnose de falhas em G e mudar a supervisão do sistema. Para diagnosticar a falha f, projeta-se no diagnosticador-controlador um diagnosticador D que, para ter ação efetiva no CTF, deve possuir informações que garantam que o sistema G_f seja controlável seguro pela diagnose da falha. Na arquitetura, para que a ação de controlabilidade segura possa ser exercida, são incorporadas funcionalidades ao diagnosticador, permitindo que ele desabilite S e conduza a operação do sistema para um supervisor degradado. Dependendo da evolução de G_f e do momento em que f ocorre, a ação de interrupção de S

pode acontecer em diferentes posições do espaço de estados da planta, acarretando, portanto, o chaveamento para diferentes posições do supervisor degradado S^{deg}. Na Figura 14, o exemplo mostrado para esta arquitetura de supervisão tolerante a falhas apresenta m diferentes posições de diagnose de f no espaço de estados de G_f, gerando m sinais diferentes de interrupção ($INT_i, i = 1, \ldots, m$). Na proposta, o diagnosticador-controlador é formado por um único autômato que incorpora estados de D e os estados do supervisor degradado, possuindo então a ação de diagnose de f, a ação de interrupção de S e a ação de controle, pós-falha, do sistema.

Figura 14 – Arquitetura de supervisão tolerante a falhas proposta por Paoli, Sartini e Lafortune (2011).



Fonte: Adaptada de Paoli, Sartini e Lafortune (2011).

Todo o projeto do CTF proposto na arquitetura de supervisão é feito de forma *offline* e assistida, ou seja, depende da ação humana para sua modelagem (definição da posição de chaveamento no espaço de estados do sistema e da estratégia de controle de destino) e para a análise dos resultados obtidos (diagnosticabilidade da falha e controlabilidade segura do sistema pela diagnose). A arquitetura prevê apenas o tratamento de uma única falha por parte do CTF, sendo esta falha considerada como permanente, já que não é apresentada uma condição de chaveamento que permita o retorno ao supervisor nominal S, o que poderia acontecer caso houvesse uma manutenção corretiva que eliminasse a falha no sistema. Apesar dos autores detalharem no modelo do diagnosticador-controlador o procedimento para efetuar o chaveamento entre supervisores, eles não apresentam uma metodologia para a sua implementação em lógica de controle.

Um conceito diferente de estruturação de CTF, baseando-se em modelos com acomodação de falhas, foi proposto por Wittmann, Richter e Moor (2012). No estudo, os autores transformaram a resolução do problema de desenvolvimento de um CTF na obtenção de um supervisor, por meio da TCS, capaz de controlar o sistema em malha fechada antes e após a

ocorrência da falha. Conforme eles descrevem, projetar o CTF em uma estrutura composta apenas por um supervisor, substitui a necessidade de se desenvolver uma estrutura mais complexa, que seja composta por diagnosticadores e módulo de chaveamento.

Na proposta, a obtenção do CTF parte da ideia de se ter uma linguagem que represente o comportamento do sistema em malha aberta com acomodação da falha (L_{fa}). Esta linguagem deve ser formada pela união de outras duas linguagens que representam, respectivamente, o comportamento nominal e o comportamento defeituoso do sistema, sendo esses dois comportamentos também em malha aberta. Aplicando-se em L_{fa} restrições que definem o comportamento defeituoso aceitável para o sistema, similares as restrições de controle existentes na TCS, é possível obter uma linguagem que os autores denominam de liguagem com comportamento aceitável (E_A). Portanto, a solução para o CTF com acomodação de falhas proposto é a linguagem não vazia $K = supC(E_A, L_{fa})$.

No estudo, são apresentados exemplos em que o CTF é modelado de forma assistida. Como é previsto o uso de um único supervisor na estrutura do CTF, sua ação deve iniciar pelo estado inicial de seu modelo, que representa o comportamento nominal da planta. Apesar dos autores classificarem o CTF como sendo de abordagem ativa, o fato de existir uma única estrutura de controle o caracteriza como sendo um CTF de abordagem passiva. A forma como a falha é acomodada no supervisor não traz garantia de que sua diagnose será efetuada de forma segura nem de que exista controlabilidade segura por sua diagnose. Conforme descrevem os autores, é possível acomodar falhas diferentes no CTF, sendo estas falhas consideradas como irreparáveis. Entretanto, não é previsto que, após uma falha, possa acontecer em algum momento alguma outra. Além disso, no estudo não foi apresentada uma metodologia para síntese do CTF em lógica de controle.

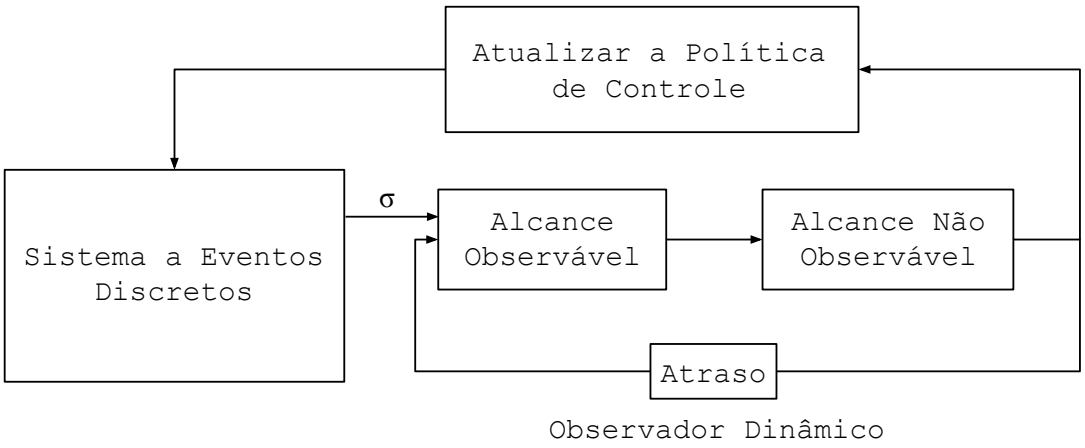
O conceito de CTF baseado em acomodação de falhas também foi explorado por Moor e Schmidt (2015). Neste estudo, os autores utilizam os requisitos de desenvolvimento de CTF com acomodação de falhas para arquiteturas de controle hierárquicas, propondo um relaxamento nos requisitos de controle em malha fechada do sistema. A condição de relaxamento do controle em malha fechada é aplicável nos casos em que as restrições, que definem o comportamento defeituoso aceitável para o sistema, acabam gerando um controle em malha fechada muito restritivo, impedindo que algumas operações desejáveis sejam executadas no sistema. A ação de relaxamento proposta é caracterizada pela minimização das restrições aplicadas no comportamento defeituoso aceitável para o sistema. Isto faz com que se obtenha uma linguagem com comportamento aceitável (E_A) menos restritiva, possibilitando que a linguagem K também possa ser menos restrita. O que foi observado no CTF proposto por Wittmann, Richter e Moor (2012) também é observado nesta proposta de CTF: sua estrutura é baseada na abordagem passiva; sua modelagem é feita de forma assistida; é possível acomodar falhas diferentes em sua estrutura, entretanto o CTF não prevê transições no sistema de um comportamento degradado para outro; não são analisadas as condições de diagnose segura da falha e de controlabilidade segura do sistema por meio da diagnose; e não é tratada a implementação do CTF em lógica de controle.

Adicionalmente, também é possível verificar na proposta que a execução de um relaxamento no comportamento defeituoso aceitável para o sistema é uma tarefa não trivial e que pode levar a condições de bloqueio na operação do sistema, sendo necessário posteriormente tratar estes bloqueios na estrutura de controle hierárquica por meio de ações associadas à recuperação de falhas.

A arquitetura de CTF de abordagem ativa apresentada por Shu e Lin (2013) propõem uma estrutura que deve atuar sobre o controle normal, que os autores consideram que seja algo já implementado no sistema, apenas quando ocorrer alguma falha. O objetivo do CTF é promover atualizações na política de controle do sistema que sejam seguras, minimamente restritivas e que atuem na desabilitação de eventos. A condição de segurança nas atualizações da política de controle deve garantir controlabilidade segura do sistema por meio da diagnose, evitando a execução de tarefas consideradas proibidas após a ocorrência das falhas. Para a análise, é criado um modelo do sistema representado por um autômato denominado de autômato extendido (G_{exd}). G_{exd} é um autômato composto, formado pelo autômato G^{nom}, que modela a operação normal do sistema, e por outros autômatos $G_{f,i}$, $i = 1, \ldots, i$, que modelam operações degradadas do sistema. Na proposta, considera-se que G_{exd} sempre inicia sua operação no estado inicial de G^{nom} e que alcança uma operação degradada $G_{f,i}$ por meio de uma determinada falha f_i. Para isso, deve existir em G_{exd} uma transição $\delta_{exd}(x_{n,0}, f_i) = x_{m,i}$, na qual $x_{n,0}$ e $x_{m,i}$ são estados que pertencem a G^{nom} e a $G_{f,i}$, respectivamente. Os autores apresentam duas formas de estruturação do CTF, uma forma levando em consideração que todos os eventos são observáveis e outra forma que considera que alguns eventos, incluindo algumas falhas, são não observáveis no sistema. Para o caso em que todas os eventos são observáveis, o CTF é projetado totalmente *offline*. Já para o caso em que há observação parcial, parte do CTF é projetada *offline* e parte é projetada *online*. A Figura 15 apresenta a estrutura da arquitetura para o caso em que há observação parcial dos eventos no sistema. Para definir as atualizações que devem ser aplicadas na política de controle, o CTF inicialmente obtém a saída atual do sistema (evento σ) e estima os estados que compõem o alcance observável do sistema. Caso na estimativa exista a certeza de uma determinada falha, calcula-se o alcance não observável do sistema, obtendo-se com isso os eventos que devem ser desabilitados, e se define a atualização na política de controle. No estudo, o CTF é desenvolvido de forma assistida e considera a possibilidade de atualizar a política de controle quando há a transição do sistema de um comportamento degradado, comportamento após a ocorrência de uma falha, para outro, comportamento após a ocorrência de uma outra falha. Para os casos em que o sistema opera sobre observação parcial, o fato da estimativa de estados ser feita *online* reduz a complexidade computacional do projeto, entretanto, os autores destacam a importância de se investigar, nessas aplicações, a complexidade computacional do cálculo do alcance não observável, que se encontra inserido na estimativa de estados. A proposta não é aplicável quando se considera recuperação da falha por uma ação externa. Além disso, não foi apresentada no estudo uma metodologia para implementação do CTF em lógica de controle.

Dando continuidade ao trabalho apresentado por Schmidt e Lunze (2014), Schuh, Zgor-

Figura 15 – Arquitetura de controle tolerante a falhas baseada em políticas de controle, proposta por Shu e Lin (2013).

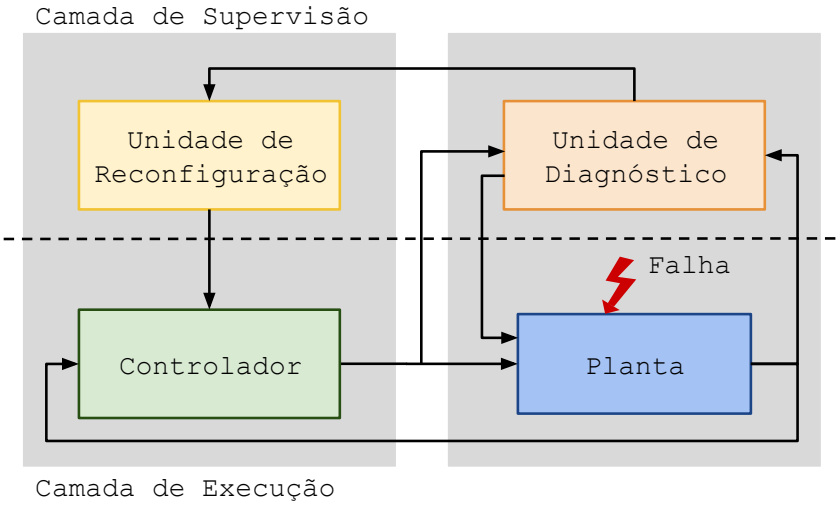


Fonte: Adaptada de (SHU; LIN, 2013).

zelski e Lunze (2015) apresentaram uma metodologia de CTF de abordagem ativa que, após a diagnose da falha, executa ações de parada e reconfiguração do controle, reconduzindo a planta a uma operação que permita a conclusão das tarefas. A metodologia utiliza modelos baseados em autômatos determinísticos com entradas e saídas e possui a arquitetura mostrada na Figura 16. A ação do CTF é dividida em etapas operacionais que são conduzidas pelas Unidades de Diagnóstico e Reconfiguração. A primeira etapa se caracteriza pela detecção da falha na planta por meio de um algoritmo que opera em paralelo com o sistema, observando se a planta apresenta alguma inconsistência em relação ao seu modelo nominal. Após a detecção da ocorrência da falha, o controlador é desconectado da planta. Em seu lugar, passa a operar um diagnosticador ativo, em malha fechada com a planta, com o intuito de realizar as identificações da falha, do estado em que ela foi detectada e do momento em que ela impactou o sistema. Após a obtenção destes dados, a Unidade de Reconfiguração reconduz a planta até o seu último estado de operação correta, antes da manifestação da falha no sistema. Por fim, o controlador é conectado novamente à planta e reconfigurado com um novo controle que permita a conclusão das tarefas. Os autores não consideram na proposta a necessidade do sistema atuar de forma segura após falha. O principal objetivo da metodologia é reconduzir o sistema a um estado operacional e pôr o controle novamente em operação. No estudo, foi realizado um experimento prático no qual a metodologia foi testada em um sistema de manipulação de peças. Seu desenvolvimento foi feito de forma assistida, sendo sua estrutura implementada em blocos no *software* Matlab Simulink. Sua ação considera que uma única falha ocorre por vez, já que a detecção e a identificação da falha acarretam em uma ação que reconduz a planta para seu último estado de operação normal, e que não há recuperação da falha por uma ação externa ao sistema.

No trabalho de Dai, Karimoddini e Lin (2016), é apresentada uma arquitetura de controle ativa que possui desenvolvimento *online* e é baseada em modificações do algoritmo L^**-learning* (ANGLUIN, 1987). A proposta considera inicialmente um modelo do sistema em autômato,

Figura 16 – Metodologia de controle tolerante a falhas proposta por Schuh, Zgorzelski e Lunze (2015).



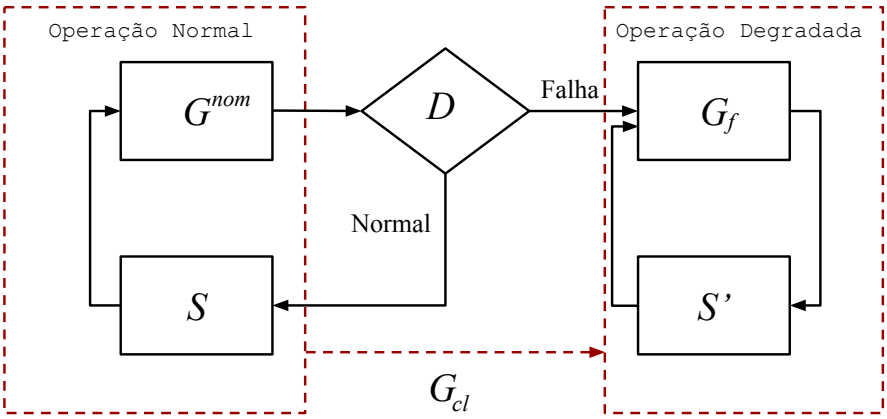
Fonte: Adaptada de (SCHUH; ZGORZELSKI; LUNZE, 2015).

que serve de base para o desenvolvimento do CTF, composto por um supervisor nominal, um diagnosticador de falhas e um supervisor pós-falha. A Figura 17 apresenta a estrutura do CTF proposto na arquitetura. Na estrutura, considera-se que G_f é o modelo geral do sistema, incluindo em seu alfabeto de eventos a falha f, que G_{cl} é o modelo do sistema em malha fechada e que G^{nom} é a parte de G que não contém falha. Portanto, a linguagem $L(G^{nom})$ não deve possuir palavras que contenham o evento f. A arquitetura proposta busca solucionar os seguintes três problemas:

1. Projetar um supervisor nominal S que garanta uma linguagem $K \subseteq L(G^{nom})$ livre de falhas;
2. Projetar um diagnosticador D que seja capaz de detectar a ocorrência da falha antes do sistema violar o requisito de segurança (linguagem $K^S \subseteq L(G_f)$), portanto, garantindo que o sistema seja controlável seguro pela diagnose;
3. Uma vez que a falha é detectada pelo diagnosticador, interromper a operação de S e reconfigurar a supervisão do sistema para o supervisor S', que garanta a linguagem K', tal que $L(S' \parallel G_f) \subseteq K' \setminus K$.

Os algoritmos de síntese baseados em aprendizado, derivados do algoritmo L^*-*learning*, são aspectos positivos da estrutura proposta, pois permitem que exista uma metodologia *online* para o desenvolvimento do CTF de forma não assistida. Entretanto, a proposta não é aplicável nos casos em que existam mais de uma falha no sistema, pois a arquitetura permite apenas que o CTF execute o chaveamento do supervisor nominal S para um supervisor S' que opere em malha fechada com um comportamento faltoso da planta. A proposta também se limita a proposição de um modelo teórico para a arquitetura, não sendo apresentada uma metodologia para sua implementação em lógica de controle.

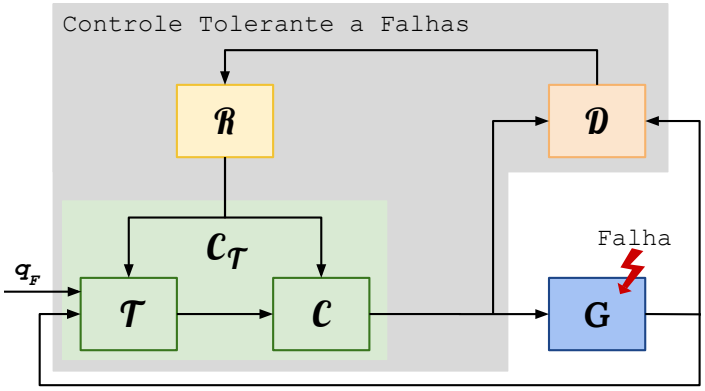
Figura 17 – Arquitetura de controle tolerante a falhas, proposta por Dai, Karimoddini e Lin (2016), com componentes desenvolvidos com base na linguagem L^*-*learning*.



Fonte: Adaptada de (DAI; KARIMODDINI; LIN, 2016).

Nas publicações de Schuh e Lunze (2016c) (2016b) (2016a) foi apresentada uma metodologia de CTF de abordagem ativa, baseada em autômatos determinísticos com entradas e saídas, que pode ser aplicada tanto para casos em que há certeza no diagnóstico da falha (SCHUH; LUNZE, 2016b), como para casos em que há ambiguidade no diagnóstico (SCHUH; LUNZE, 2016c). O CTF proposto é representado pela arquitetura apresentada na Figura 18. Seu objetivo é permitir a execução de tarefas por parte da planta G que a conduzam a um estado final q_F mesmo após a ocorrência de uma falha. Esta ação deve ser guiada pelo controlador de rastreamento do sistema C_T, composto pela unidade de planejamento de trajetória T e pelo controlador C. Em um primeiro momento, C_T possui informações para executar o controle nominal da planta. Sua ação deve sofrer alterações caso o diagnosticador D observe alguma anomalia relacionada a operação do sistema e a informe para a unidade de reconfiguração R, que, por sua vez, deve fornecer informações para o ajuste de C_T. A anomalia na operação pode gerar o diagnóstico de uma falha ou uma condição de ambiguidade no diagnóstico. A condição de ambiguidade em relação ao diagnóstico da falha se deve ao fato do diagnosticador identificar que ocorreu uma falha no sistema, mas não ter informações suficientes para identificar qual foi a falha que ocorreu. Para que esta condição de diagnóstico possa ser utilizada pelo CTF, é necessário que o conjunto de falhas, que fazem parte da condição de ambiguidade observada, influenciem de forma similar no comportamento da planta. Dessa forma, a reconfiguração a ser adotada é a mesma para qualquer uma das falhas no conjunto. Nos trabalhos publicados, os autores apresentam exemplos em que o CTF é desenvolvido de forma assistida e não consideram questões relativas à controlabilidade segura do sistema após a diagnose da falha. O desenvolvimento fica restrito a obtenção dos controles modelados por autômatos, não sendo apresentada uma representação do CTF em lógica de controle. O CTF atua de forma ativa, permitindo apenas mudanças da política de controle associada à operação normal do sistema para uma política de controle associada a uma operação degradada, sendo assim, a proposta não considera as possibilidades de ocorrência de múltiplas falhas no sistema e de recuperação dessas falhas.

Figura 18 – Arquitetura de controle tolerante a falhas proposta por Schuh e Lunze (2016c).

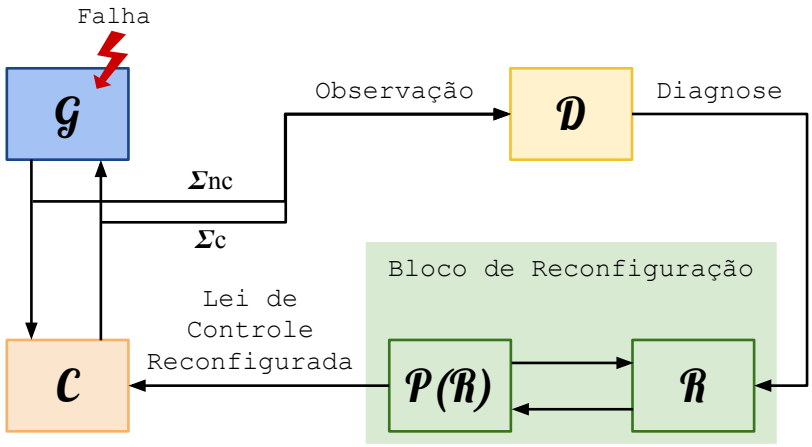


Fonte: Adaptada de (SCHUH; LUNZE, 2016c).

Niguez, Amari e Faure (2017) apresentaram uma estrutura de CTF que utiliza autômatos temporizados com guardas, com o intuito de detectar falhas que modifiquem a dinâmica temporal do sistema. Posteriormente foi apresentada por Gatwaza et al. (2020) uma aplicação prática da estrutura em uma malha de fornecimento de energia para sistemas de transporte sobre trilhos. A Figura 19 apresenta a estrutura do CTF, que possui uma arquitetura baseada na abordagem ativa. Sua ação depende que haja uma diagnose segura da falha por parte do diagnosticador D e que o sistema seja controlável de forma segura por meio dessa diagnose. A informação da diagnose é repassada para o bloco de reconfiguração, que é composto por uma unidade de reconfiguração R e um conjunto de possíveis trajetórias para o sistema $P(R)$. Com base na falha diagnosticada, a unidade de reconfiguração define a trajetória a ser adotada para a planta G. A trajetória escolhida representa uma nova lei de controle a ser adotada pelo controlador C no sistema em malha fechada. O cálculo da nova trajetória é efetuado *online* e se baseia apenas na diagnose de falhas associadas a questões temporais que indiquem a não ocorrência de um evento ou sua ocorrência de forma antecipada ou tardia. Com base nos exemplos apresentados nos trabalhos, o CTF é desenvolvido de forma assistida, tendo em sua composição um modelo que representa o comportamento global do sistema com suas restrições temporais. Nos modelos apresentados, considera-se que uma única falha acontece por vez. Sendo assim, espera-se que uma nova falha ocorra apenas após a última falha ter sido reconhecida e resolvida, com o sistema retornando à sua operação normal.

Tahiri et al. (2018) propuseram uma arquitetura de CTF baseada nas propriedades temporais dos sistemas industriais e capaz de resolver apenas um conjunto específico de problemas que podem acontecer em sensores. A ideia é que a perda de informação, ocasionada pela falha de um sensor, seja observada pelo diagnosticador no CTF, fazendo com que o CTF substitua a informação do sensor por uma informação temporal que garanta a continuidade da operação da planta. Conforme mostra a Figura 20, a arquitetura é estruturada por meio de um CTF de abordagem ativa, tendo um elemento denominado de coordenador para exercer a ação de chaveamento entre controles distribuídos. Segundo os autores, a estrutura do coordenador será apresentada

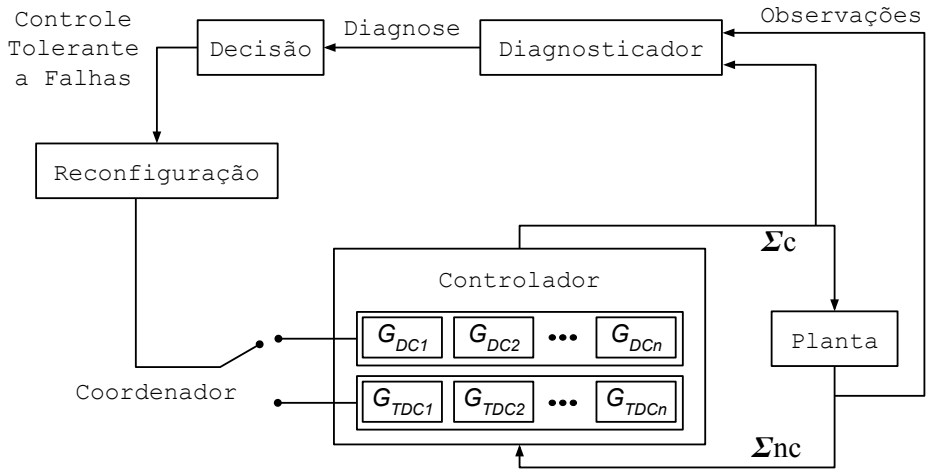
Figura 19 – Proposta de arquitetura de controle tolerante a falhas, apresentada por Gatwaza et al. (2020), baseada em autômatos temporizados.



Fonte: Adaptada de (GATWAZA et al., 2020).

em trabalhos futuros. O controle nominal da planta ($G_{DC1}, \ldots, G_{DCn}$) e o controle pós-falha ($G_{TDC1}, \ldots, G_{TDCn}$) são projetados *offline* e de forma assistida. Em específico, o controle pós-falha, denominado de controle temporizado, é projetado com base em informações temporais das respostas dos sensores no sistema industrial e possui uma metodologia de interpretação em liguagem Grafcet para implementação em controladores baseados na norma IEC 61131-3. Os autores não tratam na proposta à respeito da recuperação de falhas no sistema nem de assuntos relacionados à controlabilidade segura do sistema por meio da diagnose da falha.

Figura 20 – Arquitetura de controle tolerante a falhas baseada nas propriedades temporais dos sistemas industriais, apresentada por Tahiri et al. (2018).



Fonte: Adaptada de (TAHIRI et al., 2018).

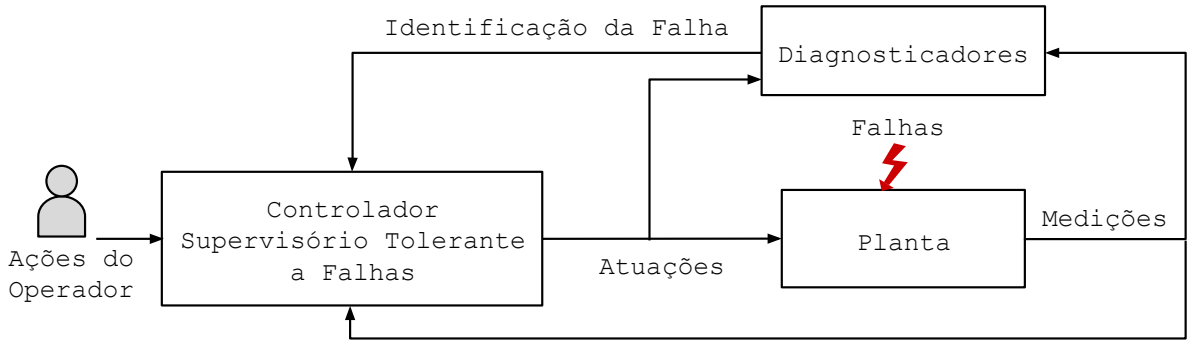
Reijnen et al. (2021) apresentaram uma metodologia de projeto de CTF composta por ações de modelagem, síntese, validação, geração de código e implementação. O estudo é uma extensão do trabalho apresentado por Reijnen et al. (2018) e é baseado em uma aplicação prática

em uma ponte giratória. A metodologia opera no desenvolvimento de um CTF de abordagem ativa produzindo, por meio da TCS, um supervisor cuja aplicação é denominada de controlador supervisório. No processo de engenharia presente na metodologia, tem-se como sequência de ações:

1. A definição dos requisitos da planta e do controlador supervisório;
2. A obtenção de um projeto e de um modelo da planta, por meio dos requisitos definidos;
3. A definição de especificações de controle, por meio do modelo da planta e dos requisitos definidos para o controlador supervisório;
4. A síntese do supervisor que origina o controlador supervisório, por meio do modelo da planta e das especificações de controle.

Caso a consistência de funcionamento do controlador supervisório seja atestada, é feita sua implementação em linguagem de texto estruturado utilizando o algoritmo proposto por Swartjes, Beek e Reniers (2014). No estudo, o modelo do CTF não leva em consideração propriedades associadas à diagnose segura e a controlabilidade segura do sistema por meio da diagnose. Seu desenvolvimento é feito de forma assistida, porém, teste, validação e implementação são realizados de forma não assitida. Parte dos requisitos que são definidos para o controlador supervisório contemplam regras associadas aos diagnósticos de falhas no sistema. Por conta disso, a metodologia trabalha com o desenvolvimento de um único supervisor com ações relacionadas à tolerância a falhas. O controle do sistema não atua de forma totalmente automática, pois parte das ações de controle são realizadas pelo operador do sistema. A diagnose de falhas no sistema é feita por sensores que medem os estados dos componentes que são acionados. Isto faz com que os diagnosticadores sejam representados por modelos simples e que prevêm a recuperação de falhas. Na metodologia, os diagnosticadores são incorporados ao modelo da planta, permitindo, conforme descrito anteriormente, a definição de requisitos relacionados às falhas. Um exemplo da arquitetura de CTF utilizada é mostrado na Figura 21.

Figura 21 – Arquitetura de controle tolerante utilizada por Reijnen et al. (2021).



Fonte: Adaptada de (REIJNEN et al., 2021).

3.3 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo apresentou uma revisão bibliográfica a respeito do controle tolerante a falhas de SEDs, trazendo na Seção 3.1 uma visão geral em relação às suas classificações e na Seção 3.2 a descrição e a análise de arquiteturas de CTF propostas na literatura.

Na análise das arquiteturas de CTF apresentadas na Seção 3.2, a revisão buscou avaliá-las com relação ao atendimento dos seguintes requisitos elencados na Seção 1.1 e que, em conjunto, produzem uma solução de CTF mais completa e mais próxima à realidade das aplicações industriais atuais:

Requisito 1: Ser um CTF de abordagem ativa;

Requisito 2: Possibilitar a atuação em cenários que permitam trocas entre políticas de controle tanto do comportamento nominal para um comportamento degradado, como de um comportamento degradado para outro. Ou seja, operar com a condição de ocorrência de mais de uma falha no sistema;

Requisito 3: Considerar a condição de recuperação da falha no sistema e, consequentemente, o retorno para o controle nominal (política de controle para o comportamento nominal);

Requisito 4: Ser desenvolvida de forma não assistida;

Requisito 5: Ter síntese em lógica de controle;

Requisito 6: Tratar do problema da controlabilidade segura do sistema por meio da diagnose da falha.

Como resultado da avaliação, obteve-se o resumo que é apresentado na Tabela 1, no qual é possível observar que nenhuma das propostas atende o conjunto de requisitos por completo. Em vista disso, identifica-se uma lacuna na área na qual a metodologia de CTF apresentada no Capítulo 4 desta tese, a seguir, procura preencher integralmente.

Tabela 1 – Resultado do atendimento por parte dos trabalhos apresentados na Seção 3.2 aos requisitos elencados.

	Requisito					
	1	2	3	4	5	6
Wen et al. (2008)			✓			✓
Paoli, Sartini e Lafortune (2011)	✓					✓
Wittmann, Richter e Moor (2012)						
Moor e Schmidt (2015)			✓			
Shu e Lin (2013)	✓	✓				✓
Schmidt e Lunze (2014) e Schuh, Zgorzelski e Lunze (2015)	✓		✓		✓	
Dai, Karimoddini e Lin (2016)	✓			✓		✓
Schuh e Lunze (2016c) (2016b) (2016a)	✓					
Niguez, Amari e Faure (2017) e Gatwaza et al. (2020)	✓		✓			✓
Tahiri et al. (2018)	✓				✓	
Reijnen et al. (2018) e Reijnen et al. (2021)	✓		✓		✓	✓

Fonte: Elaborada pelo autor (2022).

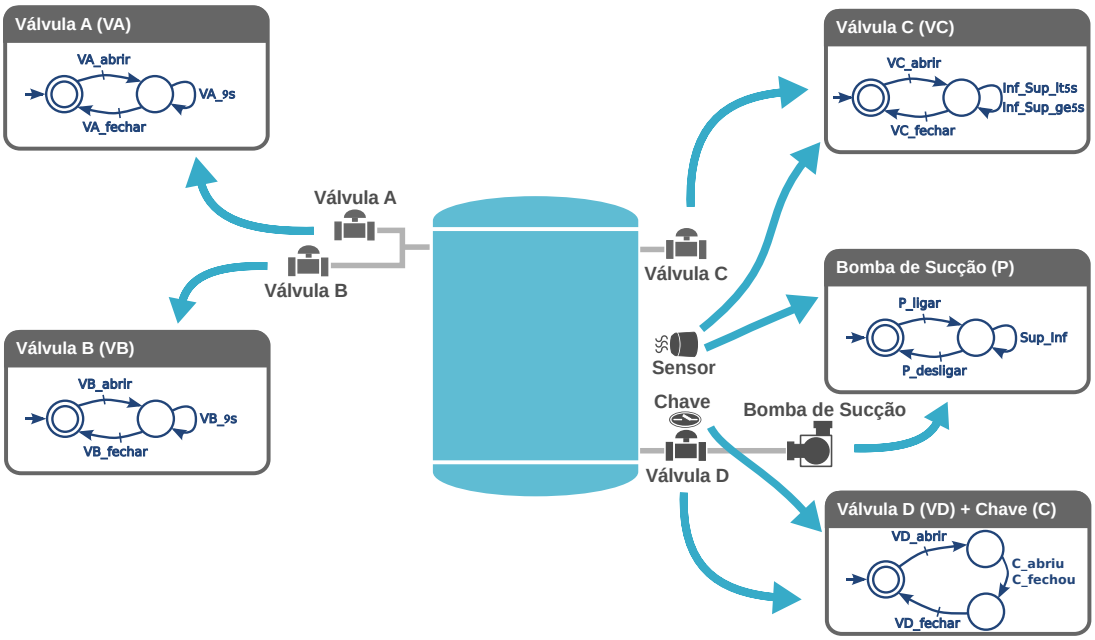
4 DIAGNOSE E CONTROLE TOLERANTE A FALHAS EM SISTEMAS A EVENTOS DISCRETOS

Este capítulo apresenta os resultados teóricos e práticos obtidos na tese, encontrando-se estruturado em quatro seções. Na Seção 4.1, é apresentado um processo industrial de mistura de substâncias químicas. Esse processo de mistura é utilizado como estudo de caso na tese, servindo para detalhar o uso da metodologia de controle tolerante a falhas que é proposta. A Seção 4.2 descreve a metodologia de projeto de controle tolerante a falhas (CTF) proposta, principal contribuição da tese. Nessa seção, a metodologia de projeto é descrita por meio: da apresentação de sua arquitetura; das análises relacionadas à diagnosticabilidade de falhas nela inseridas; da metodologia proposta para desenvolver o sistema que modela o CTF; e da metodologia proposta para implementar o CTF modelado em blocos de função da IEC 61499. A Seção 4.3 apresenta uma descrição do teste que foi executado na tese para avaliar e validar o sistema computacional desenvolvido a partir da metodologia de CTF proposta. Por fim, a Seção 4.4 é composta pelas considerações finais do capítulo.

4.1 ESTUDO DE CASO: PROCESSO DE MISTURA

O estudo de caso apresentado nesta seção é utilizado para demonstrar como a metodologia de CTF, proposta neste estudo, pode ser aplicada. O exemplo utilizado é um sistema industrial baseado em um processo de mistura, mostrado na Figura 22, que representa uma adaptação do processo apresentado por Derbel et al. (2006). O uso deste exemplo na seção é de forma didática, portanto, nem todas as possíveis falhas, que podem eventualmente ocorrer no sistema, são consideradas.

Figura 22 – Processo de mistura.



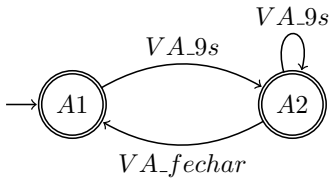
Fonte: Adaptada de (LEITÃO et al., 2020).

O sistema industrial é constituído por quatro válvulas, um sensor de concentração, uma bomba de sucção e uma chave fim-de-curso. As válvulas A (VA) e B (VB) são redundantes e responsáveis pela entrada de solvente. A válvula C (VC) é responsável por despejar soluto no tanque. Já a válvula D (VD) é utilizada na operação de drenagem do tanque, juntamente com a bomba de sucção (P). Sua ação de abertura no sistema é monitorada pela chave fim-de-curso. O sensor de concentração é um sensor de resposta booleana que atua na medição da concentração da mistura no tanque.

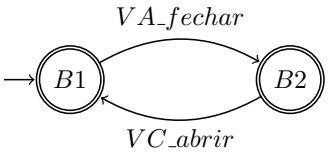
O processo de mistura trabalha de forma cíclica e contínua. Sua operação inicia com a abertura da válvula A para encher o tanque com solvente, aguardando 9 segundos com essa válvula aberta para então fechá-la e abrir a Válvula C, a fim de encher o tanque com soluto. A próxima ação no processo depende do sensor de concentração instalado, quando ele emite um dos sinais de nível de concentração alto (*Inf_Sup_lt5s* ou *Inf_Sup_ge5s*), indicando que a concentração da mistura alcançou o nível desejado para o processo, a válvula C fecha. Os sinais *Inf_Sup_lt5s* e *Inf_Sup_ge5s* representam, respectivamente, que o nível de concentração foi alcançado em um tempo menor que 5 s e em um tempo maior ou igual a 5 s. Com a mistura pronta, o próximo passo é drenar o tanque. Para isso, sequencialmente, a válvula D (VD) abre, a chave C emite um sinal *C_abriu* ou *C_fechou* e a bomba de sucção P liga. Assim que o tanque esvazia, o sensor de concentração emite o sinal de nível de concentração baixo (*Sup_Inf*). Com isso, a válvula D fecha e a bomba de sucção desliga, permitindo que se possa iniciar um novo ciclo. A válvula A e a válvula B são dispositivos redundantes no sistema, portanto, na ocorrência de falha em um desses dispositivos, o outro deve substituir. Com base nessas informações e nos modelos dos subsitemas mostrados na Figura 22, foi possível projetar as especificações de controle mostradas nas Figuras 23 e 24 e, por meio da TCS, obter os supervisores da Figura 25. O supervisor 1 é modelado para operar com a válvula A e o supervisor 2, com a válvula B.

A seguir, três falhas são estudadas na análise do processo de mistura: falha de vazamento na válvula A (evento f_{VA}), falha de vazamento na válvula B (evento f_{VB}) e falha de travamento fechado na válvula D (evento f_{VD}). Nas falhas de vazamento, considera-se que parte do soluto é despejado fora do tanque no período em que as válvulas permanecem abertas. Os comportamentos dessas válvulas com as falhas e como estas falhas afetam a ocorrência de certos eventos do sistema estão representados na Figura 26.

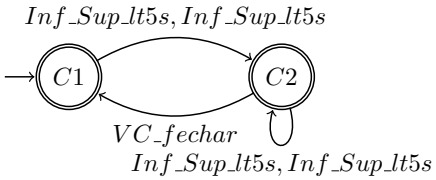
Figura 23 – Conjunto de especificações de controle para a operação do processo de mistura com a válvula *A*.



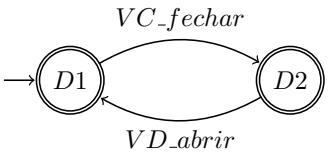
(a) Especificação de controle *A*, com $\Sigma_c = \{VA\_fechar\}$.



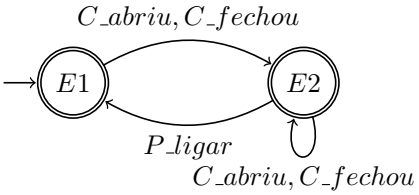
(b) Especificação de controle *B*, com $\Sigma_c = \{VA\_fechar, VC\_abrir\}$.



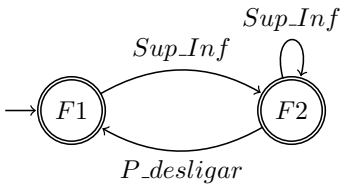
(c) Especificação de controle *C*, com $\Sigma_c = \{VC\_fechar\}$.



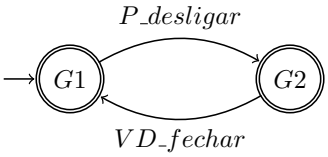
(d) Especificação de controle *D*, com $\Sigma_c = \{VC\_fechar, VD\_abrir\}$.



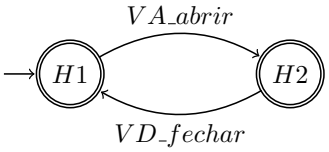
(e) Especificação de controle *E*, com $\Sigma_c = \{P\_ligar\}$.



(f) Especificação de controle *F*, com $\Sigma_c = \{P\_desligar, VD\_abrir\}$.



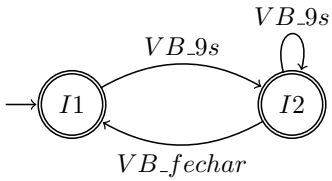
(g) Especificação de controle *G*, com $\Sigma_c = \{P\_desligar\}$.



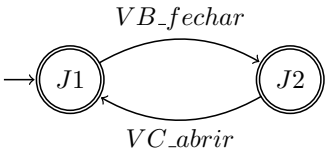
(h) Especificação de controle *H*, com $\Sigma_c = \{VA\_abrir, VD\_fechar\}$.

Fonte: Elaborada pelo autor (2022).

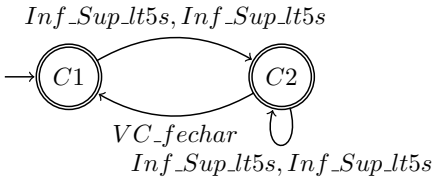
Figura 24 – Conjunto de especificações de controle para a operação do processo de mistura com a válvula *B*.



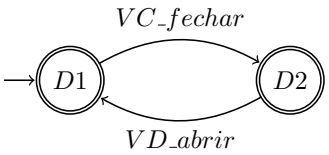
(a) Especificação de controle *I*, com $\Sigma_c = \{VB\_fechar\}$.



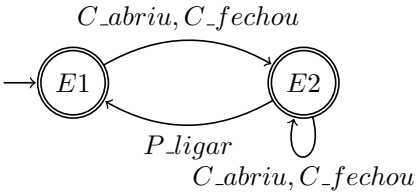
(b) Especificação de controle *J*, com $\Sigma_c = \{VB\_fechar, VC\_abrir\}$.



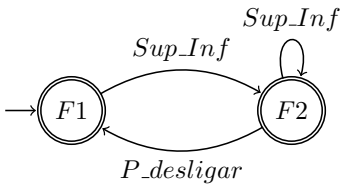
(c) Especificação de controle *C*, com $\Sigma_c = \{VC\_fechar\}$.



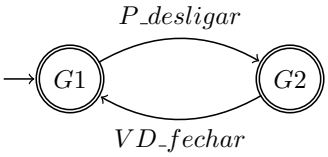
(d) Especificação de controle *D*, com $\Sigma_c = \{VC\_fechar, VD\_abrir\}$.



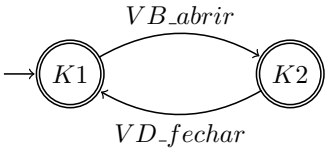
(e) Especificação de controle *E*, com $\Sigma_c = \{P\_ligar\}$.



(f) Especificação de controle *F*, com $\Sigma_c = \{P\_desligar, VD\_abrir\}$.



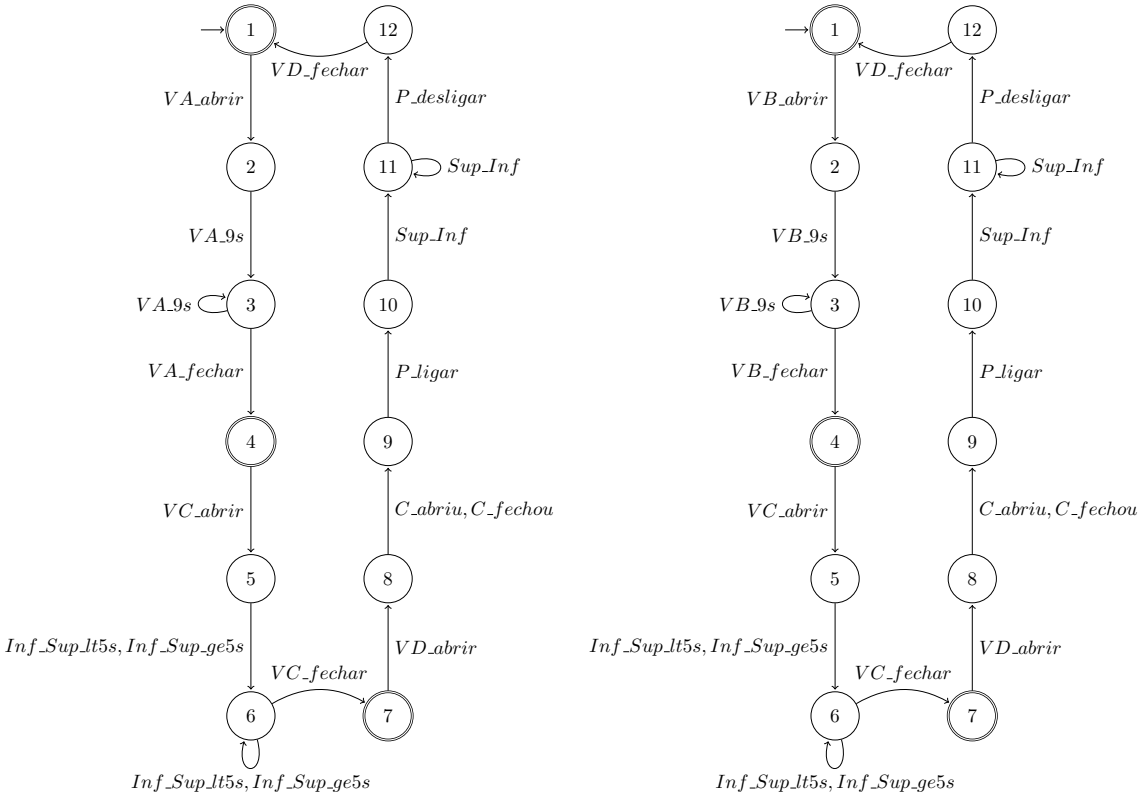
(g) Especificação de controle *G*, com $\Sigma_c = \{P\_desligar\}$.



(h) Especificação de controle *K*, com $\Sigma_c = \{VB\_abrir, VD\_fechar\}$.

Fonte: Elaborada pelo autor (2022).

Figura 25 – Supervisores do processo de mistura.

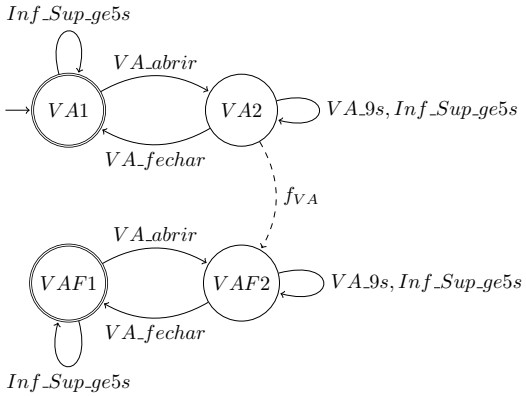


(a) Supervisor 1 (S_1), com $\Sigma_{nc} = \{VA\_9s, Inf\_Sup\_ge5s, Inf\_Sup\_lt5s, Sup\_Inf, C\_abriu, C\_fechou\}$.

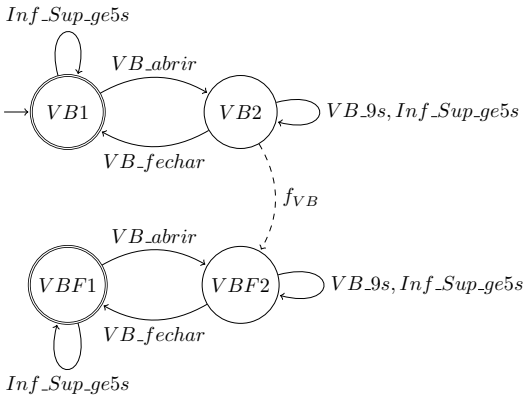
(b) Supervisor 2 (S_2), com $\Sigma_{nc} = \{VB\_9s, Inf\_Sup\_ge5s, Inf\_Sup\_lt5s, Sup\_Inf, C\_abriu, C\_fechou\}$.

Fonte: Elaborada pelo autor (2022).

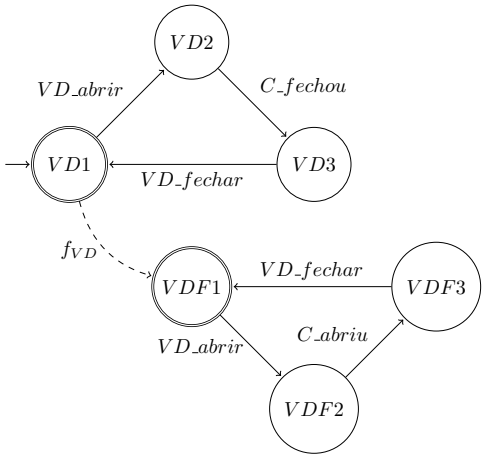
Figura 26 – Modelos que representam os componentes do sistema com falhas e como as falhas afetam a ocorrência de eventos do sistema.



(a) Modelo que representa a válvula *A* com a possível ocorrência da falha de vazamento f_{VA} e como esta falha afeta a ocorrência dos eventos Inf_Sup_ge5s e Inf_Sup_lt5s (autômato $G_{f_{VA}}$).



(b) Modelo que representa a válvula *B* com a possível ocorrência da falha de vazamento f_{VB} e como esta falha afeta a ocorrência dos eventos Inf_Sup_ge5s e Inf_Sup_lt5s (autômato $G_{f_{VB}}$).



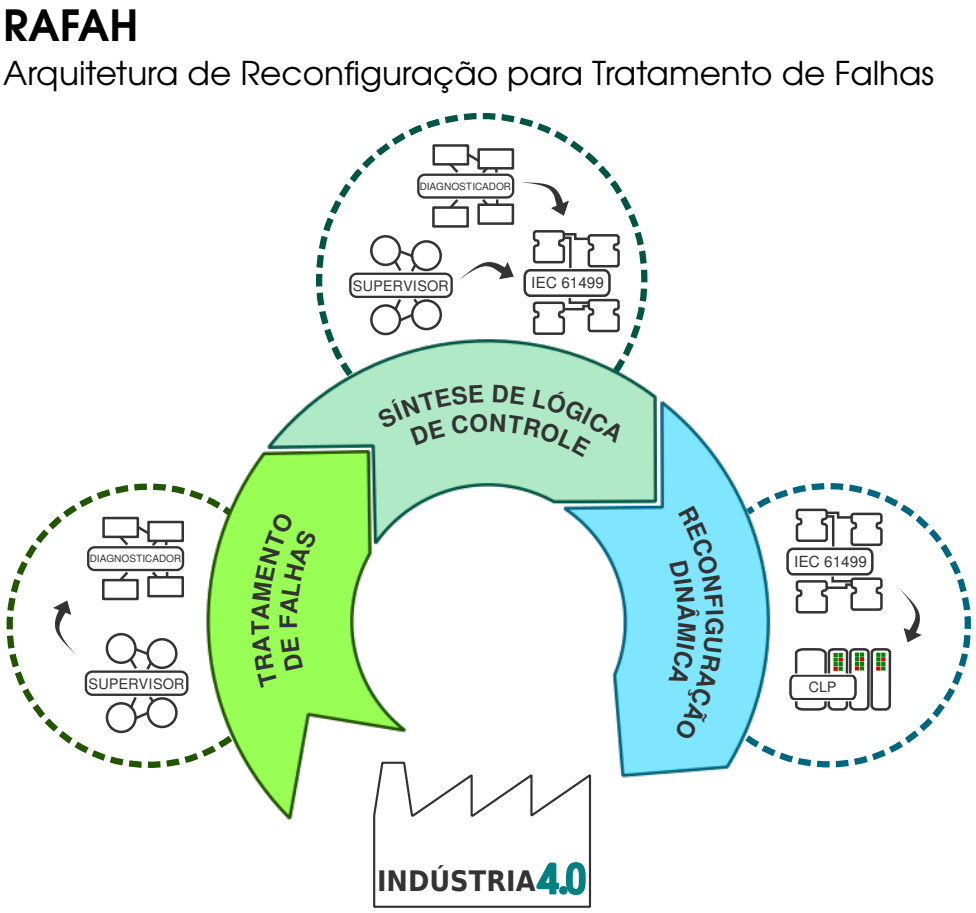
(c) Modelo que representa a válvula *D* com a possível ocorrência da falha de travamento fechado f_{VD} e como esta falha afeta a ocorrência dos eventos C_abriu e C_fechou (autômato $G_{f_{VD}}$).

Fonte: Elaborada pelo autor (2022).

4.2 METODOLOGIA DE PROJETO DE CONTROLE TOLERANTE A FALHAS

A metodologia proposta na tese está vinculada à RAFAH em sistemas de produção automatizados (SPA). Conforme mostra a Figura 27, RAFAH é composta por três estágios de operação denominados: tratamento de falhas, síntese de lógica de controle e reconfiguração dinâmica (LEITÃO et al., 2020).

Figura 27 – RAFAH, arquitetura de reconfiguração para tratamento de falhas em sistemas de produção automatizados.



Fonte: Adaptada de (LEITÃO et al., 2020).

O estágio de tratamento de falhas tem como objetivos avaliar como o SPA em análise se comporta, perante a ocorrência das falhas, e modelar uma estrutura flexível de controle para o sistema, permitindo, quando possível, o uso de uma estratégia de controle específica para cada falha diagnosticada. Sua ação é operacionalizada por um sistema que analisa a condição de diagnosticabilidade das falhas, denominado de sistema de análise da diagnosticabilidade de falhas, *Fault Analysis System* (FAS), e por um sistema que efetua a síntese do modelo do CTF, denominado de sistema de síntese de controle tolerante a falhas, *Fault Tolerant Control Synthesis System* (FTCSS).

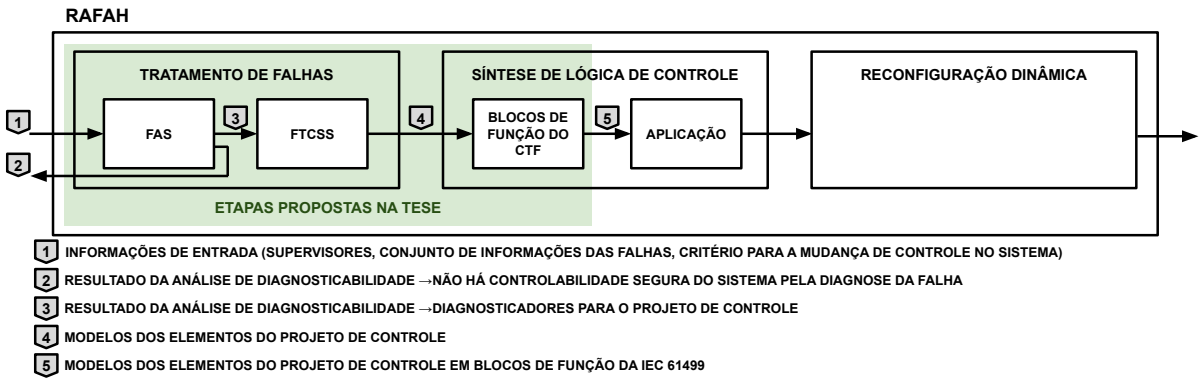
No estágio de síntese de lógica de controle, os elementos componentes do CTF, obtido no estágio de tratamento de falhas, são convertidos em blocos de função da norma IEC 61499.

Os blocos de função são incorporados posteriormente a uma rede de blocos, que na IEC 61499 é chamada de aplicação, gerando o projeto global de controle do sistema a ser transferido para o controlador.

A reconfiguração dinâmica é a operação que põe o sistema de controle projetado em atividade no SPA. Sua execução é baseada no conceito de reconfigurabilidade presente na IEC 61499, sendo provida por blocos de função administrativos. Para efetuar a reconfiguração, RAFAH deve ter um sistema de reconfiguração com uma composição específica de blocos administrativos e que permita, em tempo de execução, o carregamento da rede de blocos do CTF nos controladores do sistema.

A estrutura interna proposta para a arquitetura de reconfiguração RAFAH é detalhada na Figura 28. As etapas destacadas com fundo verde na figura representam as partes da arquitetura de reconfiguração RAFAH desenvolvidas nesta tese. A arquitetura possui como entradas uma lista de informações das falhas, uma lista de supervisores e o critério a ser adotado para mudança do controle no sistema. A lista de falhas é composta pelas falhas que devem ser tratadas pelo CTF no sistema. A lista de supervisores é composta por supervisores monolíticos que podem ser utilizados no controle do SPA. A composição da lista de supervisores deve seguir uma ordem baseada na prioridade para fins de controle no sistema, indo do supervisor que tem maior prioridade para o supervisor que tem menor prioridade. O critério adotado para a mudança do controle depende da proposta a ser adotada no projeto de automação. Entende-se que o momento mais adequado para a mudança de estratégia de controle por parte do CTF é algo singular de cada projeto e, portanto, torna-se interessante a adoção de um critério específico para cada caso.

Figura 28 – Detalhamento dos estágios de operação da arquitetura de reconfiguração RAFAH.



Fonte: Elaborada pelo autor (2020).

4.2.1 Modelo de Controle Tolerante a Falhas

O modelo de CTF utilizado na metodologia proposta é baseado na arquitetura de supervisão tolerante a falhas apresentada por Paoli, Sartini e Lafortune (2011), que foi descrita na Seção 3.2. Embora sejam utilizados no modelo os mesmos conceitos e procedimentos que foram

utilizados por Paoli, Sartini e Lafortune na diagnose da falha, sua estrutura e sua capacidade de ação sobre o sistema diferem em relação à arquitetura que eles apresentaram.

A Figura 29 mostra a estrutura do modelo do CTF proposto na tese. Diferentemente do que ocorre na arquitetura de supervisão apresentada por Paoli, Sartini e Lafortune, que mostra a ação do CTF para apenas uma falha no sistema, a ação do CTF proposto não fica restrita a uma única falha no sistema, inclusive sendo possível monitorar falhas diferentes em condições diferentes de operação do sistema. No modelo proposto, não existe um componente diagnosticador-controlador. O chaveamento é exercido por um componente denominado de módulo de chaveamento, sendo a ação de habilitação, ou desabilitação, efetuada no supervisor e no conjunto de diagnosticadores seguros[1] que operam em conjunto com o supervisor. Na proposta, considera-se que as falhas não afetam simultaneamente o sistema e que sejam do tipo permanente, ou seja, a recuperação depende de uma manutenção corretiva no componente defeituoso. Portanto, a estrutura lógica do módulo de chaveamento possibilita o retorno do sistema a um modelo de supervisão, anteriormente desabilitado por conta de falha, quando se tem a recuperação do componente que a apresentou. Os diagnosticadores que compõem o modelo possuem sinais de interação com o módulo de chaveamento e com o supervisor ao qual estão relacionados. Para o módulo de chaveamento, os sinais provenientes dos diagnosticadores representam a informação do momento no qual o chaveamento deve acontecer por conta da diagnose de uma falha. E para os supervisores, os sinais operam como parte integrante da estrutura de controle, sendo algo específico para a proposta de síntese do modelo na IEC 61499.

4.2.2 Sistema de Análise da Diagnosticabilidade de Falhas

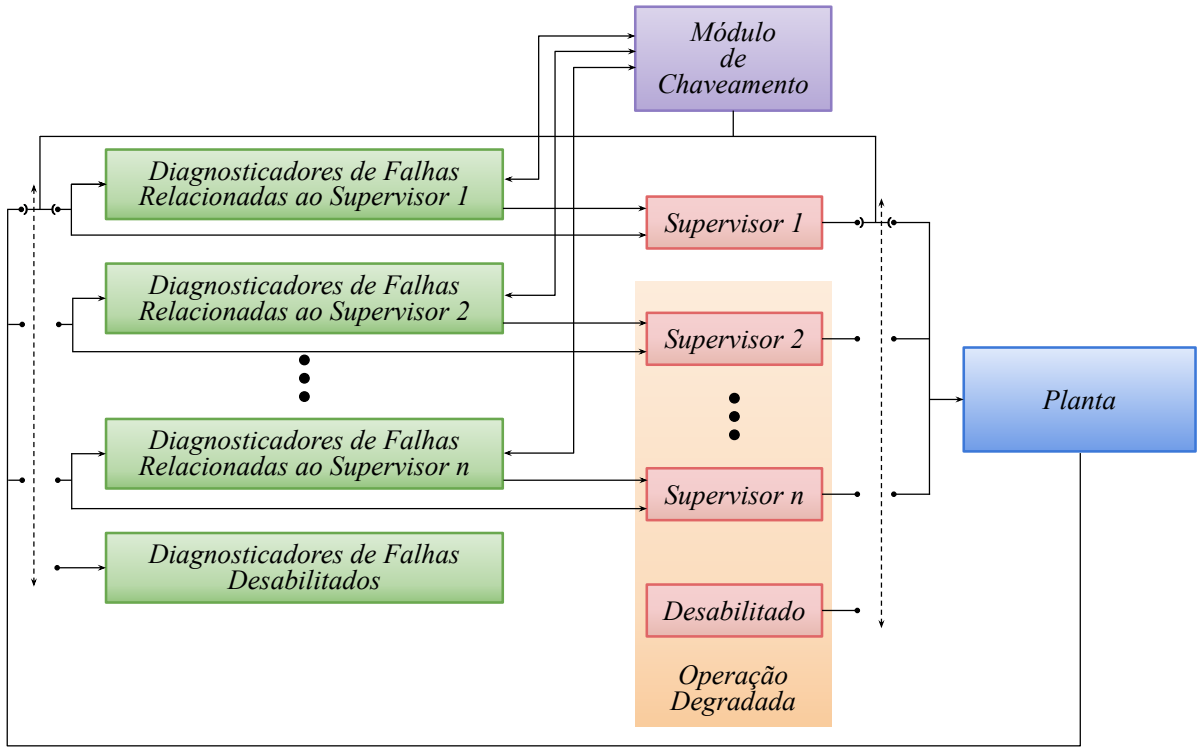
O FAS é uma etapa da arquitetura de reconfiguração RAFAH que sistematiza o diagnóstico das falhas no SPA por meio de uma sequência de operações relacionadas com a diagnosticabilidade de falhas em SEDs. Ele pode ser classificado como um sistema especialista, pois opera um conjunto de regras de análise que fornecem informações referentes às falhas, se é diagnosticável, não diagnosticável, diagnosticável segura, ou se permite controlabilidade segura no sistema por meio da diagnose. Também, quando a análise aponta a possibilidade de controlabilidade segura no sistema, ele produz um conjunto de informações que contribuem para a síntese do modelo do CTF, realizada na etapa subsequente da arquitetura.

O FAS tem sua operação descrita na sequência de algoritmos que vai do Algoritmo 1[2] ao Algoritmo 7. Sua ação depende de uma lista de informações das falhas F_{lista}, de uma lista de supervisores S_{lista} (ordenados por prioridade), pelo critério Cr a ser adotado para a mudança de controle no sistema e pelos autômatos dos supervisores, dos componentes com as representações das falhas e dos rotuladores. A lista de informações das falhas é um conjunto cujos elementos

[1] Por questão de simplificação, de agora em diante, no texto, o termo diagnosticador seguro passará a ser chamado simplesmente de diagnosticador, ou diagnosticador D.

[2] Nestes e nos demais algoritmos apresentados na tese, todas as variáveis declaradas de forma literal serão apresentadas em itálico.

Figura 29 – Modelo de controle tolerante a falhas proposto no estudo.



Fonte: Elaborada pelo autor (2021).

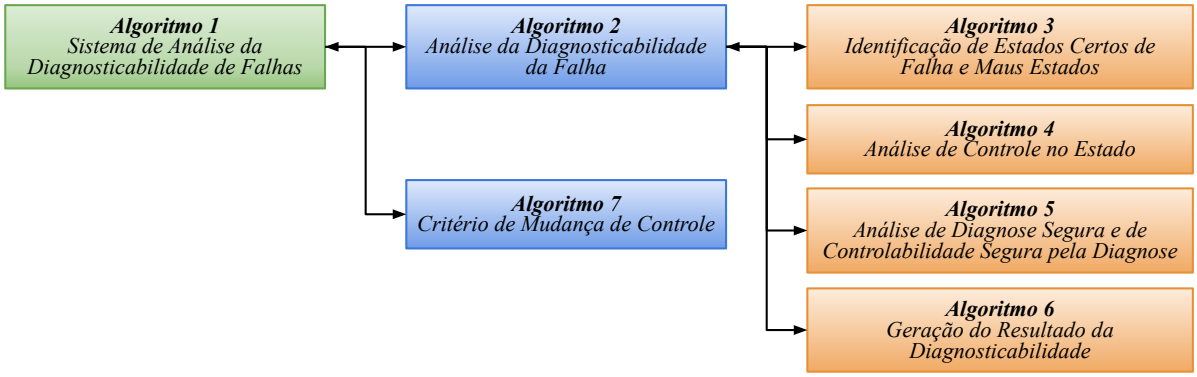
Inf_f são quádruplas formadas:

- pela identificação da falha f;
- pela identificação do modelo do componente do sistema com a representação da falha G_f;
- pela identificação do supervisor S ($S \in S_{lista}$) ao qual o componente susceptível a falha pertence e cuja diagnosticabilidade da falha é importante;
- e pela identificação do autômato rotulador da falha f e reconhecedor de palavras proibidas R_{l_f} que serve para detalhar a condição do sistema sob ação da falha.

O diagrama apresentado na Figura 30 mostra como o FAS se encontra modularizado. Para a análise da diagnosticabilidade de cada falha f no sistema, o Algoritmo 2 inicialmente calcula o autômato $A_{l_f} = (X_{l_f}, \Sigma_{l_f}, \delta_{l_f}, \Gamma_{l_f}, x_{l_{0_f}})$, rotulado com informações referentes à f no comportamento do sistema controlado, e o diagnosticador $D = (X_D, \Sigma_o, \delta_D, \Gamma_D, x_{D_0})$. Na sequência, é avaliada a condição de diagnosticabilidade de f. A avaliação inicia com a verificação da existência de ciclos indeterminados observáveis ou de ciclos indeterminados escondidos em D (linha 4 do Algoritmo 2). Posteriormente, no Algoritmo 3, identifica-se no diagnosticador D os estados certos de falha, estados com marcação Y (*Yes*) no rótulo, e os maus estados, estados com marcação B (*Bad*) no rótulo e que correspondem, na análise da diagnose segura, às situações nas

quais uma sequência proibida foi executada após a falha. Para os estados que sejam classificados como sendo certos de falha e que não sejam maus estados, o Algoritmo 4 avalia a possibilidade de ação de controle seguro nesses estados (ação de chaveamento, por parte do CTF, antes que o sistema execute uma subpalavra proibida), fazendo uma classificação dos estados que se enquadrem nessa situação. As informações obtidas nas etapas anteriores servem de suporte para as análises finais de diagnose segura e de controlabilidade segura no sistema, realizadas no Algoritmo 5, sendo posteriormente gerado o resultado da diagnosticabilidade da falha f no Algoritmo 6.

Figura 30 – Diagrama de modularização das operações executadas pelo sistema de análise da diagnosticabilidade de falhas (FAS). Da esquerda para a direita, as setas indicam chamadas de funções (algoritmos) e, da direita para a esquerda, indicam retorno das funções.



Fonte: Elaborada pelo autor (2022).

Também faz parte do FAS a obtenção de dados para efetuar a controlabilidade segura no sistema (linha 6 do Algoritmo 1). Esta ação depende do critério Cr adotado e tem como resultado um conjunto de dados CS_f para a controlabilidade segura do sistema por meio da diagnose de f. Todas as informações, relacionadas a diagnosticabilidade de f, são agrupadas em um conjunto denominado de *Informações da Diagnosticabilidade de* f (linhas 8 ou 10 do Algoritmo 1). Posteriormente, os resultados de diagnosticabilidade de todas as falhas, assim como várias outras informações obtidas na análise, são organizados pelo FAS em um conjunto de informações (linha 11 do Algoritmo 1). Portanto, como saída do FAS, para cada falha f, além do resultado da diagnosticabilidade, para os casos em que é possível exercer no sistema a ação de controlabilidade segura pela diagnose, têm-se:

- o diagnosticador D;
- um conjunto dos primeiros maus estados $CPME = \{x_D : [x_D \in X_D^{Y,B}] \wedge [\exists x'_D \in X_D^{Y,nB}$ para o qual $\delta(x'_D, \sigma) = x_D!]\}$, sendo $X_D^{Y,B}$ o conjunto dos estados certos de falha (estados com marcação Y) que são maus estados (estados com marcação B) e $X_D^{Y,nB}$ o conjunto dos estados certos de falha que não são maus estados (estados com marcação nB);

- um conjunto de estados passíveis de controle $CEPC = \{x_D : [x_D \in X_D^{Y,nB}] \wedge [\forall s\sigma t$, com $s,t \in \Sigma^*$, se $\delta_D(x_D, s\sigma t) = x'_D, x'_D \in X_D^{Y,B}$, então $\sigma \in \Sigma_c]\}$, que representam estados certos de falha nos quais se pode exercer a ação de controlabilidade segura no sistema;
- um conjunto de dados CS_f para a execução da controlabilidade segura no sistema;
- e o conjunto de maus estados de D.

Algoritmo 1: Sistema de Análise da Diagnosticabilidade de Falhas

Entrada: (F_{lista}, S_{lista}, Cr, *Supervisores*, *Modelos dos Componentes com as Falhas*, *Autômatos Rotuladores*)
Saída: *Informações das Diagnosticabilidades das Falhas*, *Diagnosticadores*

1 **início**
2 *Informações da Diagnosticabilidade de* $f \leftarrow \emptyset$
3 **para todo** $Inf_f \in F_{lista}$ **faça**
`/* Função de Análise da Diagnosticabilidade da Falha (Algoritmo 2) */`
4 (*Resultado da Diagnosticabilidade*, *D*, *CPME*, *CEPC*, *Maus Estados*) $\leftarrow$ *Análise da Diagnosticabilidade da Falha* (Inf_f)
5 **se** *Resultado da Diagnosticabilidade = "O sistema é controlável seguro pela diagnose"* **então**
`/* Função de Análise do Critério de Mudança de Controle (Algoritmo 7) */`
`/* CS_f → Dados de Controlabilidade Segura */`
6 $CS_f \leftarrow$ *Critério de Mudança de Controle* (*D*, *CPME*, *CEPC*)
7 obter a identificação do componente que apresenta a falha f ($Comp_f$)
8 *Informações da Diagnosticabilidade de* $f \leftarrow \{f, Comp_f, S$, *Resultado da Diagnosticabilidade*, *D*, *CPME*, *CEPC*, CS_f, *Maus Estados*$\}$
9 **senão**
10 *Informações da Diagnosticabilidade de* $f \leftarrow \{f, S$, *Resultado da Diagnosticabilidade*$\}$
11 *Informações das Diagnosticabilidades das Falhas* $\leftarrow$ *Informações das Diagnosticabilidades das Falhas* $\cup$ {*Informações da Diagnosticabilidade de* f}

O Exemplo 1 a seguir mostra como a metodologia proposta para o FAS pode ser utilizada na análise da diagnosticabilidade das falhas existentes no processo de mistura da Seção 4.1.

Exemplo 1. Neste exemplo, o sistema de análise da diagnosticabilidade de falhas é utilizado para avaliar as condições de diagnosticabilidade das falhas do processo de mistura da Figura 22. São utilizadas como entradas para o sistema $F_{lista} = \{(f_{VA}, G_{f_{VA}}, S_1, A_{l_{f_{VA}}}), (f_{VD}, G_{f_{VD}}, S_1, A_{l_{f_{VD}}}), (f_{VB}, G_{f_{VB}}, S_2, A_{l_{f_{VB}}}), (f_{VD}, G_{f_{VD}}, S_2, A_{l_{f_{VD}}})\}$, $S_{lista} = \{S_1, S_2\}$ e o critério Cr de manter o sistema operando até onde for possível após a detecção da falha e antes de executar uma ação ilegal. Na análise, são considerados os autômatos rotuladores mostrados na Figura 31.

Algoritmo 2: Análise da Diagnosticabilidade da Falha.

Entrada: (Inf_f)
Saída: (*Resultado da Diagnosticabilidade*, *D*, *CEF*, *CEPC*, *Maus Estados*)

início
  /* $Inf_f = (f, G_f, S, R_{l_f})$ */
  calcular $A_{l_f} = S \parallel G_f \parallel R_{l_f}$ /* Autômato Rotulado */
  calcular $D = Obs(A_{l_f}, \Sigma_o)$ /* $D = (X_D, \Sigma_o, \delta_D, \Gamma_D, x_{D_0})$ */
  verificar a existência de ciclos indeterminados (*CI*) e de ciclos indeterminados escondidos (*CIE*)
  se $\exists(CI \vee CIE)$ **então**
    Resultado da Diagnosticabilidade = "f é não diagnosticável"
  senão
    /* Função de Identificação de Estados Certos de Falha e Maus Estados (Algoritmo 3) */
    (*Controlabilidade Segura, Estados Certos de Falha, Maus Estados, CPME*) ← *Identificação de Estados Certos de Falha e Maus Estados* (X_D)
    se *Controlabilidade Segura* = *VERDADEIRO* **então**
      para todo $x_D \in X_D$ **faça**
        se $(x_D \in$ *Estados Certos de Falha*$) \wedge (x_D \notin$ *Maus Estados*$)$ **então**
          /* Função de Análise de Controle no Estado (Algoritmo 4) */
          /* *CEA* = Conjunto de estados em análise */
          (*Estado Passível de Controle, CEA, CEPC, CENPC*) ← *Análise de Controle no Estado* (*CEA*, *CEPC*, *CENPC*, *Maus Estados*, x_D)
    /* Função de Análise de Diagnose Segura e de Controlabilidade Segura pela Diagnose (Algoritmo 5) */
    (*Diagnose Segura, Controlabilidade Segura*) ← *Análise de Diagnose Segura e de Controlabilidade Segura pela Diagnose (D, CPME, CEPC, Controlabilidade Segura)*
    /* Função de Geração do Resultado da Diagnosticabilidade (Algoritmo 6) */
    Resultado da Diagnosticabilidade ← *Geração do Resultado da Diagnosticabilidade (Diagnose Segura, Controlabilidade Segura)*

Algoritmo 3: Identificação de Estados Certos de Falha e Maus Estados.

Entrada: (X_D)
Saída: (*Controlabilidade Segura*, *Estados Certos de Falha*, *Maus Estados*)

```
1  início
2      CPME ← ∅
3      Maus Estados ← ∅
4      Estados Certos de Falha ← ∅
5      Controlabilidade Segura ← VERDADEIRO
6      para todo x_D ∈ X_D faça
7          se ∃{{_,N,nB},{_,Y,B}} ⊆ x_D então
8              Controlabilidade Segura ← FALSO
9              sair do laço de repetição
10         senão se ∀x_l ∈ x_D, x_l = {_,Y,_} então
11             Estados Certos de Falha ← Estados Certos de Falha ∪ {x_D}
12             se ∃{_,Y,B} ⊆ x_D então
13                 Maus Estados ← Maus Estados ∪ {x_D}
14                 para todo δ(x'_D,_) = x_D faça
15                     se (∀x'_l ∈ x'_D, x'_l = {_,Y,nB}) então
16                         CPME ← CPME ∪ {x'_D}
```

Algoritmo 4: Análise de Controle no Estado.

Entrada: (*CEA*, *CEPC*, *CENPC*, *Maus Estados*, x_D)
Saída: (*Estado Passível de Controle, CEA, CEPC, CENPC*)

```
1  início
       /* CEA = Conjunto de Estados em Análise                          */
2      CEA ← CEA ∪ {x_D}
3      Estado Passível de Controle ← VERDADEIRO
4      para todo δ(x_D,e)! faça
5          se (e ∈ Σ_nc) ∧ (δ(x_D,e) = x'_D)[x'_D ≠ x_D] então
6              se (x'_D ∈ CENPC) ∨ (x'_D ∈ Maus Estados) então
7                  Estado Passível de Controle ← FALSO
8              senão se (x'_D ∉ CEPC) ∧ (x'_D ∉ CEA) então
9                  (Estado Passível de Controle, CEA, CEPC, CENPC) ← Análise de
                    Controle no Estado (CEA,CEPC,CENPC, Maus Estados, x'_D)
10     se Estado Passível de Controle = VERDADEIRO então
11         CEPC ← CEPC ∪ {x_D}
12     senão se ∃CENPC então
13         CENPC ← CENPC ∪ {x_D}
14     senão
15         CENPC ← {x_D}
16     CEA ← CEA − {x_D}
```

Algoritmo 5: Análise de Diagnose Segura e de Controlabilidade Segura pela Diagnose.

Entrada: (*D*, *CEF*, *CEPC*, *Controlabilidade Segura*)
Saída: *Diagnose Segura, Controlabilidade Segura*

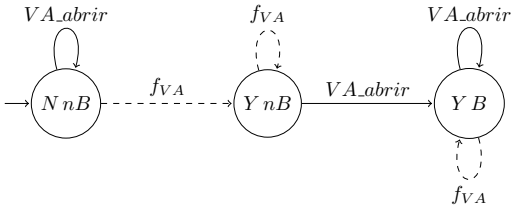
```
1  início
2      obter o diagnosticador D                          /* D = (X_D, Σ_o, δ_D, Γ_D, x_D0) */
3      Diagnose Segura = VERDADEIRO
4      para todo x_F ∈ CEF faça
5          Conjunto de Estados a Avaliar ← ∅
6          para todo δ_D(x_D, _) = x_F faça
7              se x_D ≠ x_F então
8                  se (∀x_l ∈ x_D, x_l = {_, Y, nB}) ∧ (x_D ∉ CEPC) então
9                      Conjunto de Estados a Avaliar ← Conjunto de Estados a Avaliar
                         ∪{x_D}
10                 senão se ∀x_l ∈ x_D, x_l = {_, N, nB} então
11                     Diagnose Segura = FALSO
12                     Controlabilidade Segura = FALSO
13         enquanto (Conjunto de Estados a Avaliar ≠ ∅) ∧ (Diagnose Segura =
             VERDADEIRO) faça
14             obter o primeiro elemento do Conjunto de Estados a Avaliar (x_i)
15             Conjunto de Estados a Avaliar ← Conjunto de Estados a Avaliar −{x_i}
16             para todo δ_D(x_D, _) = x_i faça
17                 se x_D ≠ x_i então
18                     se (∀x_l ∈ x_D, x_l = {_, Y, nB}) ∧ (x_D ∉ CEPC) então
19                         Conjunto de Estados a Avaliar ← Conjunto de Estados a Avaliar
                             ∪{x_D}
20                     senão se ∀x_l ∈ x_D, x_l = {_, N, nB} então
21                         Controlabilidade Segura = FALSO
```

Algoritmo 6: Geração do Resultado da Diagnosticabilidade.

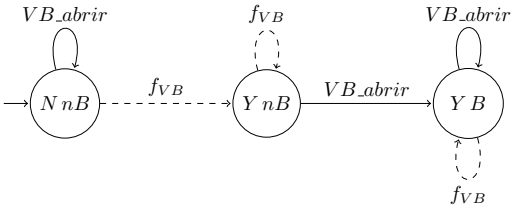
Entrada: (*Diagnose Segura*, *Controlabilidade Segura*)
Saída: (*Resultado da Diagnosticabilidade*)

```
1  início
2      se Controlabilidade Segura = VERDADEIRO então
3          Resultado da Diagnosticabilidade = "O sistema é controlável seguro pela
             diagnose"
4      senão se Diagnose Segura = VERDADEIRO então
5          Resultado da Diagnosticabilidade = "f tem diagnose segura"
6      senão
7          Resultado da Diagnosticabilidade = "f é não diagnosticável"
```

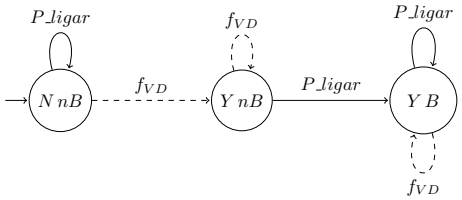
Figura 31 – Autômatos rotuladores das falhas existentes no processo de mistura da Figura 22.



(a) Autômato rotulador da falha de vazamento da válvula A ($A_{l_{f_{VA}}}$).



(b) Autômato rotulador da falha de vazamento da válvula B ($A_{l_{f_{VB}}}$).



(c) Autômato rotulador da falha de travamento fechado da válvula D ($A_{l_{f_{VD}}}$).

Fonte: Elaborada pelo autor (2022).

Com base nas informações utilizadas como entradas, o sistema calcula, para cada falha, o seu respectivo diagnosticador (Figuras 32 e 33). Após a obtenção do diagnosticador, seus estados são analisados com o objetivo de se obter no Algoritmo 3:

1. O conjunto denominado de *Estados Certos de Falha* (X_D^Y). Os estados pertencentes a este conjunto encontram-se destacados em amarelo e vermelho nos diagnosticadores das Figuras 32 e 33.

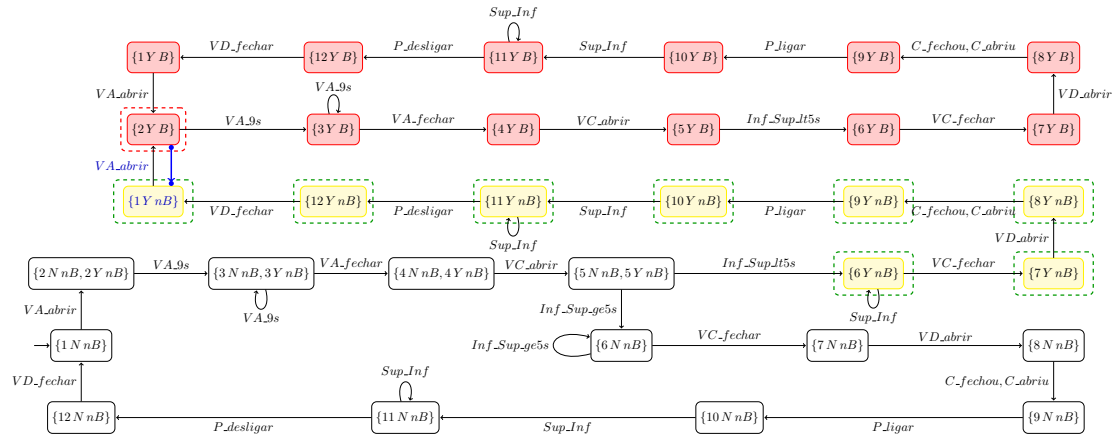
2. O conjunto de estados denominado de *Maus Estados* (X_D^B), destacados em vermelho nos diagnosticadores das Figuras 32 e 33.

3. O conjunto de estados nos quais é possível desabilitar eventos controláveis, com o intuito de exercer controlabilidade segura no sistema por meio diagnose. Este conjunto é denominado de conjunto de estados passíveis de controle ($CEPC$) e é formado pelos estados que se encontram destacados pelo tracejado verde nos diagnosticadores das Figuras 32 e 33.

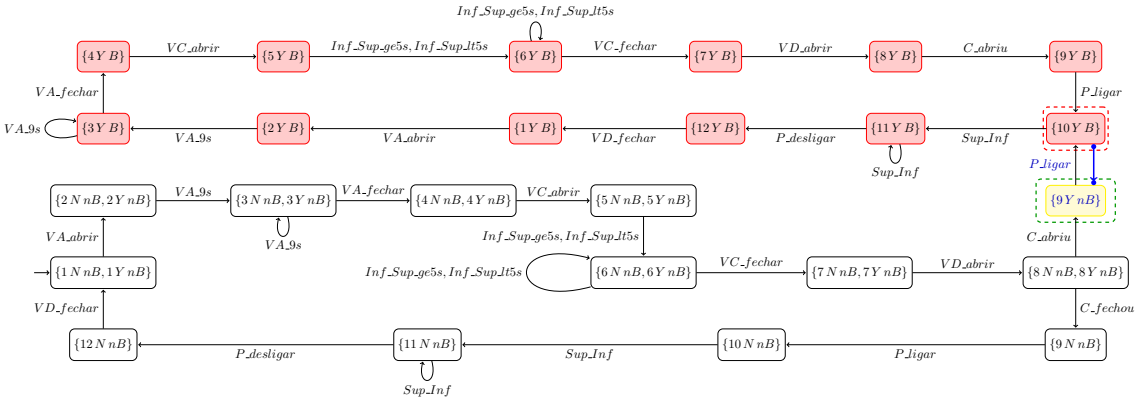
4. O conjunto de estados nos quais não é possível desabilitar eventos controláveis, com o intuito de exercer controlabilidade segura no sistema por meio diagnose, denominado de conjunto de estados não passíveis de controle ($CENPC$). Este conjunto deve ser formado por estados do conjunto $X_D^{Y,nB}$ que não pertencem a $CEPC$.

5. O conjunto dos primeiros maus estados ($CPME$), destacados pelo tracejado vermelho nos diagnosticadores das Figuras 32 e 33.

Figura 32 – Diagnosticadores de falhas relacionados ao supervisor S_1 com: estados certos de falha destacados em amarelo e vermelho; maus estados destacados em vermelho; estados passíveis de controle destacados por tracejado verde; estados que não devem ser alcançados destacados por tracejado vermelho; e sentido do fluxo de análise da diagnose segura e da controlabilidade segura no diagnosticador destacado por seta em azul.



(a) Diagnosticador da falha de vazamento na válvula A ($D_{f_{VA}}$).

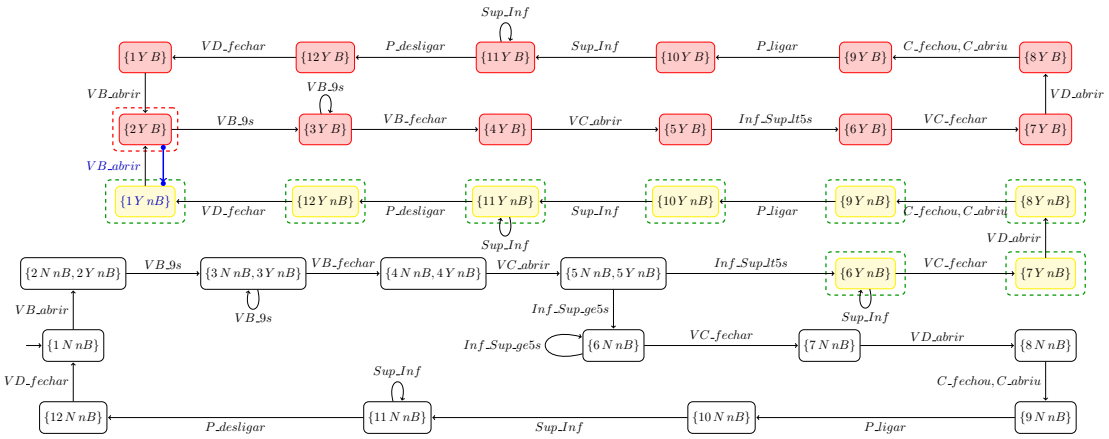


(b) Diagnosticador da falha de travamento fechado da válvula D ($D_{f_{VD_{S_1}}}$).

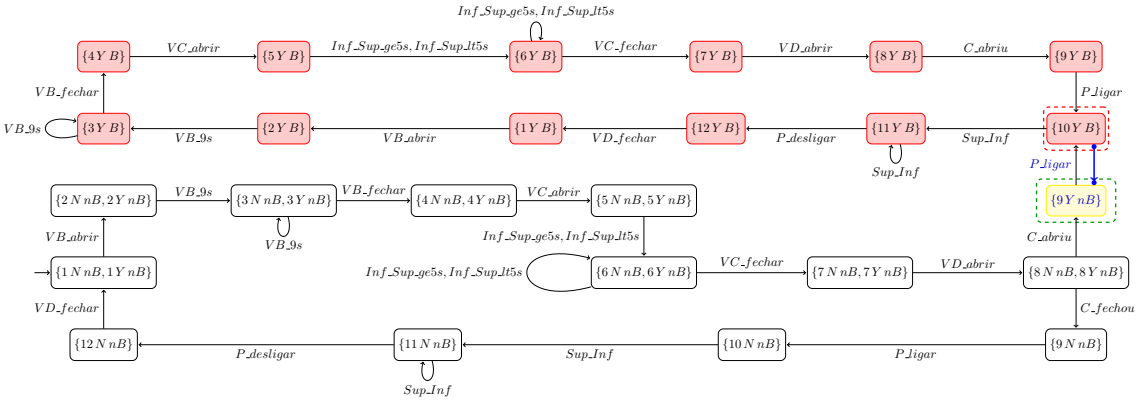
Fonte: Elaborada pelo autor (2022).

A análise efetuada pelo Algoritmo 5 indica que o sistema é controlável seguro por meio da diagnose das falhas. Este resultado é obtido ao se observar as sequências de eventos existentes nos diagnosticadores que levam o sistema a alcançar os estados que compõem o conjunto de estados de fronteira (CEF). Como existem, para todas as sequências, estados certos de falha nos quais é possível desabilitar eventos controláveis para exercer controlabilidade segura pela diagnose, as falhas podem receber esta classificação. A execução desta análise no Algoritmo 5 inicia pelos estados pertencentes ao CEF e evolui verificando outros estados dos diagnosticadores, seguindo o fluxo mostrado pelos caminhos destacados em azul nas Figuras 32 e 33. Por fim, devido a condição de existir controlabilidade segura no sistema por meio da diagnose das falhas, o sistema

Figura 33 – Diagnosticadores de falhas relacionados ao supervisor S_2 com: estados certos de falha destacados em amarelo e vermelho; maus estados destacados em vermelho; estados passíveis de controle destacados por tracejado verde; estados que não devem ser alcançados destacados por tracejado vermelho; e sentido do fluxo de análise da diagnose segura e da controlabilidade segura no diagnosticador destacado por seta em azul.



(a) Diagnosticador da falha de vazamento na válvula B ($D_{f_{VB}}$).



(b) Diagnosticador da falha de travamento fechado da válvula D ($D_{f_{VD_{S_2}}}$).

Fonte: Elaborada pelo autor (2022).

executa a função de *Critério de Mudança de Controle*, presente no Algoritmo 1.

Para o critério de mudança de controle adotado no exemplo, tem-se como função o Algoritmo 7. No algoritmo, são obtidas informações relacionadas às falhas e que vão colaborar com o projeto do CTF. Portanto, para as falhas f_{VA} e f_{VD}, em S_1, e para as falhas f_{VB} e f_{VD}, em S_2, as informações obtidas são, respectivamente, $\{\{1,Y,nB\},VA\_abrir\}$, $\{\{9,Y,nB\},P\_ligar\}$, $\{\{1,Y,nB\},VB\_abrir\}$ e $\{\{9,Y,nB\},P\_ligar\}$.

Algoritmo 7: Critério de Mudança de Controle (Continuar a Operação do Sistema até onde for Possível).

Entrada: (D, $CPME$, $CEPC$)
Saída: *Dados para a Controlabilidade Segura em S*

```
1  início
2      obter o diagnosticador D                    /* D = (X_D, Σ_o, δ_D, Γ_D, x_D0) */
3      Dados para a Controlabilidade Segura em S ← ∅
4      Conjunto de Estados a Avaliar ← ∅
5      para todo x_F ∈ CPME´ faça
6          Conjunto de Estados a Avaliar ← ∅
7          para todo δ_D(x_D, e) = x_F faça
8              se x_D ≠ x_F então
9                  se x_D ∈ CEPC então
10                     Dados para a Controlabilidade Segura em S ← Dados para a
                         Controlabilidade Segura em S ∪ {x_D, e}
11                 senão
12                     Conjunto de Estados a Avaliar ← Conjunto de Estados a Avaliar
                         ∪{x_D}
13         enquanto Conjunto de Estados a Avaliar ≠ ∅ faça
14             obter o primeiro elemento do Conjunto de Estados a Avaliar (x_a)
15             Conjunto de Estados a Avaliar ← Conjunto de Estados a Avaliar −{x_a}
16             para todo δ_D(x_D, e) = x_a faça
17                 se x_D ≠ x_a então
18                     se x_D ∈ Conjunto de Estados Passíveis de Controle então
19                         Dados para a Controlabilidade Segura em S ← Dados para a
                             Controlabilidade Segura em S ∪ {x_D, e}
20                     senão
21                         Conjunto de Estados a Avaliar ← Conjunto de Estados a Avaliar
                             ∪{x_D}
```

4.2.3 Sistema de Síntese de Controle Tolerante a Falhas (FTCSS)

O FTCSS opera interligado ao FAS e tem o objetivo de projetar um sistema de CTF que opere conforme o modelo descrito na Seção 4.2.1. Este sistema projetado deve ter uma estrutura que seja flexível, permitindo tanto o uso de uma estratégia de controle (supervisor e diagnosticadores) para a condição normal de operação, como o uso de outras estratégias de controle que permitam a operação degradada do sistema. Os elementos que compõem o sistema de CTF são os diagnosticadores, os supervisores e o módulo de chaveamento.

Os diagnosticadores e os supervisores utilizados no FTCSS são, respectivamente, os diagnosticadores que foram obtidos nas análises das falhas realizadas no FAS e os supervisores apresentados na entrada da RAFAH (S_{lista}) e que estão presentes em uma lista, denominada S_{CTF}, criada pelo FAS. A disposição dos supervisores na lista S_{CTF} segue a mesma disposição

na qual eles são apresentados na lista S_{lista}. Se considerarmos as listas como conjuntos formados por supervisores, podemos dizer que $S_{CTF} \subseteq S_{lista}$. A criação da lista S_{CTF} no FAS ocorre por conta da possibilidade do diagnóstico de uma falha não apresentar condições de controlabilidade segura para o sistema, o que faz com que o supervisor relacionado à ela se torne inelegível para uso no CTF e, portanto, não seja utilizado na análise do FTCSS.

O módulo de chaveamento é o elemento da arquitetura responsável por habilitar ou desabilitar estratégias de controle no sistema de CTF. Sua ação leva em consideração informações recebidas dos diagnosticadores que devem estar ativos no sistema. Na síntese de CTF efetuada pelo FTCSS, a escolha de qual estratégia de controle adotar no SPA, supervisor e diagnosticadores, segue a ordem em que os supervisores estão dispostos na lista S_{CTF}.

Fundamentalmente o FTCSS realiza uma análise das informações provenientes do FAS com o intuito de desenvolver o módulo de chaveamento do CTF. O resultado da análise produz uma tabela com informações que servem de suporte para a síntese desse módulo, que será posteriormente sintetizado em bloco de função da IEC 61499. Podemos considerar o módulo de chaveamento como sendo uma máquina de estados composta por três tipos de estados:

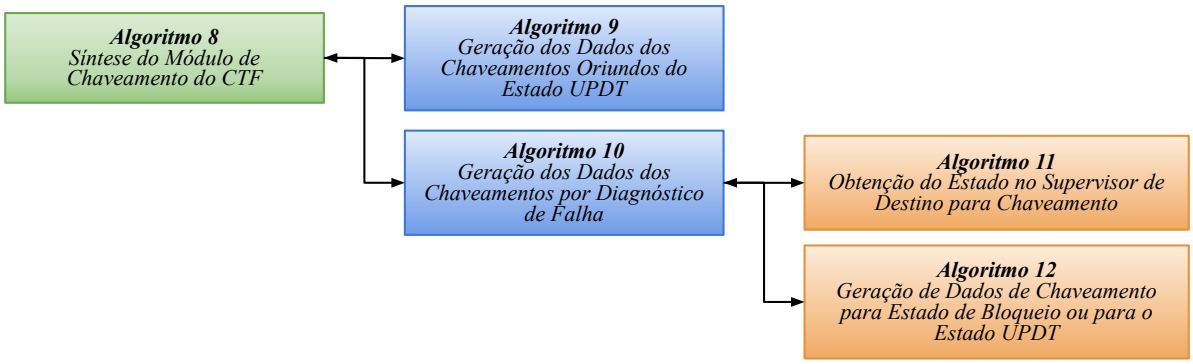
1. Estado de atualização da condição do SPA e de definição da estratégia de controle a adotar;
2. Estados de operação das estratégias de controle;
3. Estados de bloqueio das estratégias de controle.

A operação efetuada pelo FTCSS encontra-se modularizada nos Algoritmos 8, 9, 10, 11 e 12. A interação entre estes algoritmos é mostrada na Figura 34. O Algoritmo 8, denominado de Síntese do Módulo de Chaveamento do CTF, é o algoritmo que fica responsável por distribuir informações referentes as diagnoses das falhas para os demais algoritmos e por coletar as informações de chaveamento no sistema (*Dados para Chaveamento*) geradas por eles.

O estado de atualização da condição do SPA e de definição da estratégia de controle a adotar é denominado no módulo de chaveamento de $UPDT$. O estado $UPDT$ define uma estratégia de controle para uso, levando em consideração a condição atual dos componentes do SPA, que possuem falhas passíveis de diagnóstico nessa estratégia de controle. Ou seja, estes componentes precisam não estar em estado de falha para que ocorra chaveamento para a estratégia de controle. No modelo, é definido que o estado $UPDT$ sempre executa o chaveamento para o estado inicial do supervisor pertencente a estratégia de controle que está sendo habilitada. Os dados dos chaveamentos oriundos do estado $UPDT$ são definidos pelo Algoritmo 9.

A geração dos dados dos chaveamentos por diagnóstico de falha encontram-se modularizados nos Algoritmos 10, 11 e 12. No CTF, um chaveamento por diagnóstico de falha tem como origem um estado que representa a operação da estratégia de controle a qual a falha está associada. Este chaveamento pode ter como destino uma outra estratégia de controle, um estado de bloqueio ou o estado $UPDT$.

Figura 34 – Diagrama de modularização das operações executadas pelo sistema de síntese de controle de falhas (FTCSS). Da esquerda para a direita, as setas indicam chamadas de funções (algoritmos) e, da direita para a esquerda, indicam retorno das funções.



Fonte: Elaborada pelo autor (2022).

Algoritmo 8: Síntese do Módulo de Chaveamento do CTF

Entrada: (*Informações das Diagnoses das Falhas*, S_{CTF}, *Supervisores*, *Diagnosticadores*, *Modelos dos Componentes com as Falhas*)

Saída: (*Dados para Chaveamento*, *Comandos de Chaveamento em D*, *Comandos de Chaveamento em S*)

1 **início**

    /* Função de geração dos dados dos chaveamentos oriundos do estado *UPDT* (Algoritmo 9) */
    /* estado de *UPDT* → estado de atualização das condições dos componentes do sistema e de definição de estratégia de supervisão */
2   *Dados para Chaveamento* ← *Geração dos Dados dos Chaveamentos Oriundos do Estado UPDT*(*Informações das Diagnoses das Falhas*, S_{CTF})
3   **para todo** *Informações da Diagnose de* $f \in$ *Informações das Diagnoses das Falhas* **faça**
        /* Função de geração dos dados dos chaveamentos por diagnóstico de falha (Algoritmo 10) */
4   *Novos Dados de Chaveamento* ← *Geração dos Dados dos Chaveamentos por Diagnóstico de Falha* (*Informações da Diagnose de* f, S_{CTF})
5   *Dados para Chaveamento* ← *Dados para Chaveamento* ∪ *Novos Dados de Chaveamento*
6   criar os conjuntos *Comandos de Chaveamento em D* e *Comandos de Chaveamento em S* a partir de *Dados para Chaveamento*

Os dados de chaveamento para outra estratégia de controle são calculados pelo Algoritmo 10. Para uma certa falha f, na qual a diagnose permite controlabilidade ao SPA, a análise efetuada pelo FTCSS para obter um supervisor S', que substitua S, é efetuada no Algoritmo 11 e se baseia em três condições: (I) S' deve ser um supervisor posterior a S em S_{CTF} (linha 14 do Algoritmo 10); (II) em seu conjunto de eventos $\Sigma_{S'}$, não deve existir eventos do componente do sistema que teve a falha diagnosticada e eventos controláveis que estejam sendo desabilitados por conta da

ação de controlabilidade segura (linha 16 do Algoritmo 10); e (III) Considerando os eventos que S possui em comum com S' e que não possuem transição definida nos estados de S em que ocorrerão o chaveamento, S' deve possuir estados com condições idênticas (Algoritmo 11). A condição (I) se deve ao fato que se um supervisor de maior prioridade não está em uso no sistema, é porque alguma falha já ocorreu em um de seus componentes. Na condição (II), se um componente apresentou falha no sistema, o mesmo deve ficar em uma situação não operacional em uma nova estratégia de controle, portanto, seus eventos não devem aparecer no novo supervisor a ser adotado. O mesmo acontece com os eventos controláveis que serão desabilitados, pois é a desabilitação desses eventos que garantirá a controlabilidade segura do sistema. A condição (III) é baseada no espaço de estados do sistema, para que se possa garantir sincronismo e continuação das atividades em operação no sistema, é necessário que o novo supervisor possua um estado que reflita o estado atual da planta no momento em que ocorrerá o chaveamento.

Algoritmo 9: Geração dos Dados dos Chaveamentos Oriundos do Estado *UPDT*

Entrada: (*Informações das Diagnoses das Falhas*, S_{CTF})
Saída: (*Dados para Chaveamento*)

```
1  início
2      para todo S ∈ S_CTF faça
3          Componentes com Falhas em S ← ∅
4          Dados para Chaveamento ← ∅
5          para todo (f, Comp_f, S', ...) ∈ Informações das Diagnoses das Falhas faça
6              se S = S' então
7                  Componentes com Falhas em S ← Componentes com Falhas em S
                     ∪ {Comp_f}
8          inserir informações de chaveamento do estado UPDT para o supervisor S e dos
             componentes com falhas em Dados para Chaveamento
```

Os dados de chaveamento por diagnóstico de falha que têm como destino o estado $UPDT$ ou um estado de bloqueio são definidos no Algoritmo 12. O chaveamento do estado que representa a estratégia de controle em uso para o estado $UPDT$ acontece quando a diagnose da falha ocorre no estado inicial do supervisor em uso e não existe uma outra estratégia de controle que permita as condições impostas pelo espaço de estados atual do SPA. Já a condução do SPA para um estado de bloqueio ocorre quando uma estratégia de controle em uso precisa ser chaveada por conta da diagnose de uma falha, porém, o supervisor em uso não se encontra no estado inicial e não existe uma outra estratégia de controle que permita as condições impostas pelo espaço de estados atual do SPA. O retorno do estado de bloqueio para o estado que representa a estratégia de controle ocorre quando o componente do sistema, que apresentou a falha, é reparado.

Ao final, o FTCSS deve separar e reorganizar as informações presentes no conjunto denominado de *Dados para Chaveamento*, criando os conjuntos de informações *Comandos de Chaveamento em D* e *Comandos de Chaveamento em S* (linha 6 do Algoritmo 8). Estes

Algoritmo 10: Geração dos Dados dos Chaveamentos por Diagnóstico de Falha

Entrada: (*Informações da Diagnose de* f, S_{CTF})
Saída: (*Novos Dados de Chaveamento*)

```
1  início
       /* Informações das Diagnoses das Falhas = (f, Comp_f, S, ..., Dados
          de Controlabilidade Segura, ...)                                  */
2      obter S                                  /* S = (X_S, Σ_S, δ_S, Γ_S, x_S0, X_Sm)  */
3      obter o conjunto de eventos de S (Σ_S)
4      obter D
5      obter o autômato G_f do componente que apresenta a falha f (Comp_f)
6      obter o conjunto de eventos de G_f (Σ_Gf)
       /* Σ_des → conjunto de eventos a serem desabilitados                */
       /* CENP → conjunto de eventos não possíveis no estado atual do
          sistema                                                           */
7      para todo {x_D, Σ_des} ∈ Dados de Controlabilidade Segura faça
8          obter o conjunto de eventos de x_D (Σ_xD)
9          Novos Dados de Chaveamento ← ∅
10         CENP ← Σ_S − Σ_xD − Σ_Gf − Σ_des
11         Supervisor Escolhido ← ∅
12         para todo S' ∈ S_CTF faça
13             se S' é posterior a S em S_CTF então
14                 obter o conjunto de eventos de S' (Σ_S')
15                 se Σ_S' ∩ (Σ_Gf ∪ {Σ_des}) = ∅ então
16                     Supervisor Escolhido ← S'
                       /* Função de obtenção do estado no supervisor de
                          destino para chaveamento (Algoritmo 11)           */
17                     Estado Encontrado ← Obtenção do Estado no Supervisor de Destino
                        para Chaveamento (S', CENP)
18                     se Estado Encontrado ≠ NULO então
19                         inserir informações de chaveamento do supervisor S para o
                            supervisor S', do estado encontrado e da diagnose em Novos
                            Dados de Chaveamento
20                         sair da estrutura de repetição
21                     senão
22                         Supervisor Escolhido ← ∅
23         se Supervisor Escolhido = ∅ então
               /* Função de geração de dados de chaveamento para estado de
                  bloqueio ou para o estado UPDT (Algoritmo 12)             */
24             Novos Dados de Chaveamento ← Geração de Dados de Chaveamento para
                Estado de Bloqueio ou para o Estado UPDT (f, x_D, S, Novos Dados de
                Chaveamento)
```

Algoritmo 11: Obtenção do Estado no Supervisor de Destino para Chaveamento

Entrada: $(S', CENP)$
Saída: (*Estado Encontrado*)

início
  /* $CENP \rightarrow$ conjunto de eventos não possíveis no estado atual do sistema */
  Estado Encontrado = *NULO*
  Estados a Avaliar $\leftarrow \emptyset$
  Estados Avaliados $\leftarrow \emptyset$
  obter S' /* $S' = (X_{S'}, \Sigma_{S'}, \delta_{S'}, \Gamma_{S'}, x_{S'_0}, X_{S'_m})$ */
  obter o conjunto de eventos de $x_{S',0}$ (Σ_x) /* $\Sigma_x = \{e : \delta_{S'}(x_{S'_0}, e)!\}$ */
  obter o número de estados de S' ($NE_{S'}$)
  se $\Sigma_x \cap CENP \neq \emptyset$ **então**
    Estados Avaliados $\leftarrow$ *Estados Avaliados* $\cup \{x_{S',0}\}$
    para todo $\delta_{S'}(x_{S'_0}, e) = x_{S'}$ **faça**
      Estados a Avaliar $\leftarrow$ *Estados a Avaliar* $\cup \{x_{S'}\}$
  senão
    Estado Encontrado $\leftarrow x_{S'_0}$
  /* $NE_{aval} \rightarrow$ número de estados em *Estados Avaliados* */
  enquanto *(Estado Encontrado = NULO)* $\wedge$ $(NE_{aval} \neq NE_{S'})$ **faça**
    obter primeiro elemento ($x_{S'_1}$) de *Estados a Avaliar*
    se $x_{S'_1} \in$ *Estados Avaliados* **então**
      Estados a Avaliar $\leftarrow$ *Estados a Avaliar* $-\{x_{S'_1}\}$
    senão
      obter o conjunto de eventos de $x_{S'_1}$ (Σ_x)
      se $\Sigma_x \cap CENP \neq \emptyset$ **então**
        Estados Avaliados $\leftarrow$ *Estados Avaliados* $\cup \{x_{S'_1}\}$
        para todo $\delta_{S'}(x_{S'_1}, e) = x_{S'_2}$ **faça**
          Estados a Avaliar $\leftarrow$ *Estados a Avaliar* $\cup \{x_{S'_2}\}$
      senão
        Estado Encontrado $\leftarrow x_{S'_1}$
    obter o número de estados em *Estados Avaliados* (NE_{aval})

novos conjuntos possuem informações que servirão para a síntese dos blocos de função dos diagnosticadores e dos supervisores, respectivamente.

Algoritmo 12: Geração de Dados de Chaveamento para Estado de Bloqueio ou para o Estado UPDT

Entrada: (f, x_D, S, *Novos Dados de Chaveamento*)
Saída: (*Novos Dados de Chaveamento*)

1 **início**
  /* $x_D = \{\{x_{S_1}, Y, nB\}, \ldots, \{x_{S_n}, Y, nB\}\}, n \in \mathbb{N}^*$ */
2 **se** $x_D = \{\{x_{S_1}, Y, nB\}\} \wedge x_{S_1} = x_{S_0}$ **então**
3 inserir informações de chaveamento do supervisor S para o estado *UPDT* e da diagnose em *Novos Dados de Chaveamento*
4 **senão**
  /* `estado de bloqueio` ← `bloquear o sistema no estado atual` */
5 inserir informações de chaveamento do supervisor S para um estado de bloqueio (*BLQ*) e da diagnose em *Novos Dados de Chaveamento*
6 inserir informações de chaveamento do estado de bloqueio *BLQ* para o supervisor S, do estado do supervisor antes do bloqueio e do componente que gerou a falha f em *Novos Dados de Chaveamento*

O Exemplo 2 a seguir mostra como o FTCSS constrói a tabela de dados para o módulo de chaveamento do CTF proposto para o processo de mistura da Figura 22.

Exemplo 2. Considerando os resultados gerados pelo FAS no Exemplo 1, o FTCSS obtem os dados de chaveamentos apresentados na Tabela 2, que servirão para a construção do módulo de chaveamento do CTF. O processo inicia pela definição dos chaveamentos que têm como origem o estado $UPDT$ (linha 2 do Algoritmo 8). Para isso, cria-se um conjunto para cada supervisor $S \in S_{CTF}$, com as informações dos seus componentes que possuem falha monitorada (Algoritmo 9). As informações dos componentes são obtidas por meio da leitura das informações de diagnose de cada falha f no conjunto de informações das diagnoses das falhas. Portanto, para os supervisores S_1 (Figura 25a) e S_2 (Figura 25b), verificam-se, respectivamente, os conjuntos de componentes $\{VA, VD\}$ e $\{VB, VD\}$ (linhas 1 e 2 da Tabela 2). A seguir, são geradas as informações de chaveamento por diagnóstico de falha (Algoritmos 10, 11 e 12). Para a definição do destino a ser adotado no chaveamento (uma outra estratégia de controle, o estado $UPDT$ ou um estado de bloqueio), o FTCSS avalia a condição de chaveamento para cada falha f obtendo:

- Para f_{VA} o chaveamento para S_2, pois o conjunto de eventos do estado inicial de S_2 ($\Sigma_x = \{VB\_abrir\}$) atende ao conjunto de eventos não possíveis no estado atual do sistema ($\Sigma_x \cap (\Sigma_{S_1} - \{VA\_abrir, VA\_fechar, VA\_9s\} - \{VA\_abrir\}) = \emptyset$);
- Para f_{VD}, em S_1, o chaveamento para o estado de bloqueio BLQ_1 e o chaveamento de retorno de BLQ_1 para S_1, pois não existe supervisor substituto S que atenda ao

conjunto de eventos não possíveis no estado atual do sistema ($CENP = \Sigma_{S_1} - \{VD\_abrir, VD\_fechar, C\_abriu, C\_fechou\} - \{P\_ligar\}$);

- Para f_{VB} o chaveamento para o estado de $UPDT$, pois não existe supervisor após S_2 em S_{CTF};
- Para f_{VD}, em S_2, o chaveamento para o estado de bloqueio BLQ_2 e o chaveamento de retorno de BLQ_2 para S_2, pois não existe supervisor após S_2 em S_{CTF} e a controlabilidade segura não é exercida no estado inicial de S_2;

Tabela 2 – Informações para a síntese do CTF do processo de mistura da Figura 22.

	Origem	Destino	Estado de S_1	Estado de S_2	Sinal	Conjunto de Componentes
1	$UPDT$	SUP_1	1	0	-	VA, VD
2	$UPDT$	SUP_2	0	1	-	VB, VD
3	SUP_1	SUP_2	0	1	$DIAG\_VA$	-
4	SUP_1	BLQ_1	0	0	$DIAG\_VD1$	-
5	BLQ_1	SUP_1	9	0	-	VD
6	SUP_2	$UPDT$	0	0	$DIAG\_VB$	-
7	SUP_2	BLQ_2	0	0	$DIAG\_VD2$	-
8	BLQ_2	SUP_2	0	9	-	VD

Fonte: Elaborado pelo autor (2021).

4.2.4 Síntese do Controle Tolerante a Falhas em Blocos de Função

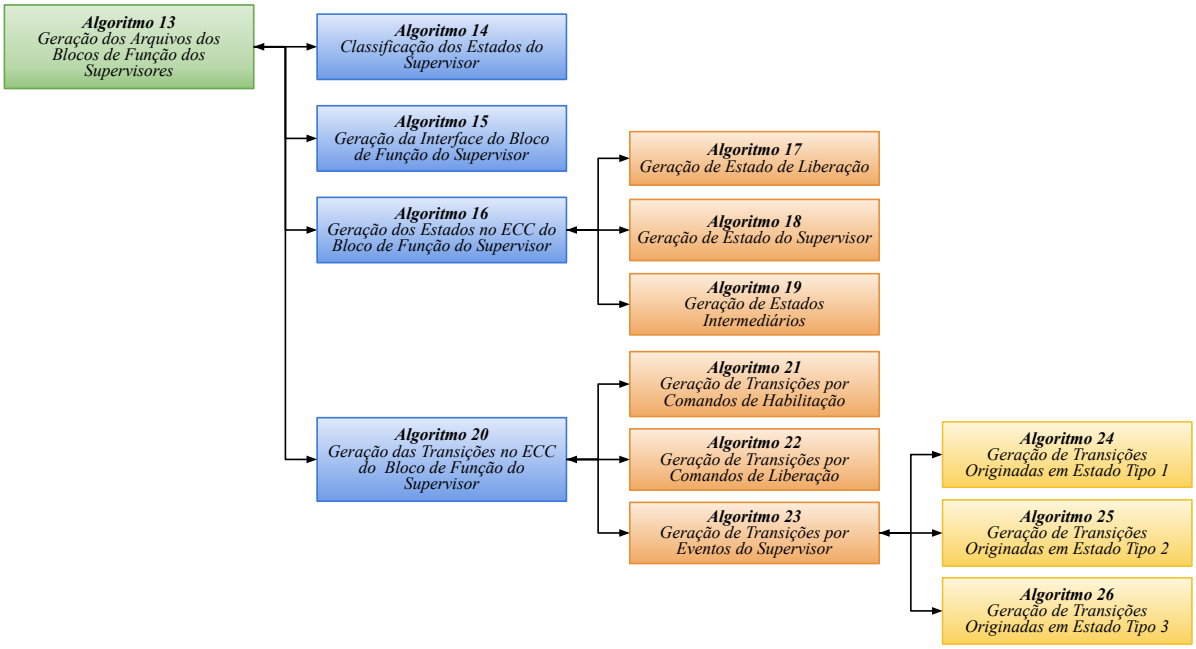
A conversão da estrutura do CTF (supervisores, diagnosticadores e módulo de chaveamento), obtida na Seção 4.2.3, em blocos de função básicos da IEC 61499 é executada na etapa de síntese de lógica de controle da RAFAH. Por possuirem propósitos diferentes dentro da estrutura e, por conta disso, executarem ações diferentes no CTF, cada um dos elementos da estrutura possui um conjunto específico de algoritmos para efetuar a sua síntese em blocos de função. Como a IEC 61499 define representações em arquivos com linguagem de marcação do tipo *eXtensible Markup Language* (XML), estes elementos são representados em arquivos deste tipo.

As Seções 4.2.4.1, 4.2.4.2 e 4.2.4.3 a seguir mostram, respectivamente, como são realizadas as sínteses dos supervisores, dos diagnosticadores e do módulo de chaveamento, pertencentes a estrutura do CTF, em blocos de função básicos da IEC 61499.

4.2.4.1 Síntese de Supervisores

A síntese de um supervisor em bloco de função se baseia em sua modelagem no ECC. Conforme visto na Seção 2.2, o ECC consiste em uma máquina de estados. Como a síntese possui uma ação integrada aos demais elementos do CTF, sua representação no ECC deve ser composta de informações adicionais que, em determinados momentos, interferem em sua ação sobre o SPA. Para sintetizar os blocos de função dos supervisores, a etapa de síntese de lógica de controle da RAFAH utiliza um conjunto de informações, denominado de *Informações para os BFBs dos Supervisores*, proveniente do FTCSS. Os blocos de função dos supervisores do CTF são gerados pelos Algoritmos 13 – 26. O fluxo de operações destes algoritmos encontra-se detalhado no diagrama mostrado na Figura 35.

Figura 35 – Diagrama de modularização das operações executadas na geração dos blocos de função dos supervisores do CTF. Da esquerda para a direita, as setas indicam chamadas de funções (algoritmos) e, da direita para a esquerda, indicam retorno das funções.



Fonte: Elaborada pelo autor (2022).

A síntese dos supervisores inicia no Algoritmo 13 por meio da avaliação das informações contidas no conjunto *Informações para os BFBs dos Supervisores*. Neste conjunto, existe um grupo de informações para a geração do bloco de função de cada supervisor. O bloco de função do supervisor é desenvolvido pela sequência de ações a seguir:

1. Descrições das informações básicas de definição do bloco de função;
2. Desenvolvimento da interface do bloco de função;
3. Desenvolvimento do ECC do bloco de função.

Algoritmo 13: Geração dos Arquivos dos Blocos de Função dos Supervisores

Entrada: (*Informações para os BFBs dos Supervisores*)

1 **início**

2 **para todo** *(S, Dados para a Controlabilidade Segura em S, Comandos de Chaveamento em S)* $\in$ *Informações para os BFBs dos Supervisores* **faça**

`/* Arquivo xml */`

3 escrever cabeçalho

4 escrever a definição de tipo de documento (DTD) que valida o bloco de função

5 escrever elemento inicial *FBType* (definição do bloco) acompanhado dos atributos *Comment* (comentário) e *Name* (nome do bloco)

6 escrever elemento vazio *Identification* acompanhado do atributo *Standard* para identificar a versão da norma à qual o bloco é compatível

7 escrever elemento vazio *VersionInfo* acompanhado dos atributos *Author* (autor), *Date* (data) e *Version* (versão) para registrar informações relacionadas à versão do arquivo

`/* Função para classificação dos estados do supervisor (Algoritmo 14) */`

8 *Estados Tipo 1*, *Estados Tipo 2*, *Estados Tipo 3*, *Conjunto de Estados com Necessidade de Liberação* $\leftarrow$ *Classificação dos Estados do Supervisor (S, Dados para a Controlabilidade Segura em S)*

9 escrever elemento inicial *InterfaceList* para identificar os elementos que compõem a interface do bloco

`/* Função para geração da interface do bloco de função do supervisor (Algoritmo 15) */`

10 *Geração da Interface do Bloco de Função do Supervisor (S, Estados Tipo 2, Estados Tipo 3)*

11 escrever elemento final *InterfaceList*

12 escrever elemento inicial *BasicFB* para identificar a estrutura interna do bloco

13 escrever elemento inicial *ECC* para identificar o gráfico de controle de execução

`/* Funções para geração da estrutura interna do bloco de função do supervisor (Algoritmos 16 e 20) */`

14 *Geração dos Estados no ECC do Bloco de Função do Supervisor (S, Estados Tipo 2, Estados Tipo 3, Dados para a Controlabilidade Segura em S)*

15 *Geração das Transições no ECC do Bloco de Função do Supervisor (S, Conjunto de Estados com Necessidade de Liberação, Comandos de Chaveamento em S)*

16 escrever elemento final *ECC*

17 escrever elemento final *BasicFB*

18 escrever elemento final *FBType*

A escrita das informações básicas de definição do bloco de função do supervisor são realizadas no Algoritmo 13 pelo intervalo de comandos da linha 3 a linha 7. Nestas descrições estão presentes informações como a definição do tipo do bloco, a versão da norma IEC 61499 à qual ele é compatível, o autor e a versão do bloco.

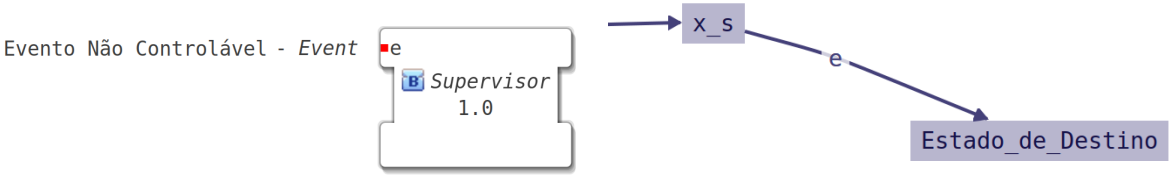
O desenvolvimento da interface do bloco de função do supervisor é realizada pelo Algoritmo 15. A interface é composta por três tipos de eventos do CTF: os eventos do supervisor (controláveis e não controláveis), o evento de chaveamento do supervisor e um evento de liberação da ação do supervisor. Os eventos não controláveis e os eventos controláveis do supervisor são representados na interface como eventos de entrada e eventos de saída, respectivamente. O evento de chaveamento do supervisor (evento de entrada *COM_HAB*) é um evento proveniente do módulo de chaveamento do CTF. Ele opera em conjunto com uma variável de entrada do tipo inteira, denominada de *ESTADO*, e serve para habilitar ($ESTADO \geq 1$) ou desabilitar ($ESTADO = 0$) o supervisor no CTF. Já o evento de liberação da ação do supervisor (evento de entrada *LIBERAR*) é um evento que pode ou não existir na estrutura do bloco de função. Ele é utilizado quando se necessita que a ação de um determinado estado do supervisor seja liberada por um ou mais diagnosticadores. Esta ação de liberação é específica do modelo proposto nesta tese para a implementação do supervisor, sendo necessária para que o processamento do bloco de função do supervisor seja substituído por outra tarefa no sistema do controlador lógico programável. A parada da execução do bloco de função no sistema permite que uma ação de chaveamento, por parte do módulo de chaveamento, ou uma ação de liberação, por parte dos diagnosticadores, possa ocorrer no estado do bloco de função.

Para que a interface do bloco de função do supervisor seja gerada, é necessária que seja feita antes uma classificação dos estados do supervisor (Algoritmo 14), definindo no modelo os estados com necessidade de liberação (*Conjunto de Estados com Necessidade de Liberação*). A ação de classificação executada por este algoritmo é mais ampla e serve também para classificar os estados em função da controlabilidade dos eventos presentes em seu conjunto de eventos ativos. Em um autômato supervisor S, a função de eventos ativos $\Gamma(x_s)$ de um estado de S ($\Gamma(x_s) = \{e : e \in \Sigma_S\}$) pode ter três configurações possíveis em relação a controlabilidade dos seus evento, são elas: $\Gamma(x_s)$ pode ser formada exclusivamente por eventos não controláveis, $\Gamma(x_s)$ pode ser formada exclusivamente por eventos controláveis ou $\Gamma(x_s)$ pode ser formada tanto por eventos não controláveis quanto por eventos controláveis. Apesar da TCS permitir que $\Gamma(x_s)$ possa ter mais de um evento controlável, isto acarreta no que é conhecido na literatura como problema da escolha (FABIAN; HELLGREN, 1998). Portanto, para evitar o problema da escolha, os supervisores utilizados no CTF consideram a possibilidade de uma única transição por evento controlável em um estado. Dessa forma, as sínteses dos estados dos supervisores no bloco de função são fundamentadas pelas Regras 1, 2 e 3 a seguir.

Regra 1 (*Estados Tipo 1*). Para os casos em que $\Gamma(x_s)$ é formado exclusivamente por eventos não controláveis ($e \in \Sigma_{nc}$), por serem estes eventos provenientes da planta (sistema controlado),

suas ocorrências precisam ser observadas pelo bloco de função do supervisor como eventos de entrada. Além disso, a ação de transição proporcionada por estes eventos deve ser condicionada no ECC a habilitação do estado e a posterior ocorrência do evento (Figura 36).

Figura 36 – Exemplo de implementação de estado com evento não controlável no bloco de função do supervisor.



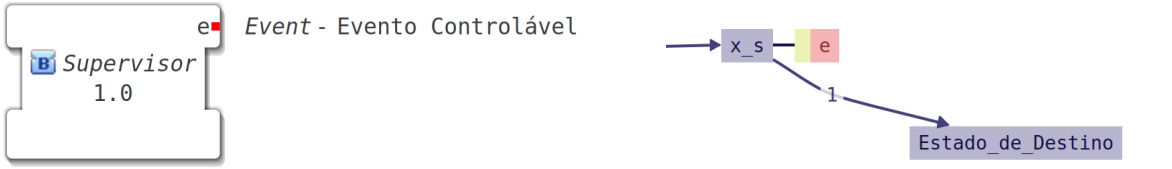
(a) Representação parcial da interface do supervisor.

(b) Representação parcial do ECC do supervisor.

Fonte: Elaborada pelo autor (2022).

Regra 2 (*Estados Tipo 2*). Para os casos em que $\Gamma(x_s)$ é formado exclusivamente por um evento controlável ($e \in \Sigma_c$), por ser este evento de natureza de ativação regida pelo supervisor, ele deve aparecer como evento de saída no bloco de função, representando a ação a ser executada na planta (sistema controlado). Esta geração do evento controlável deve estar condicionada no ECC a habilitação do estado, que representa o estado x_S do autômato, com posterior transição para outro estado por meio de uma transição de guarda 1 (Figura 37), transição incondicional que ocorre de forma automática.

Figura 37 – Exemplo de implementação de estado com evento controlável no bloco de função do supervisor.



(a) Representação parcial da interface do supervisor.

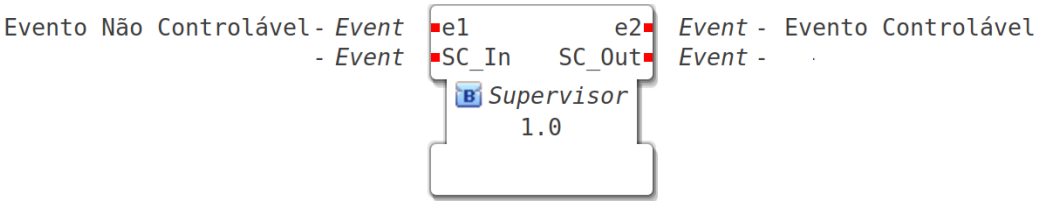
(b) Representação parcial do ECC do supervisor.

Fonte: Elaborada pelo autor (2022).

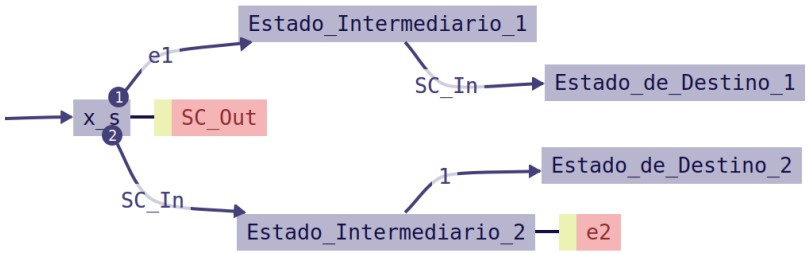
Regra 3 (*Estados Tipo 3*). Nos estados em que $\Gamma(x_s)$ é formada por um evento controlável e por um ou mais eventos não controláveis, os eventos não controláveis, por serem eventos provenientes da planta (sistema controlado), precisam que suas ocorrências sejam observadas pelo bloco de função como eventos de entrada. Além disso, a ação de transição proporcionada por estes eventos deve ser condicionada no ECC a habilitação do estado e a posterior ocorrência do evento. Já o evento controlável, por ser de natureza de ativação regida pelo supervisor, deve aparecer como evento de saída no bloco de função, representando a ação a ser executada na planta.

Para que o bloco possa observar a ocorrência de um evento não controlável, quando este existir, diminuindo o risco da ocorrência do problema da sincronização inexata (FABIAN; HELLGREN, 1998) (VIEIRA et al., 2016), sua ação é priorizada em relação a ação do evento controlável. Para isso, é necessário que o bloco habilite o evento controlável apenas após a realização de uma autoverificação (*self-check*) por meio de um evento de saída SC_{Out} conectado a um evento de entrada SC_{In}. A ação de transição proporcionada por um evento não controlável deve estar condicionada no ECC a habilitação do estado do supervisor, que acarreta na publicação do evento SC_{Out}, a posterior ocorrência do evento não controlável, com uma transição para um estado intermediário, e a transição para o estado de destino por intermédio do evento de entrada SC_{In}. Por outro lado, a geração de um evento controlável deve estar condicionada no ECC a habilitação do estado, que representa o estado do supervisor, a transição para um estado intermediário por intermédio do evento de entrada SC_{In}, com posterior transição para o estado de destino por meio de uma transição de guarda 1 (Figura 38).

Figura 38 – Exemplo de implementação de estado com eventos controlável e não controlável em bloco de função de *S*.



(a) Representação parcial da interface do supervisor.



(b) Representação parcial do ECC do supervisor.

Fonte: Elaborada pelo autor (2022).

Após a criação da interface do bloco de função do supervisor, inicia-se o desenvolvimento de sua estrutura interna composta pelo ECC. Este desenvolvimento é executado em duas etapas: primeiro são gerados os estados e depois as transições que compõem o ECC.

A geração dos estados no ECC do bloco de função do supervisor é executada pelos Algoritmos 16, 17, 18 e 19. Existem quatro tipos de estados criados no ECC: o estado de desabilitação do supervisor, os estados de liberação, os estados do supervisor e os estados intermediários. O estado de desabilitação do supervisor é um estado do ECC que serve para fazer com que o supervisor fique sem executar ações no CTF. Sua geração ocorre no Algoritmo 16. Os estados de liberação são estados que podem ou não existir no ECC, sua geração depende do

Algoritmo 14: Classificação dos Estados do Supervisor

Entrada: (*S*, *Dados de Controlabidade Segura*)
Saída: (*Estados Tipo 1*, *Estados Tipo 2*, *Estados Tipo 3*, *Conjunto de Estados com Necessidade de Liberação*)

```
1  início
       /* Estados Tipo 1 → conjunto de estados com transições de
          saídas por eventos não controláveis                          */
       /* Estados Tipo 2 → conjunto de estados com transição de saída
          por evento controlável                                       */
       /* Estados Tipo 3 → conjunto de estados com transições de
          saídas por evento controlável e por eventos não controláveis
          */
       /* S = (X_S, Σ_S, δ_S, Γ_S, x_S0, X_Sm)                         */
2      obter o autômato S
3      Estados Tipo 1 ← ∅
4      Estados Tipo 2 ← ∅
5      Estados Tipo 3 ← ∅
6      para todo x_S ∈ S faça
7          se Γ_S(x_S) ⊆ Σ_nc então
8              Estados Tipo 1 ← Estados Tipo 1 ∪ {x_S}
9          senão se ∀e ∈ Γ_S(x_S), e ∈ Σ_c ∨ δ(x_S, e) = x_S então
10             Estados Tipo 2 ← Estados Tipo 2 ∪ {x_S}
11         senão
12             Estados Tipo 3 ← Estados Tipo 3 ∪ {x_S}
13     para todo ({x_S, Y, nB}, e) ∈ Dados de Controlabidade Segura faça
14         se x_S ∈ Estados Tipo 2 então
15             Conjunto de Estados com Necessidade de Liberação ←
               Conjunto de Estados com Necessidade de Liberação ∪ {x_S}
```

resultado da análise realizada no Algoritmo 14. Se *Conjunto de Estados com Necessidade de Liberação* $\neq \emptyset$, então o Algoritmo 17 gera um estado de liberação para cada estado pertencente a este conjunto. Os estados denominados de estados do supervisor são os estados obtidos no autômato do supervisor que está sendo sintetizado em bloco de função. Eles são gerados no Algoritmo 18. Os estados denominados de intermediários são estados gerados no Algoritmo 19 que aparecem na implementação do supervisor em bloco de função para complementar a modelagem de estados do supervisor com função de eventos ativos composta por evento controlável e por eventos não controláveis (implementação baseada na Regra 3).

As transições no ECC do bloco de função do supervisor são geradas pelos Algoritmos 20 – 26. Seguindo a estrutura de estados desenvolvida para o ECC, as transições também são classificadas por tipos, dependendo do evento de entrada ao qual está associada. Portanto, as transições podem ser por comandos de habilitação (comandos de chaveamento), por comandos de liberação ou por eventos do supervisor.

Algoritmo 15: Geração da Interface do Bloco de Função do Supervisor

Entrada: (S, *Estados Tipo 3*, *Conjunto de Estados com Necessidade de Liberação*)

início
- `/* Estados Tipo 3 → conjunto de estados com transições de saída por evento controlável e por eventos não controláveis */`
- /* $S = (X_S, \Sigma_S, \delta_S, \Gamma_S, x_{S_0}, X_{S_m})$ */
- obter o autômato S
- escrever elemento inicial *EventInputs* para identificar os eventos de entrada
- escrever elemento inicial *Event* acompanhado dos atributos *Comment* (comentário), *Name="COM_HAB"* (evento *COM_HAB*) e *Type* (tipo do evento) para definir o comando de habilitação do diagnosticador
- escrever elemento vazio *With* acompanhado do atributo *Var="ESTADO"*
- escrever elemento final *Event*
- **se** *Conjunto de Estados com Necessidade de Liberação* $\neq \emptyset$ **então**
  - escrever elemento inicial *Event* acompanhado dos atributos *Comment* (comentário), *Name="LIBERAR"* e *Type* (tipo do evento)
  - escrever elemento final *Event*
- **se** *Estados Tipo 3* $\neq \emptyset$ **então**
  - escrever elemento inicial *Event* acompanhado dos atributos *Comment* (comentário), *Name="SC_In"* (evento para autoverificação do estado) e *Type* (tipo do evento)
  - escrever elemento final *Event*
- **para todo** $e \in \Sigma_S$ **faça**
  - **se** $e \in \Sigma_{nc}$ **então**
    - escrever elemento inicial *Event* acompanhado dos atributos *Comment* (comentário), *Name=String(e)* (evento e) e *Type* (tipo do evento)
    - escrever elemento final *Event*
- escrever elemento final *EventInputs*
- escrever elemento inicial *EventOutputs* para identificar os eventos de saída
- **se** *Estados Tipo 3* $\neq \emptyset$ **então**
  - escrever elemento inicial *Event* acompanhado dos atributos *Comment* (comentário), *Name="SC_Out"* (evento para autoverificação do estado) e *Type* (tipo do evento)
  - escrever elemento final *Event*
- **para todo** $e \in \Sigma_S$ **faça**
  - **se** $e \in \Sigma_c$ **então**
    - escrever elemento inicial *Event* acompanhado dos atributos *Comment* (comentário), *Name=String(e)* (evento e) e *Type* (tipo do evento)
    - escrever elemento final *Event*
- escrever elemento final *EventOutputs*
- escrever elemento inicial *InputVars* para identificar as variáveis de entrada
- escrever elemento vazio *VarDeclaration* acompanhado dos atributos *Comment* (comentário), *Name="ESTADO"* (indicação do estado para chaveamento no bloco) e *Type="INT"* (variável do tipo inteira)
- escrever elemento final *InputVars*

Algoritmo 16: Geração dos Estados no ECC do Bloco de Função do Supervisor

Entrada: (S, *Estados Tipo 2*, *Estados Tipo 3*, *Conjunto de Estados com Necessidade de Liberação*)

1 **início**

/* $S = (X_S, \Sigma_S, \delta_S, \Gamma_S, x_{S_0}, X_{S_m})$ */

/* *Estados Tipo 2* → conjunto de estados com transição de saída por evento controlável */

/* *Estados Tipo 3* → conjunto de estados com transições de saída por evento controlável e por eventos não controláveis */

2 obter o autômato S

/* Criação do estado de desabilitação do supervisor */

3 escrever elemento vazio *ECState* acompanhado dos atributos *Comment* (comentário), *Name="Desabilitado"* (estado de desabilitação do diagnosticador), x e y (coordenadas de posicionamento do estado)

4 **para todo** $x_S \in X_S$ **faça**

/* Função para geração de estado de liberação (Algoritmo 17) */

5 *Geração de Estado de Liberação (x_S, Conjunto de Estados com Necessidade de Liberação)*

/* Função para geração de estado do supervisor (Algoritmo 18) */

6 *Geração de Estado do Supervisor (x_S, Estados Tipo 2, Estados Tipo 3)*

/* Função para geração de estados intermediários (Algoritmo 19) */

/* Estados intermediários → estados que complementam a modelagem no ECC de estados $x_s \in$ *Estados Tipo 3* */

7 *Geração de Estados Intermediários (x_S, Estados Tipo 3)*

Algoritmo 17: Geração de Estado de Liberação

Entrada: (x_S, Conjunto de Estados com Necessidade de Liberação)

1 **início**

2 **se** $x_S \in$ *Conjunto de Estados com Necessidade de Liberação* **então**

3 *rótulo* ← *"Liberar_Estado_"* + *String(x_S)*

4 escrever elemento inicial *ECState* acompanhado dos atributos *Comment* (comentário), *Name=rótulo* (estado do supervisor), x e y (coordenadas de posicionamento do estado)

5 escrever elemento final *ECState*

Algoritmo 18: Geração de Estado do Supervisor

Entrada: (x_S, *Estados Tipo 2*, *Estados Tipo 3*)

início

`/* Estados Tipo 2 → conjunto de estados com transição de saída por evento controlável */`

`/* Estados Tipo 3 → conjunto de estados com transições de saída por evento controlável e por eventos não controláveis */`

rótulo ← *"Estado_"* + *String*(x_S)

escrever elemento inicial *ECState* acompanhado dos atributos *Comment* (comentário), *Name=rótulo* (estado do supervisor), *x* e *y* (coordenadas de posicionamento do estado)

se $x_S \in$ *Estados Tipo 2* **então**

  obter *e* `/*` $e \in \Sigma_c$ `e` $\delta(x_S, e)!$ `*/`

  escrever elemento vazio *ECAction* acompanhado do atributo *Output=String(e)* (evento controlável do sistema)

senão se $x_S \in$ *Estados Tipo 3* **então**

  escrever elemento vazio *ECAction* acompanhado do atributo *Output="SC_Out"* (evento para autoverificação do estado)

escrever elemento final *ECState*

Algoritmo 19: Geração de Estados Intermediários

Entrada: (x_S, *Estados Tipo 3*)

início

`/* Estados Tipo 3 → conjunto de estados com transições de saída por evento controlável e por eventos não controláveis */`

se $x_S \in$ *Estados Tipo 3* **então**

  contagem ← 1

  para todo $\delta(x_S, e) = x'_S$ **faça**

    se $x_S \neq x'_S$ **então**

      rótulo ← *"Estado_Intermediario_"* + *String*(x_S) + *"_"* + *String(contagem)*

      contagem ← *contagem* + 1

      escrever elemento inicial *ECState* acompanhado dos atributos *Comment* (comentário), *Name=rótulo* (estado de transição entre estados do supervisor), *x* e *y* (coordenadas de posicionamento do estado)

      se $e \in \Sigma_c$ **então**

        escrever elemento vazio *ECAction* acompanhado do atributo *String(e)* (evento controlável do sistema)

      escrever elemento final *ECState*

Na etapa de geração das transições, o Algoritmo 20 coordena a chamada dos demais algoritmos, permitindo que sejam geradas, em sequência, as transições por comandos de habilitação (Algoritmo 21), as transições por comandos de liberação (Algoritmo 22) e as transições baseadas em eventos do supervisor (Algoritmo 23). Para as transições baseadas em eventos do supervisor, a geração depende da classificação do estado em relação à sua função de ativação (Regras 1, 2 e 3 apresentadas anteriormente), existindo um algoritmo específico para cada uma das regras (Algoritmos 24, 25 e 26).

Algoritmo 20: Geração das Transições no ECC do Bloco de Função do Supervisor

Entrada: (*S*, *Estados Tipo 1, Estados Tipo 2, Estados Tipo 3, Conjunto de Estados com Necessidade de Liberação*, *Comandos de Chaveamento em S*)

```
1 início
    /* Função para Geração de Transições por Comandos de Habilitação
       (Algoritmo 21)                                                 */
2   Geração de Transições por Comandos de Habilitação (Conjunto de Estados com
     Necessidade de Liberação, Comandos de Chaveamento em S)
    /* Função para Geração de Transições por Comandos de Liberação
       (Algoritmo 22)                                                 */
3   Geração de Transições por Comandos de Liberação (Conjunto de Estados com
     Necessidade de Liberação)
    /* Função para Geração de Transições por Eventos do Supervisor
       (Algoritmo 23)                                                 */
4   Geração de Transições por Eventos do Supervisor (S, Estados Tipo 1, Estados Tipo
     2, Estados Tipo 3, Conjunto de Estados com Necessidade de Liberação)
```

Algoritmo 21: Geração de Transições por Comandos de Habilitação

Entrada: (*Conjunto de Estados com Necessidade de Liberação*, *Comandos de Chaveamento em S*)

início
  para todo $(x_{origem}, x_{desino}, código) \in$ *Comandos de Chaveamento em S* **faça**
    se x_{origem} ="*Desabilitado*" **então**
      origem $\leftarrow$ "*Desabilitado*"
    senão se $x_{origem} \in$ *Conjunto de Estados com Necessidade de Liberação* **então**
      origem $\leftarrow$ "*Liberar_Estado_*" + *String*(x_{origem})
    senão
      origem $\leftarrow$ "*Estado_*" + *String*(x_{origem})
    se $x_{destino}$ ="*Desabilitado*" **então**
      destino $\leftarrow$ "*Desabilitado*"
    senão se $x_{destino} \in$ *Conjunto de Estados com Necessidade de Liberação* **então**
      destino $\leftarrow$ "*Liberar_Estado_*" + *String*($x_{destino}$)
    senão
      destino $\leftarrow$ "*Estado_*" + *String*($x_{destino}$)
    escrever elemento vazio *ECTransition* acompanhado dos atributos *Comment* (comentário), *Condition="COM_HAB[ESTADO="+ String(código) +"]"* (condição de transição), *Destination=destino*, *Source=origem*, *x* e *y* (coordenadas de posicionamento)

Algoritmo 22: Geração de Transições por Comandos de Liberação

Entrada: (*Conjunto de Estados com Necessidade de Liberação*)

início
  para todo $x_S \in$ *Conjunto de Estados com Necessidade de Liberação* **faça**
    origem $\leftarrow$ "*Liberar_Estado_*" + *String*(x_S)
    destino $\leftarrow$ "*Estado_*" + *String*(x_S)
    escrever elemento vazio *ECTransition* acompanhado dos atributos *Comment* (comentário), *Condition="LIBERAR"* (condição de transição), *Destination=destino*, *Source=origem*, *x* e *y* (coordenadas de posicionamento)

Algoritmo 23: Geração de Transições por Eventos do Supervisor

Entrada: (*S*, *Estados Tipo 1*, *Estados Tipo 2*, *Estados Tipo 3*, *Conjunto de Estados com Necessidade de Liberação*)

```
1  início
       /* Estados Tipo 1 → conjunto de estados com transições de
          saídas por eventos não controláveis                          */
       /* Estados Tipo 2 → conjunto de estados com transição de saída
          por evento controlável                                       */
       /* Estados Tipo 3 → conjunto de estados com transições de
          saídas por evento controlável e por eventos não controláveis
          */
       /* S = (X_S, Σ_S, δ_S, Γ_S, x_S0, X_Sm)                         */
2      obter o autômato S
       /* x_S1, x_S2 ∈ X_S  e  e ∈ Σ_S                                 */
3      para todo x_S1 ∈ X_S faça
4          se x_S1 ∈ Estados Tipo 1 então        /* ∀δ(x_S1, e)!, e ∈ Σ_nc */
5              para todo δ(x_S1, e) = x_S2 faça
6                  se x_S1 ≠ x_S2 então
                       /* Função para Geração de Transições Originadas em
                          Estado Tipo 1 (Algoritmo 24)                 */
7                      Geração de Transições Originadas em Estado Tipo 1 (x_S1, x_S2, e,
                        Conjunto de Estados com Necessidade de Liberação)
8          senão se x_S1 ∈ Estados Tipo 2 então   /* δ(x_S1, e)!, e ∈ Σ_c */
9              se x_S1 ≠ x_S2 então
                   /* Função para Geração de Transições Originadas em
                      Estado Tipo 2 (Algoritmo 25)                     */
10                 Geração de Transições Originadas em Estado Tipo 2 (x_S1, x_S2, Conjunto
                    de Estados com Necessidade de Liberação)
11         senão   /* [Γ(x_S1) ∩ Σ_c ≠ ∅] ∧ [Γ(x_S1) ∩ Σ_nc ≠ ∅]  (Estado Tipo 3) */
12             contagem ← 1
13             para todo δ(x_S1, e) = x_S2 faça
14                 se x_S1 ≠ x_S2 então
                       /* Função para Geração de Transições Originadas em
                          Estado Tipo 3 (Algoritmo 26)                 */
15                     contagem ← Geração de Transições Originadas em Estado Tipo 3
                        (x_S1, x_S2, e, Conjunto de Estados com Necessidade de Liberação,
                        contagem)
```

Algoritmo 24: Geração de Transições Originadas em Estado Tipo 1

Entrada: (x_{S_1}, x_{S_2}, *e*, *Conjunto de Estados com Necessidade de Liberação*)

início
  se $x_{S_2} \notin$ *Conjunto de Estados com Necessidade de Liberação* **então**
    escrever elemento vazio *ECTransition* acompanhado dos atributos *Comment* (comentário), *Condition=String(e)* (condição de transição), *Destination=String* (x_{S_2}), *Source=String* (x_{S_1}), *x* e *y* (coordenadas de posicionamento)
  senão
    destino ← *"Liberar_Estado_"+ String* (x_{S_2})
    escrever elemento vazio *ECTransition* acompanhado dos atributos *Comment* (comentário), *Condition=String(e)* (condição de transição), *Destination=destino*, *Source=String* (x_{S_1}), *x* e *y* (coordenadas de posicionamento)

Algoritmo 25: Geração de Transições Originadas em Estado Tipo 2

Entrada: (x_{S_1}, x_{S_2}, *Conjunto de Estados com Necessidade de Liberação*)

início
  se $x_{S_2} \notin$ *Conjunto de Estados com Necessidade de Liberação* **então**
    escrever elemento vazio *ECTransition* acompanhado dos atributos *Comment* (comentário), *Condition="1"* (condição de transição), *Destination=String* (x_{S_2}), *Source=String* (x_{S_1}), *x* e *y* (coordenadas de posicionamento)
  senão
    destino ← *"Liberar_Estado_"+ String* (x_{S_2})
    escrever elemento vazio *ECTransition* acompanhado dos atributos *Comment* (comentário), *Condition="1"* (condição de transição), *Destination=destino*, *Source=String* (x_{S_1}), *x* e *y* (coordenadas de posicionamento)

Exemplo 3. Considerando o modelo de CTF obtido pelo FCSS no Exemplo 2, a etapa de síntese de lógica de controle da RAFAH pode ser utilizada para obter os blocos de função dos supervisores S_1 e S_2 para o processo de mistura da Figura 22. Os dados, que estão inseridos no conjunto *Informações para os BFBs dos Supervisores* e que são utilizados na síntese dos supervisores, são apresentados a seguir.

Dados referentes ao supervisor 1

Identificação do Supervisor	*Dados para a Controlabilidade Segura em S*
S_1	$([\{1,Y,nB\}],VA\_abrir)$ $([\{9,Y,nB\}],P\_ligar)$
Comandos de Chaveamento em S	
$(Desabilitado,1,1)$	$(Desabilitado,9,9)$
$(1,Desabilitado,0)$	$(9,Desabilitado,0)$

Algoritmo 26: Geração de Transições Originadas em Estado Tipo 3

Entrada: (x_{S_1}, x_{S_2}, e, *Conjunto de Estados com Necessidade de Liberação*, *contagem*)
Saída: (*contagem*)

início
  $origem \leftarrow x_{S_1}$
  $intermediário \leftarrow$ *"Estado_Intermediario_"* $+ String(x_{S_1}) +$ *"_"* $+ String(contagem)$
  $contagem \leftarrow contagem + 1$
  se $x_{S_2} \notin$ *Conjunto de Estados com Necessidade de Liberação* **então**
    $destino \leftarrow x_{S_2}$
  senão
    $destino \leftarrow$ *"Liberar_Estado_"*+ $String\ (x_{S_2})$
  se $e \in \Sigma_{nc}$ **então**
    escrever elemento vazio *ECTransition* acompanhado dos atributos *Comment* (comentário), *Condition=String(e)* (condição de transição), *Destination=intermediário*, *Source=origem*, *x* e *y* (coordenadas de posicionamento)
    escrever elemento vazio *ECTransition* acompanhado dos atributos *Comment* (comentário), *Condition="SC_In"* (condição de transição), *Destination=destino*, *Source=intermediário*, *x* e *y* (coordenadas de posicionamento)
  senão
    escrever elemento vazio *ECTransition* acompanhado dos atributos *Comment* (comentário), *Condition="SC_In"* (condição de transição), *Destination=intermediário*, *Source=origem*, *x* e *y* (coordenadas de posicionamento)
    escrever elemento vazio *ECTransition* acompanhado dos atributos *Comment* (comentário), *Condition="1"* (condição de transição), *Destination=destino*, *Source=intermediário*, *x* e *y* (coordenadas de posicionamento)

Dados referentes ao supervisor 2

Identificação do Supervisor	*Dados para a Controlabilidade Segura em S*
S_2	$([\{1,Y,nB\}],VB\_abrir)$ $([\{9,Y,nB\}],P\_ligar)$
Comandos de Chaveamento em S	
$(Desabilitado,1,1)$	$(Desabilitado,9,9)$
$(1,Desabilitado,0)$	$(9,Desabilitado,0)$

A Figura 39 mostra a estrutura do bloco de função obtida para o supervisor S_1. O desenvolvimento do bloco inicia com a leitura dos dados que estão disponíveis no conjunto denominado *Informações para os BFBs dos Supervisores* e que auxiliam na sua construção (Algoritmo 13). Na sequência, os estados são classificados por tipo no Algoritmo 14, o que faz com que os estados 2, 5, 8 e 10 sejam classificados no conjunto *Estados Tipo 1* e os demais estados sejam classificados no conjunto *Estados Tipo 2*. Além disso, o Algoritmo 14 ainda classifica os estados 1 e 9 no *Conjunto de Estados com Necessidade de Liberação*. A interface do bloco é gerada no Algoritmo 15, sendo composta pelo evento de entrada que representa os sinais de chaveamento

(evento *COM_HAB*), pelos eventos existentes em S_1 (eventos não controláveis como eventos de entrada e eventos controláveis como eventos de saída) e pelo evento de entrada para liberação de estados (evento *LIBERAR*). O evento de liberação de estados no supervisor é gerado para efetuar a liberação dos estados 1 e 9 no bloco, conforme informação existente no *Conjunto de Estados com Necessidade de Liberação*. O desenvolvimento da interface finaliza com a criação da variável de entrada *ESTADO*. Após o desenvolvimento da interface, é realizado o desenvolvimento do ECC do bloco, tendo como primeira etapa a geração dos seus estados. O primeiro estado gerado é o estado de desabilitação do supervisor (Algoritmo 16), denominado de *Desabilitado*. Na sequência, são gerados os estados de liberação *Liberar_Estado_1* e *Liberar_Estado_9* no Algoritmo 17. Por fim, há a criação dos estados provenientes do supervisor S_1 (Algoritmo 18). A geração das transições finaliza o desenvolvimento do ECC, sendo geradas em sequência: as transições por comandos de chaveamento (transições com o evento *COM_HAB* geradas no Algoritmo 21), as transições por comandos de liberação (transições com o evento *LIBERAR* geradas no Algoritmo 22) e as transições por eventos do supervisor (demais transições do ECC geradas nos algoritmos 23, 24, 25 e 26). O conteúdo dos arquivos *XML* dos supervisores são apresentados no Apêndice A.

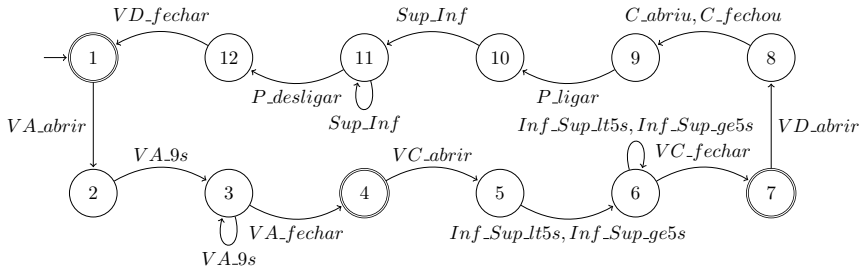
4.2.4.2 Síntese de Diagnosticadores

Na etapa de síntese de lógica de controle da arquitetura RAFAH, a síntese dos diagnosticadores em blocos de função da IEC 61499 segue uma sistemática similar a que foi efetuada na síntese dos supervisores, modelando sua ação no ECC do bloco. Para executar esta implementação, é utilizado um conjunto de informações, denominado de *Informações para os BFBs dos Diagnosticadores*, proveniente do FCSS. A ação de geração dos blocos de função dos diagnosticadores é executada de forma modularizada pelos Algoritmos 27 – 32, conforme mostra o diagrama da Figura 41.

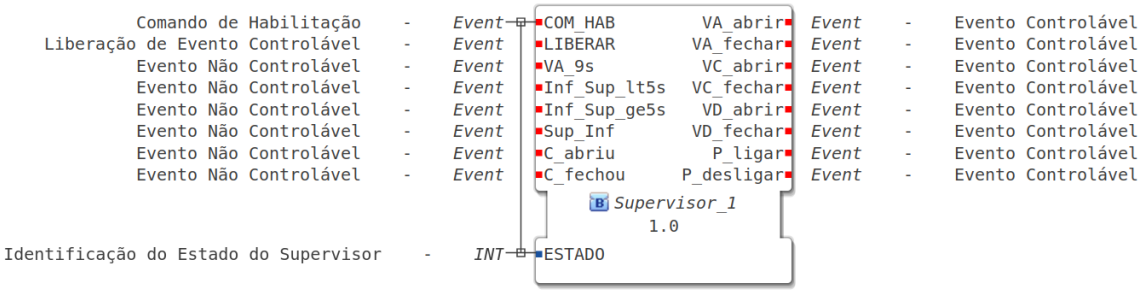
A geração dos arquivos dos blocos de função dos diagnosticadores inicia pela análise dos dados presentes no conjunto *Informações para os BFBs dos Diagnosticadores* no Algoritmo 27. Neste conjunto, existe um grupo de dados, para cada falha, composto pelas seguintes informações:

1. A identificação da falha;
2. O diagnosticador obtido para a falha no FAS;
3. O Supervisor no qual a falha é avaliada;
4. Os dados obtidos no FAS para efetuar, em função da falha, controlabilidade segura no sistema;

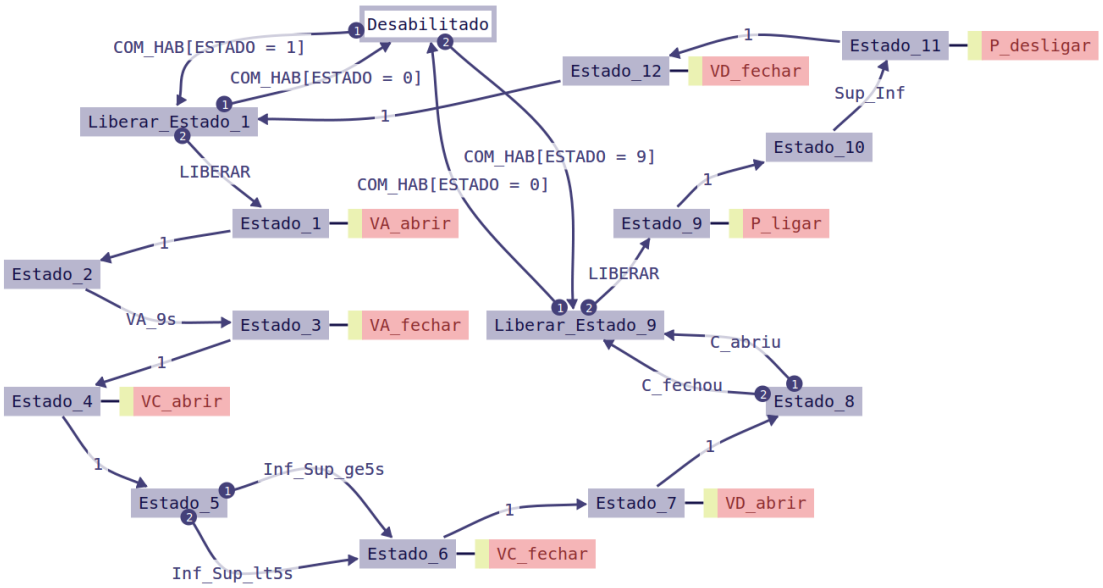
Figura 39 – Bloco de função do supervisor S_1 no CTF.



(a) Supervisor 1 (S_1) do processo de mistura, com $\Sigma_{nc} = \{VA\_9s, Inf\_Sup\_ge5s, Inf\_Sup\_lt5s, Sup\_Inf, C\_abriu, C\_fechou\}$.



(b) Interface de S_1.



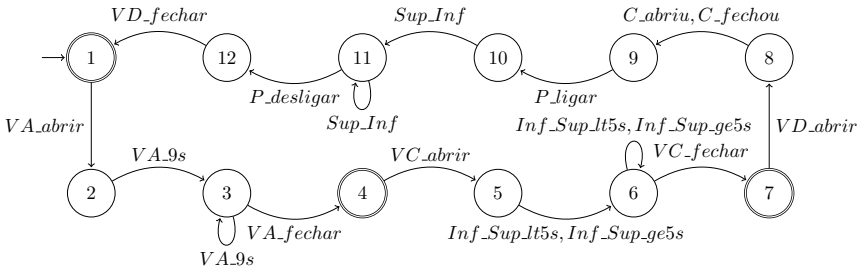
(c) ECC de S_1.

Fonte: Elaborada pelo autor (2022).

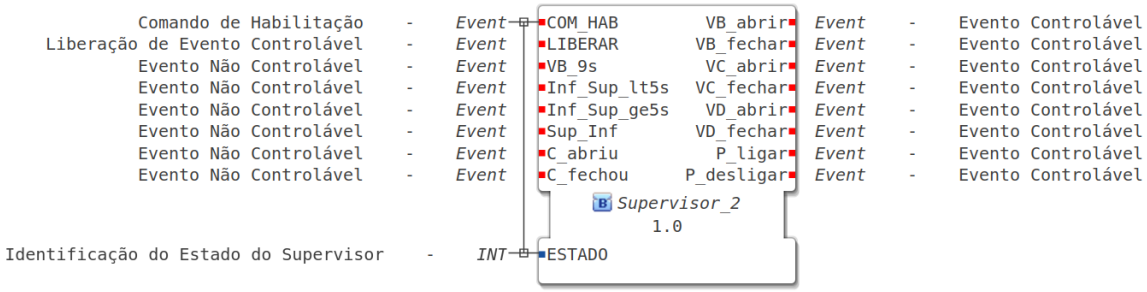
5. Um conjunto denominado de *CESRD* com os estados do diagnosticador que não devem ser representados no bloco de função, estados do diagnosticador que representam estados do sistema posteriores a ação de chaveamento;

6. Os comandos de chaveamento obtidos na análise realizada pelo FCSS.

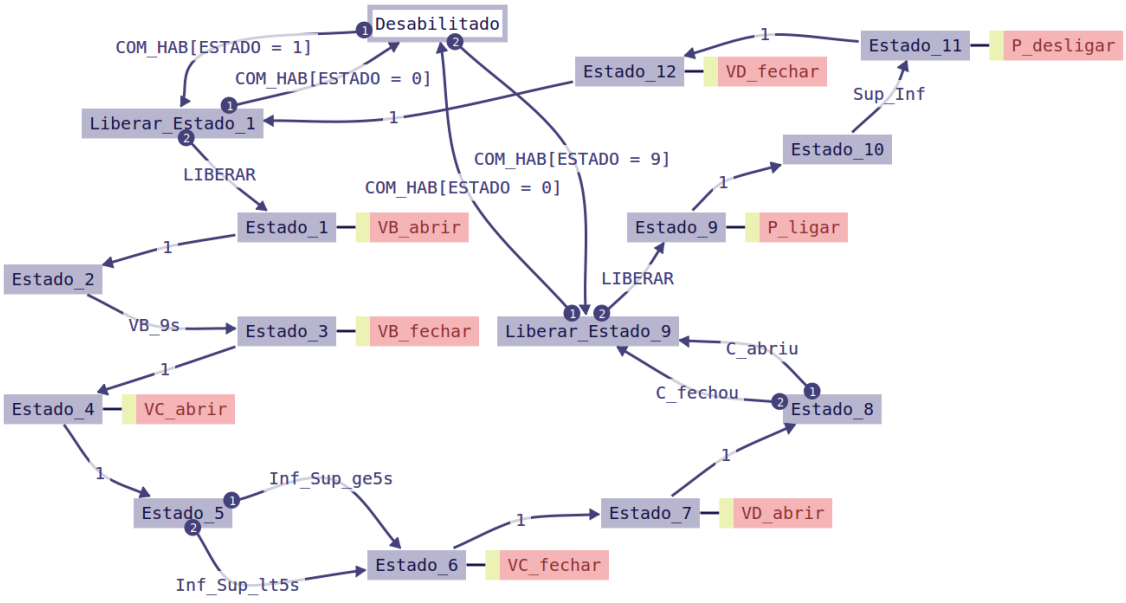
Figura 40 – Bloco de função do supervisor S_2 no CTF.



(a) Supervisor 2 (S_2) do processo de mistura, com $\Sigma_{nc} = \{VB\_9s, Inf\_Sup\_ge5s, Inf\_Sup\_lt5s, Sup\_Inf, C\_abriu, C\_fechou\}$.



(b) Interface de S_2.



(c) ECC de S_2.

Fonte: Elaborada pelo autor (2022).

O Algoritmo 27 ainda fica responsável por gerar as informações básicas de definição do bloco de função do diagnosticador e por efetuar a chamada dos algoritmos reponsáveis pela geração da interface do bloco (Algoritmo 28) e pela geração do ECC (Algoritmos 30 e 32).

Algoritmo 27: Geração dos Arquivos dos Blocos de Função dos Diagnosticadores

Entrada: (*Informações para os BFBs dos Diagnosticadores*)

1 **início**

`/* CESRD` $\rightarrow$ `Conjunto de Estados sem Representação no Diagnosticador */`

2 **para todo** *(f, D, S, Dados para a Controlabilidade Segura em S, CESRD, Comandos de Chaveamento em D)* $\in$ *Informações para os BFBs dos Diagnosticadores* **faça**

`/* Arquivo xml */`

3 escrever cabeçalho

4 escrever a definição de tipo de documento (DTD) que valida o bloco de função

5 escrever elemento inicial *FBType* (definição do bloco) acompanhado dos atributos *Comment* (comentário) e *Name* (nome do bloco)

6 escrever elemento vazio *Identification* acompanhado do atributo *Standard* para identificar a versão da norma à qual o bloco é compatível

7 escrever elemento vazio *VersionInfo* acompanhado dos atributos *Author* (autor), *Date* (data) e *Version* (versão) para registrar informações relacionadas à versão do arquivo

8 escrever elemento inicial *InterfaceList* para identificar os elementos que compõem a interface do bloco

`/* Função para geração da interface do bloco de função do diagnosticador (Algoritmo 28) */`

9 *(Estados para Liberar, Estados para Chavear)* $\leftarrow$ *Geração da Interface do Bloco de Função do Diagnosticador (D, S, Dados para a Controlabilidade Segura em S)*

10 escrever elemento final *InterfaceList*

11 escrever elemento inicial *BasicFB* para identificar a estrutura interna do bloco

12 escrever elemento inicial *ECC* para identificar o gráfico de controle de execução

`/* Funções para geração da estrutura interna do bloco de função (Algoritmos 30 e 32) */`

13 *Geração dos Estados no ECC do Bloco de Função do Diagnosticador (f, D, CESRD, Estados para Liberar, Estados para Chavear)*

14 *Geração das Transições no ECC do Bloco de Função do Diagnosticador (f, D, CESRD, Comandos de Chaveamento em D)*

15 escrever elemento final *ECC*

16 escrever elemento final *BasicFB*

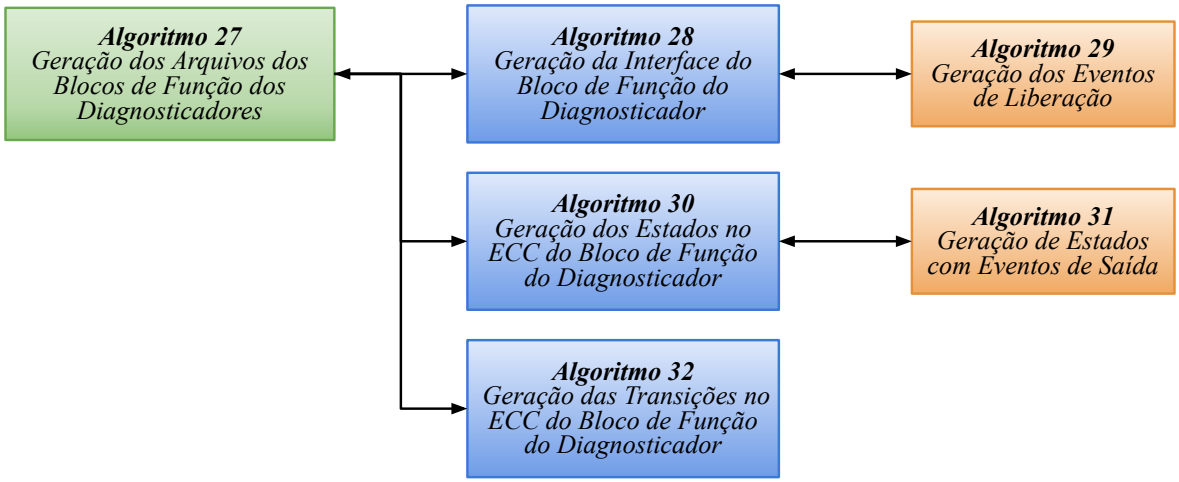
17 escrever elemento final *FBType*

Algoritmo 28: Geração da Interface do Bloco de Função do Diagnosticador

Entrada: (*D, S, Dados para a Controlabilidade Segura em S*)
Saída: *(Estados para Liberar, Estados para Chavear)*

1 **início**
  /* $D = (X_D, \Sigma_o, \delta_D, \Gamma_D, x_{D_0}, X_{D_m})$ */
2 obter o autômato *D*
3 escrever elemento inicial *EventInputs* para identificar os eventos de entrada
4 escrever elemento inicial *Event* acompanhado dos atributos *Comment* (comentário), *Name="COM_HAB"* (evento *COM_HAB*) e *Type* (tipo do evento) para definir o comando de habilitação do diagnosticador
5 escrever elemento vazio *With* acompanhado do atributo *Var="ESTADO"*
6 escrever elemento final *Event*
7 **para todo** $e \in \Sigma_D$ **faça**
8 escrever elemento inicial *Event* acompanhado dos atributos *Comment* (comentário), *Name=String(e)* (evento *e*) e *Type* (tipo do evento)
9 escrever elemento final *Event*
10 escrever elemento final *EventInputs*
11 escrever elemento inicial *EventOutputs* para identificar os eventos de saída
12 escrever elemento inicial *Event* acompanhado dos atributos *Comment* (comentário), *Name="DIAGNOSE"* (sinalização da diagnose da falha) e *Type* (tipo do evento)
13 escrever elemento final *Event*
14 escrever elemento inicial *Event* acompanhado dos atributos *Comment* (comentário), *Name="CHAVEAR"* (sinalização para que haja chaveamento no sistema) e *Type* (tipo do evento)
15 escrever elemento final *Event*
  `/* Função para geração dos eventos de liberação (Algoritmo 29) */`
16 *(Estados para Liberar, Estados para Chavear)* ← *Geração dos Eventos de Liberação (S, Dados para a Controlabilidade Segura em S)*
17 escrever elemento final *EventOutputs*
18 escrever elemento inicial *InputVars* para identificar as variáveis de entrada
19 escrever elemento vazio *VarDeclaration* acompanhado dos atributos *Comment* (comentário), *Name="ESTADO"* (indicação do estado para chaveamento no bloco) e *Type="INT"* (variável do tipo inteira)
20 escrever elemento final *InputVars*

Figura 41 – Diagrama de modularização das operações executadas na geração dos blocos de função dos diagnosticadores do CTF.



Fonte: Elaborada pelo autor (2022).

Devido ao fato do diagnosticador operar monitorando as sequências de eventos que ocorrem no SPA, temos que todos os eventos do sistema, controláveis ou não controláveis, tornam-se eventos de entrada no bloco de função que representará o diagnosticador. Além desses eventos, também existe o evento de entrada *COM_HAB* que serve, para habilitar e desabilitar a ação do diagnosticador no sistema. Por outro lado, são representadas por eventos de saída as sinalizações de diagnóstico de falha (evento *DIAGNOSE*) e de mudança da estratégia de controle (evento *CHAVEAR*). Todos estes eventos de entrada e de saída do bloco são gerados no Algoritmo 28. Completando o desenvolvimento da interface do bloco, o Algoritmo 29 identifica e classifica, em conjuntos, os estados do diagnosticador que devem publicar eventos de chaveamento para o módulo de chaveamento. Este algoritmo ainda é responsável por gerar, quando necessário, os eventos de saída de liberação para os estados do supervisor. O Algoritmo 14, já utilizado anteriormente na síntese dos supervisores, fica responsável por identificar os estados do supervisor que devem receber o sinal de liberação.

Os estados do ECC do bloco de função são gerados pelos Algoritmos 30 e 31. O Algoritmo 30 é responsável pela geração dos estados que não publicam eventos de saída e o Algoritmo 31 é responsável por gerar os estados que publicam.

O Algoritmo 32 gera as transições do ECC. No algoritmo, existem dois laços de criação de transições, sendo o primeiro responsável por gerar as transições relacionadas aos comandos de chaveamento e o segundo responsável por gerar as transições existentes entre estados do diagnosticador.

O Exemplo 4 a seguir mostra como a etapa de síntese de lógica de controle obtém os blocos de função dos diagnosticadores para o processo de mistura da Figura 22.

Exemplo 4. Considerando o modelo de CTF obtido pelo FTCSS no Exemplo 3, a etapa de

Algoritmo 29: Geração dos Eventos de Liberação

Entrada: (*S, Dados para a Controlabilidade Segura em S*)
Saída: *(Estados para Liberar, Estados para Chavear)*

início
  /* *Estados Tipo 2* → conjunto de estados com transição de saída por evento controlável */
  /* Função para classificação dos estados do supervisor (Algoritmo 14) */
  ... *Estados Tipo 2* ... ← *Classificação dos Estados do Supervisor (S, Dados para a Controlabilidade Segura em S)*
  Estados para Chavear ← ∅
  Estados para Liberar ← ∅
  /* $x_D \in X_D, x_D = \{\{x,Y,nB\}\}$ */
  para todo $(x_D, \Sigma_{liberar}) \in$ *Dados para a Controlabilidade Segura* **faça**
    Estados para Chavear ← *Estados para Chavear* $\cup \{x_D\}$
    se $x \in$ *Estados Tipo 2* **então**
      se $\{\{x,N,nB\}\} \in X_D$ **então**
        Estados para Liberar ← *Estados para Liberar* $\cup \{\{x,N,nB\}\}$
        escrever elemento inicial *Event* acompanhado dos atributos *Comment* (comentário), *Name="LIBERAR_ESTADO_"+ String(x)* (liberar a ocorrência de evento controlável) e *Type* (tipo do evento)
        escrever elemento final *Event*
      se $\{\{x,N,nB\},\{x,Y,nB\}\} \in X_D$ **então**
        Estados para Liberar ← *Estados para Liberar* $\cup \{\{x,N,nB\},\{x,Y,nB\}\}$
        escrever elemento inicial *Event* acompanhado dos atributos *Comment* (comentário), *Name="LIBERAR_ESTADO_"+ String(x)* (liberar a ocorrência de evento controlável) e *Type* (tipo do evento)
        escrever elemento final *Event*

síntese de lógica de controle da RAFAH pode ser utilizada para obter os blocos de função dos diagnosticadores $D_{f_{VA}}$, $D_{f_{VD_{S_1}}}$, $D_{f_{VB}}$ e $D_{f_{VD_{S_2}}}$. Os dados, que estão inseridos no conjunto *Informações para os BFBs dos Diagnosticadores* e que são utilizados na síntese dos diagnosticadores, são apresentados a seguir.

Dados referentes à falha de vazamento na válvula A

Identificação da Falha	Identificação do Diagnosticador	Identificação do Supervisor
f_{VA}	$D_{f_{VA}}$	S_1
Dados para a Controlabilidade Segura em S		*CESRD*
$[\{1,Y,nB\}],\{VA\_abrir\}$		$[\{1,Y,B\}],[\{2,Y,B\}],\ldots,[\{12,Y,B\}]$
Comandos de Chaveamento em D		
$(Desabilitado,[\{1,N,nB\}],1)$		$(Desabilitado,[\{9,N,nB\}],9)$
$([\{1,Y,nB\}],Desabilitado,0)$		$([\{9,N,nB\}],Desabilitado,0)$

Algoritmo 30: Geração dos Estados no ECC do Bloco de Função do Diagnosticador

Entrada: (f, D, *CESRD*, *Estados para Liberar, Estados para Chavear*)

1 **início**

/* $D = (X_D, \Sigma_o, \delta_D, \Gamma_D, x_{D_0}, X_{D_m})$ */

2 obter o autômato D

3 escrever elemento vazio *ECState* acompanhado dos atributos *Comment* (comentário), *Name="Desabilitado"* (estado de desabilitação do diagnosticador), x e y (coordenadas de posicionamento do estado)

4 **para todo** $x_D \in X_D$ **faça**

5 **se** $x_D \notin CESRD$ **então**

6 **se** $[\forall\{x,z,w\} \in x_D, z = Y] \wedge [\exists\delta(x'_D,\_) = x_D, \forall\{x,z,w\} \in x'_D, z = N]$ **então**

7 *Diagnóstico da Falha* $\leftarrow$ *VERDADEIRO*

8 **senão**

9 *Diagnóstico da Falha* $\leftarrow$ *FALSO*

10 *rótulo* $\leftarrow$ *"Estado"*

11 **para todo** $\{x,z,nB\} \in x_D$ **faça**

12 *rótulo* $\leftarrow$ *rótulo* + *"_"* + *String(x)* + *String(z)* + *"nB"*

13 **se** *($x_D \notin$ Estados para Chavear)* $\wedge$ *($x_D \notin$ Estados para Liberar)* $\wedge$ *(Diagnóstico da Falha = FALSO)* **então**

14 escrever elemento inicial *ECState* acompanhado dos atributos *Comment* (comentário), *Name=rótulo* (estado do diagnosticador), x e y (coordenadas de posicionamento do estado)

15 escrever elemento final *ECState*

16 **senão**

`/* Função para Geração de Estados com Eventos de Saída (Algoritmo 31) */`

17 *Geração de Estados com Eventos de Saída (x_D, rótulo, Diagnóstico da Falha, Estados para Chavear, Estados para Liberar)*

Dados referentes à falha de travamento fechado na válvula D no supervisor 1

Identificação da Falha	Identificação do Diagnosticador	Identificação do Supervisor
$f_{VD_{S_1}}$	$D_{f_{VD_{S_1}}}$	S_1
Dados para a Controlabilidade Segura em S	*CESRD*	
$[\{9,Y,nB\}], \{P\_ligar\}$	$[\{1,Y,B\}], [\{2,Y,B\}], \ldots, [\{12,Y,B\}]$	
Comandos de Chaveamento em D		
$(Desabilitado, [\{1,N,nB\}, \{1,Y,nB\}], 1)$	$(Desabilitado, [\{9,N,nB\}], 9)$	
$([\{1,N,nB\}, \{1,Y,nB\}], Desabilitado, 0)$	$([\{9,Y,nB\}], Desabilitado, 0)$	

Algoritmo 31: Geração de Estados com Eventos de Saída

Entrada: (x_D, *rótulo*, *Diagnóstico da Falha*, *Estados para Chavear*, *Estados para Liberar*)

início
  escrever elemento inicial *ECState* acompanhado dos atributos *Comment* (comentário), *Name=rótulo* (estado do diagnosticador), x e y (coordenadas de posicionamento do estado)
  se *Diagnóstico da Falha = VERDADEIRO* **então**
    escrever elemento vazio *ECAction* acompanhado do atributo *Output="DIAGNOSE"* (evento de saída para sinalizar diagnose da falha)
  se $x_D \in$ *Estados para Chavear* **então**
    escrever elemento vazio *ECAction* acompanhado do atributo *Output="CHAVEAR"* (evento para sinalização de chaveamento)
  senão se $x_D \cap$ *Estados para Liberar* $\neq \emptyset$ **então**
    para todo $\{x, \_, nB\} \in x_D$ **faça**
      escrever elemento vazio *ECAction* acompanhado do atributo *Output="LIBERAR_ESTADO_"+ String(x)* (liberar a ocorrência de evento controlável)
  escrever elemento final *ECState*

Dados referentes à falha de vazamento na válvula B

Identificação da Falha	Identificação do Diagnosticador	Identificação do Supervisor
f_{VB}	$D_{f_{VB}}$	S_2
Dados para a Controlabilidade Segura em S	*CESRD*	
$[\{1,Y,nB\}],\{VB\_abrir\}$	$[\{1,Y,B\}],[\{2,Y,B\}],\ldots,[\{12,Y,B\}]$	
Comandos de Chaveamento em D		
$(Desabilitado,[\{1,N,nB\}],1)$	$(Desabilitado,[\{9,N,nB\}],9)$	
$([\{1,Y,nB\}],Desabilitado,0)$	$([\{9,N,nB\}],Desabilitado,0)$	

Dados referentes à falha de travamento fechado na válvula D no supervisor 2

Identificação da Falha	Identificação do Diagnosticador	Identificação do Supervisor
$f_{VD_{S_2}}$	$D_{f_{VD_{S_2}}}$	S_2
Dados para a Controlabilidade Segura em S	*CESRD*	
$[\{9,Y,nB\}],\{P\_ligar\}$	$[\{1,Y,B\}],[\{2,Y,B\}],\ldots,[\{12,Y,B\}]$	
Comandos de Chaveamento em D		
$(Desabilitado,[\{1,N,nB\},\{1,Y,nB\}],1)$	$(Desabilitado,[\{9,N,nB\}],9)$	
$([\{1,N,nB\},\{1,Y,nB\}],Desabilitado,0)$	$([\{9,Y,nB\}],Desabilitado,0)$	

O bloco de função do diagnosticador $D_{f_{VA}}$, obtido pelo sistema de síntese de lógica de controle, é mostrado na Figura 42. Seu desenvolvimento inicia pela obtenção dos dados específicos para sua síntese no conjunto *Informações para os BFBs dos Diagnosticadores* (Algo-

Algoritmo 32: Geração das Transições no ECC do Bloco de Função do Diagnosticador

Entrada: (f, D, *CESRD*, *Comandos de Chaveamento em D*)

início
  para todo *(*x_{origem},x_{desino},*código)* $\in$ *Comandos de Chaveamento em D* **faça**
    /* Se $x_{origem}, x_{desino} \in X_D \Rightarrow x_{origem}, x_{desino} = \{\{x,N,nB\}\}$, $\{\{x,Y,nB\}\}$ ou $\{\{x,N,nB\},\{x,Y,nB\}\}$ */
    se x_{origem} *="Desabilitado"* **então**
      origem $\leftarrow$ *"Desabilitado"*
      destino $\leftarrow$ *"Estado"*
      para todo $\{x,z,nB\} \in x_{destino}$ **faça**
        destino $\leftarrow$ *destino* + "_"+ *String(x)* + *String(z)* + *"nB"*
    senão
      origem $\leftarrow$ *"Estado"*
      para todo $\{x,z,nB\} \in x_{origem}$ **faça**
        origem $\leftarrow$ *origem* + "_"+ *String(x)* + *String(z)* + *"nB"*
      destino $\leftarrow$ *"Desabilitado"*
    escrever elemento vazio *ECTransition* acompanhado dos atributos *Comment* (comentário), *Condition="COM_HAB[ESTADO="+ String(código) + "]"* (condição de transição), *Destination=destino*, *Source=origem*, x e y (coordenadas de posicionamento)
  /* $x_{D_1}, x_{D_2} \in X_D$ e $e \in \Sigma_D$ */
  /* $x_{D_1}, x_{D_2} = \{\{x,N,nB\}\}$, $\{\{x,Y,nB\}\}$ ou $\{\{x,N,nB\},\{x,Y,nB\}\}$ */
  para todo $\delta(x_{D_1}, e) = x_{D_2}$ **faça**
    se $(x_{D_1} \notin CESRD) \wedge (x_{D_2} \notin CESRD) \wedge (x_{D_1} \neq x_{D_2})$ **então**
      origem $\leftarrow$ *"Estado"*
      destino $\leftarrow$ *"Estado"*
      para todo $\{x,z,w\} \in x_{D_1}$ **faça**
        origem $\leftarrow$ *origem* + "_"+ *String(x)* + *String(z)* + *"nB"*
      para todo $\{x,z,w\} \in x_{D_2}$ **faça**
        destino $\leftarrow$ *destino* + "_"+ *String(x)* + *String(z)* + *"nB"*
      escrever elemento vazio *ECTransition* acompanhado dos atributos *Comment* (comentário), *Condition=String(e)* (condição de transição), *Destination=destino*, *Source=origem*, x e y (coordenadas de posicionamento)

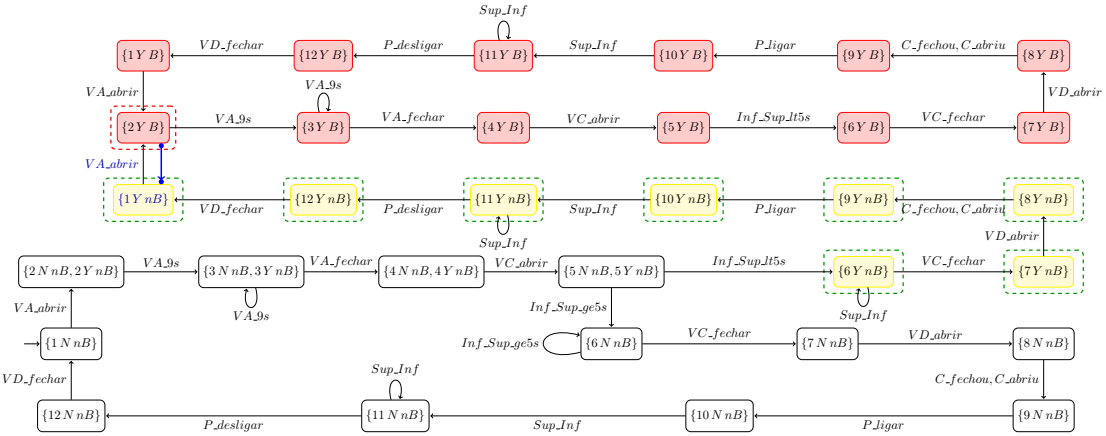
ritmo 27). No desenvolvimento de sua interface, o Algoritmo 28 gera o evento de chaveamento *COM_HAB* e os eventos pertencentes ao alfabeto do supervisor S_1, como eventos de entrada, e os eventos de sinalização *DIAGNOSE* e *CHAVEAR*, como eventos de saída. Além destes eventos, a interface ainda é composta pelo evento de saída *LIBERAR_ESTADO_1*, gerado no Algoritmo 29 por meio da análise da informação $([\{1,Y,nB\}],VA\_abrir)$ no conjunto *Dados para a Controlabilidade Segura em S*. Como o elemento $1 \in \{1,Y,nB\}$ representa o estado de rótulo 1 no supervisor S_1 e este estado pertence ao *Conjunto de Estados com Necessidade de Liberação*, o Algoritmo 29 gera um evento de saída para realizar sua liberação no supervisor S_1. A interface obtida para o bloco é mostrada na Figura 42b. Na sequência, o sistema desenvolve o ECC do bloco de função (Figura 42c). Os estados do ECC que não possuem eventos de saída e os estados que possuem eventos de saída são gerados, respectivamente, pelos Algoritmos 30 e 31. O estado denominado de *Estado_6YnB* é onde ocorre a diagnose da falha, por isso há a sinalização desse evento. O conjunto *Dados para a Controlabilidade Segura em S* informa que o estado denominado de *Estado_1YnB* é um estado onde deve ocorrer sinalização de chavemanto por parte do diagnosticador, por isso o estado publica o evento *CHAVEAR*. Como na análise realizada pelo Algoritmo 29, o estado denominado de *Estado_1NnB* foi incluído no conjunto *Estados para Liberar*, o Algoritmo 31 gera o estado com a publicação do evento de saída *LIBERAR_ESTADO_1* associada a ele. Finalizando o desenvolvimento do ECC, tem-se a geração das transições no Algoritmo 32, efetuadas em duas etapas. Na primeira etapa são geradas as transições baseadas nas informações presentes no conjunto *Comandos de Chaveamento em D*. Estas são transições que executam chaveamentos de habilitação ou de desabilitação do diagnosticador e são representadas pelo evento *COM_HAB* associado a identificação do estado de destino. Na segunda etapa, as transições geradas são as transições existentes no autômato do diagnosticador $D_{f_{VA}}$.

Os desenvolvimentos dos blocos de função para os diagnosticadores $D_{f_{VD_{S_1}}}$, $D_{f_{VB_{S_2}}}$ e $D_{f_{VD_{S_2}}}$ são realizados de formas similares ao que foi apresentado para o diagnosticador $D_{f_{VAS_1}}$. Os blocos projetados para estes diagnosticadores são mostrados nas Figuras 43, 44 e 45 e os conteúdos dos arquivos *XML* dos quatro diagnosticadores são apresentados no Apêndice B.

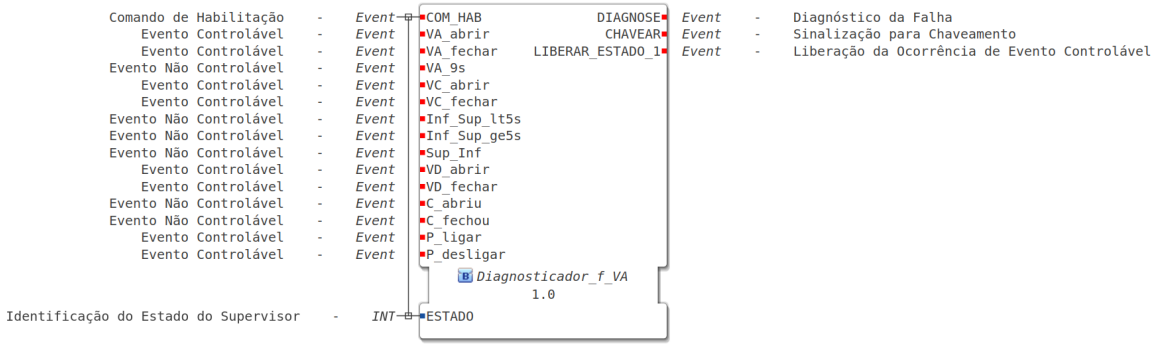
4.2.4.3 Síntese do Módulo de Chaveamento

A síntese em bloco de função do módulo de chaveamento do CTF, obtido pelo FTCSS na Seção 4.2.3, é desenvolvida a partir das informações presentes na Tabela 2. O bloco deve possuir como eventos de entrada os sinais de chaveamento provenientes de cada diagnosticador no sistema e um evento de sinalização de recuperação de componente (*REC*). A condição de recuperação torna o CTF mais flexível em termos operacionais, pois faz com que sua estrutura de controle faça uso de informações que representem recuperações de componentes do sistema. Como o CTF é desenvolvido considerando que as falhas avaliadas no sistema são do tipo contínua, a recuperação de um componente deve ocorrer por meio de uma manutenção corretiva e a sinalização desta correção deve ser efetuada de forma manual. Associadas aos eventos

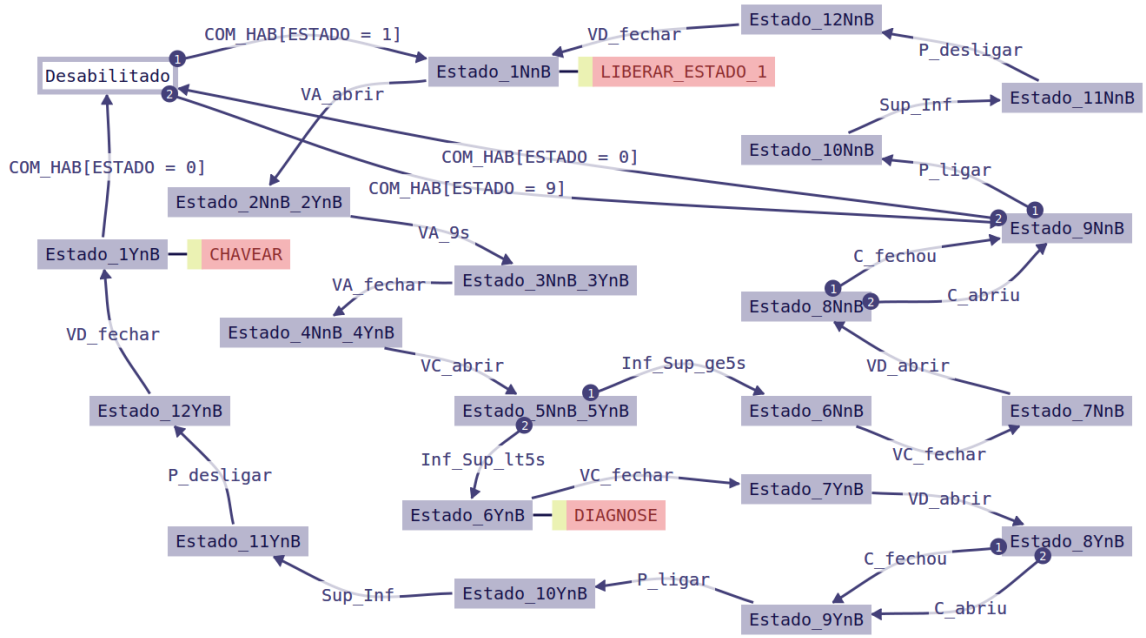
Figura 42 – Bloco de função do diagnosticador $D_{f_{VA}}$ no CTF.



(a) Diagnosticador da falha de vazamento na válvula A ($D_{f_{VA}}$).



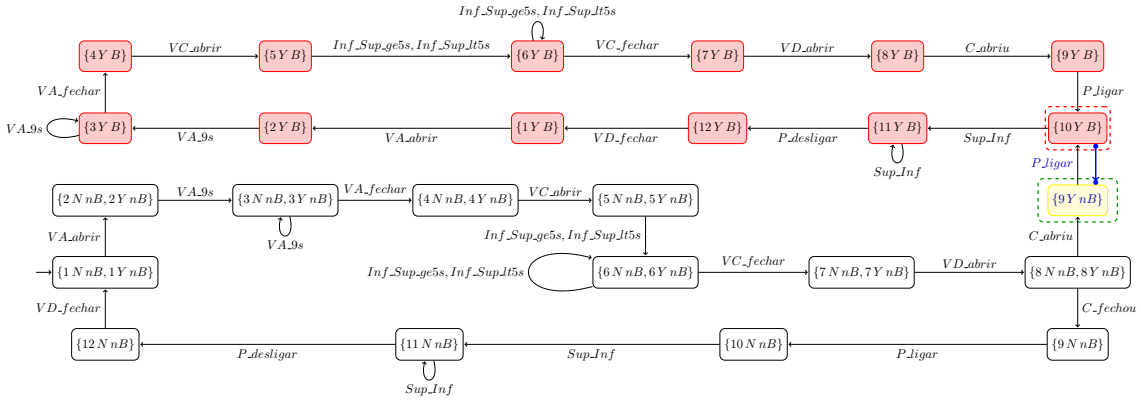
(b) Interface de $D_{f_{VA}}$.



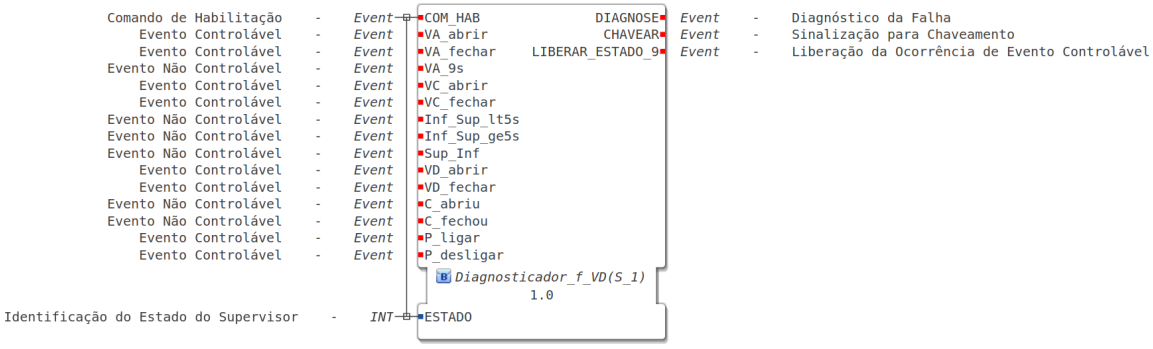
(c) ECC de $D_{f_{VA}}$.

Fonte: Elaborada pelo autor (2022).

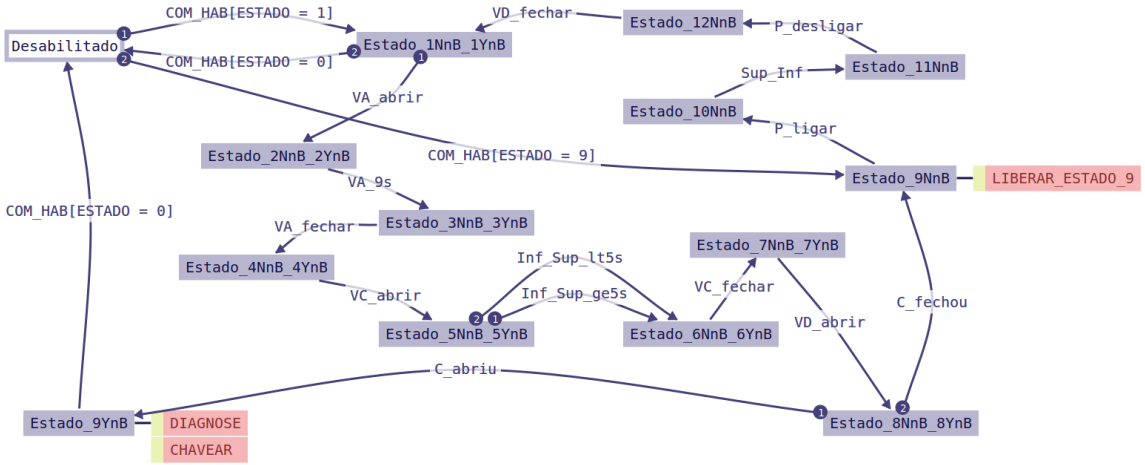
Figura 43 – Bloco de função do diagnosticador $D_{fVD_{S_1}}$ no CTF.



(a) Diagnosticador da falha de travamento fechado da válvula D ($D_{fVD_{S_1}}$).



(b) Interface de $D_{fVD_{S_1}}$.

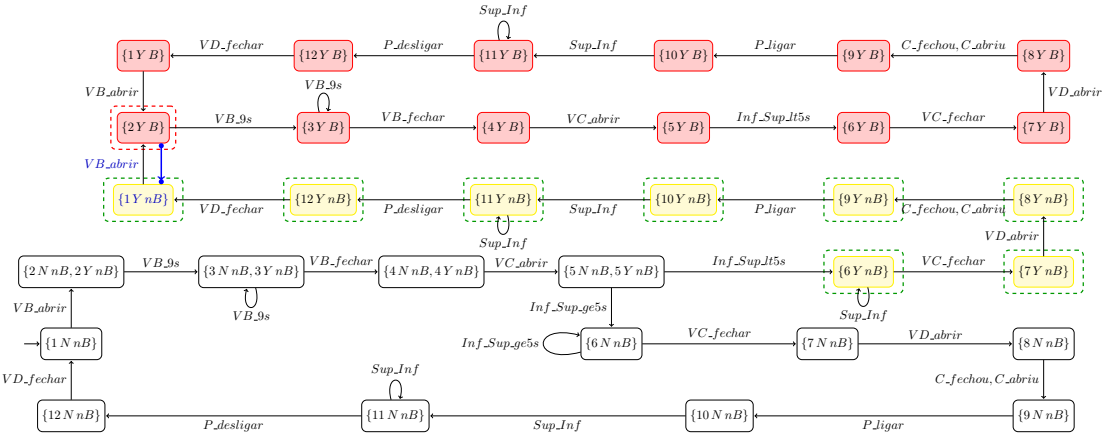


(c) ECC de $D_{fVD_{S_1}}$.

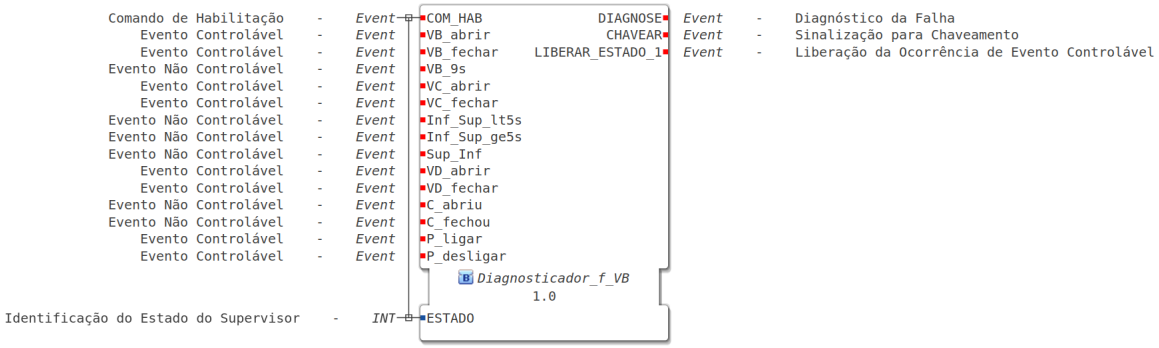
Fonte: Elaborada pelo autor (2022).

de entrada, devem existir variáveis de entrada que representam a condição física atual dos componentes que possuem falhas monitoradas, variável com valor $TRUE$ quando o componente encontra-se em estado operacional e variável com valor *FALSE* quando o componente encontra-se com falha diagnosticada. Na saída do bloco, deve existir um comando de chaveamento

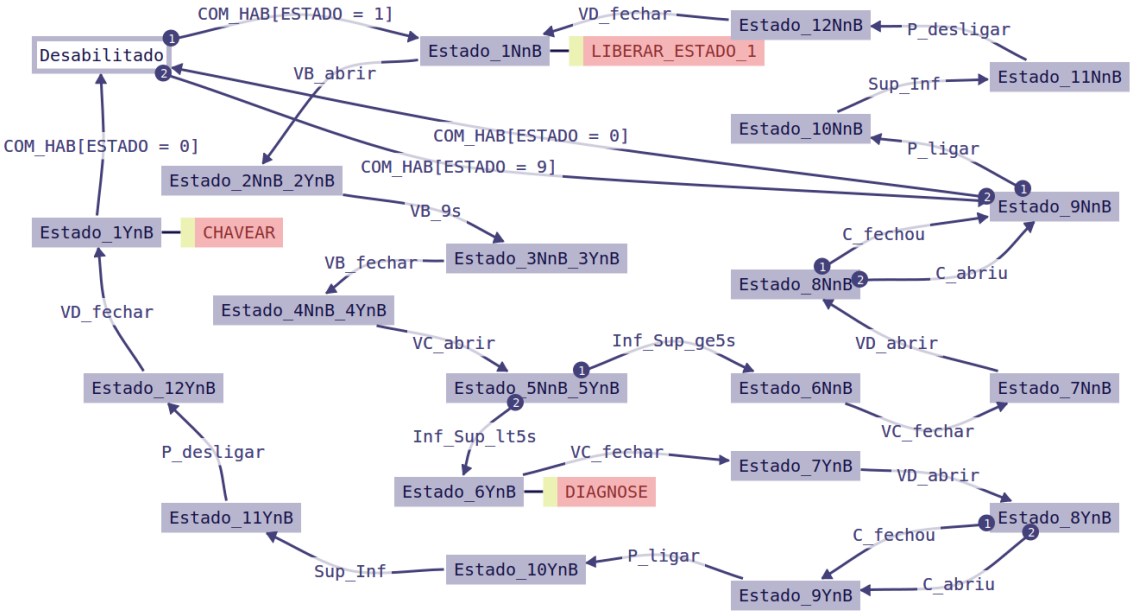
Figura 44 – Bloco de função do diagnosticador $D_{f_{VB}}$ no CTF.



(a) Diagnosticador da falha de vazamento na válvula B ($D_{f_{VB}}$).



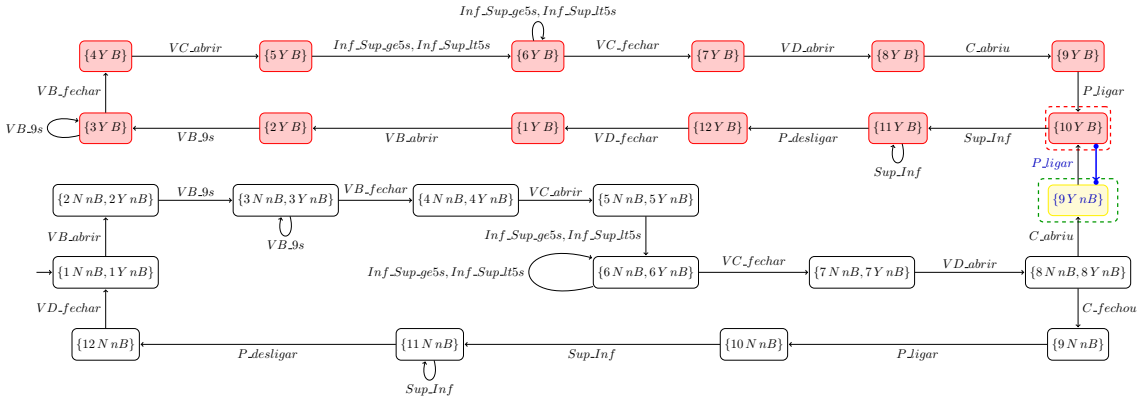
(b) Interface de $D_{f_{VB}}$.



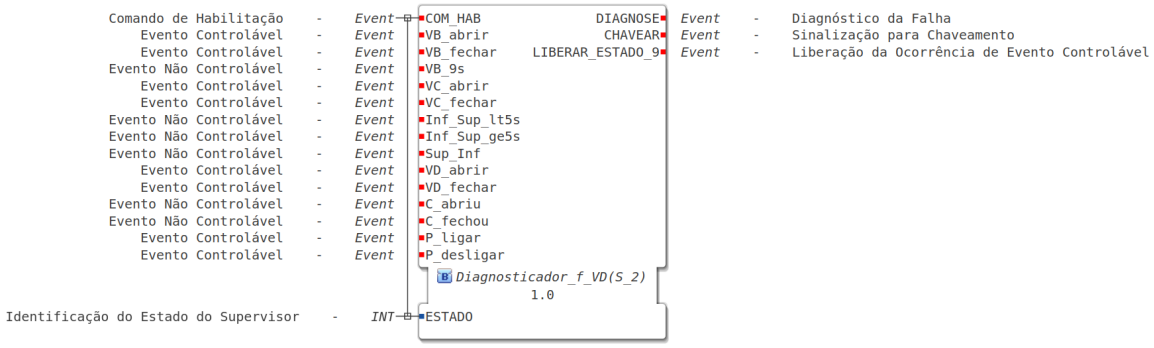
(c) ECC de $D_{f_{VB}}$.

Fonte: Elaborada pelo autor (2022).

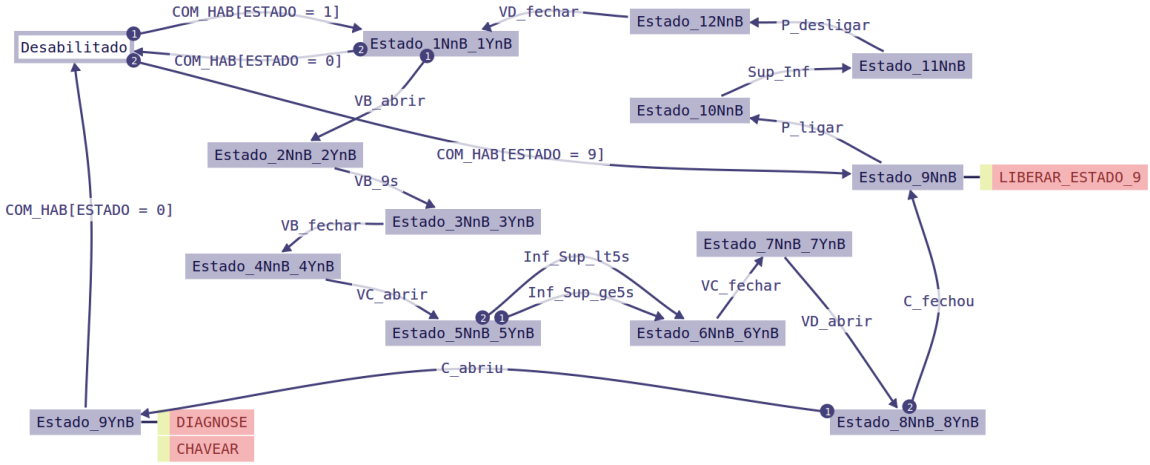
Figura 45 – Bloco de função do diagnosticador $D_{fVD_{S_2}}$ no CTF.



(a) Diagnosticador da falha de travamento fechado da válvula D ($D_{fVD_{S_2}}$).



(b) Interface de $D_{fVD_{S_2}}$.



(c) ECC de $D_{fVD_{S_2}}$.

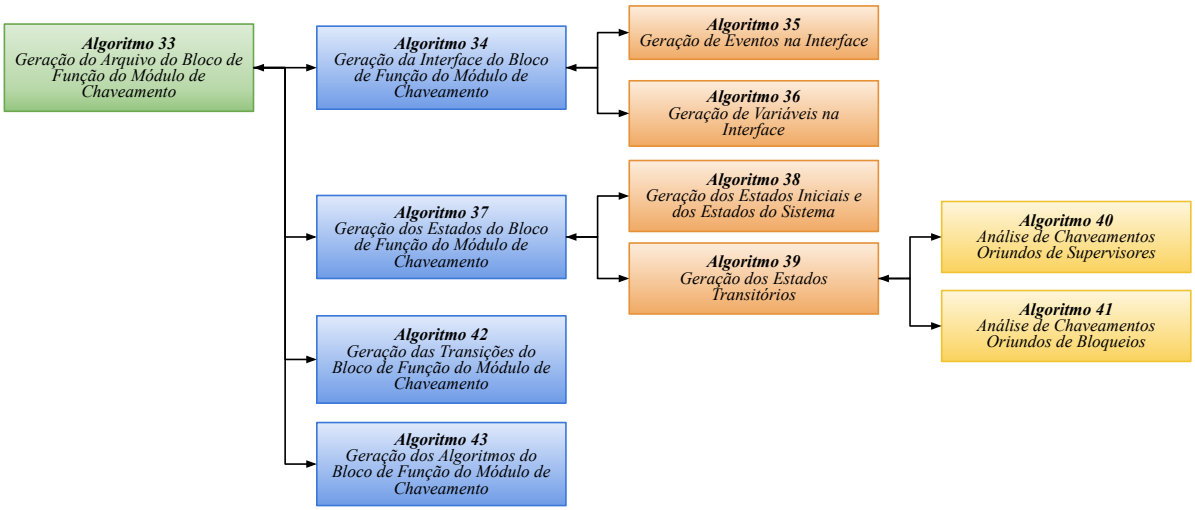
Fonte: Elaborada pelo autor (2022).

(*COM_HAB*) associado às variáveis que possuem informações específicas de chaveamento para cada supervisor $S \in S_{CTF}$ e para os diagnosticadores a ele relacionados. Na interface, ainda devem existir o evento de entrada *INIT* e o evento de saída *INITO*, o primeiro opera uma função administrativa de inicialização do bloco no sistema e o segundo indica que a inicialização foi

realizada.

O bloco de função do módulo de chaveamento é projetado de forma modularizada pelos Algoritmos 33 – 43 e segue o fluxo de operações mostrado na Figura 46. Seu desenvolvimento ocorre fundamentalmente em quatro etapas: geração da interface, geração dos estados do ECC, geração das transições do ECC e geração dos algoritmos já codificados em linguagem de programação e que serão executados pelo bloco de função.

Figura 46 – Diagrama de modularização das operações executadas na geração do bloco de função do módulo de chaveamento do CTF.



Fonte: Elaborada pelo autor (2022).

O desenvolvimento do bloco de função inicia pela definição de suas informações básicas no Algoritmo 33. Neste algoritmo, ainda existem chamadas para outros algoritmos que ficam responsáveis por estruturar a interface e os componentes internos do bloco de função.

A estruturação da interface do bloco de função é realizada pelos Algoritmos 34, 35 e 36. O Algoritmo 34 coordena a ação de estruturação da interface efetuando, em sequência, chamadas para o Algoritmo 35, responsável pelas gerações dos eventos de entrada e de saída, e para o Algoritmo 36, responsável pelas gerações das variáveis de entrada e de saída. No Algoritmo 35, ainda são criados o conjunto *variáveis de entrada* e o vetor *variáveis de saída* por meio da análise das informações existentes no conjunto denominado de *Dados de Diagnose dos Componentes do Sistema*. O conjunto *variáveis de entrada* serve para armazenar informações de todos os componentes do sistema que possuem falhas observadas no CTF. Por outro lado, o vetor *variáveis de saída* armazena informações referentes aos supervisores existentes na lista S_{CTF}.

O desenvolvimento da estrutura interna do bloco de função inicia pela geração dos estados de seu ECC. Esta ação começa no Algoritmo 37 e acontece em duas etapas. Na primeira etapa, o Algoritmo 38 gera o estado inicial do ECC (*START*), o estado para publicação da inicialização do bloco (*INITO*), o estado de atualização da condição do SPA e de definição da estratégia de controle a adotar (*UPDT*), os estados que indicam a estratégia de controle ativa e, quando existirem, estados de bloqueio. Na segunda etapa, o Algoritmo 39 fica responsável por gerar os

estados transitórios, que representam mudança na condição de operação do SPA por meio de um chaveamento no CTF. Os estados transitórios acontecem quando:

1. Existe chaveamento do estado *UPDT* para uma estratégia de controle;
2. Existe chaveamento de uma estratégia de controle para outra;
3. Existe chaveamento de um estado de bloqueio para uma estratégia de controle.

O Algoritmo 38 também fica responsável por criar a lista de algoritmos em linguagem ST (*Structured Text*, texto estruturado), os quais serão executados pelo bloco de função. A lista de algoritmos depois continuará a ser preenchida nos Algoritmos 39, 40 e 41. Esta lista, denominada ALG_{lista}, representa um conjunto cujo os elementos são tuplas formadas pelo nome do algoritmo, que deve ser criado no bloco de função, e por seu conteúdo, informações dos estados de destino nas estratégias de controle. Os algoritmos presentes na lista ALG_{lista} recebem chamadas dos estados transitórios e dos estados de bloqueio no ECC. Seu conteúdo serve de base para as gerações dos algoritmos presentes na estrutura interna do bloco e que ocorrem no Algoritmo 43. Além da lista de algoritmos, também existe uma lista, denominada de TR_{lista}, relacionada às transições que devem ocorrer no ECC. TR_{lista} é criada no Algoritmo 37 e depois é atualizada nos Algoritmos 39, 40 e 41, sendo o seu conteúdo a base para as gerações das transições que ocorrem no Algoritmo 42.

O Exemplo 5 a seguir mostra como a etapa de síntese de lógica de controle desenvolve o bloco de função do módulo de chaveamento para o processo de mistura da Figura 22.

Exemplo 5. Considerando o modelo de CTF obtido pelo FTCSS no Exemplo 2, a etapa de síntese de lógica de controle da RAFAH pode ser utilizada para desenvolver o bloco de função do módulo de chaveamento para o processo de mistura da Figura 22. Para este desenvolvimento, considera-se as informações apresentadas na Tabela 2 para o conjunto *Dados para Chaveamento* e informações apresentadas a seguir para os conjuntos S_{CTF}, BLQ_{lista} e *Dados de Diagnose dos Componentes do Sistema*.

S_{CTF}	
S_1	S_2

BLQ_{lista}	
BLQ1	*BLQ2*

Dados de Diagnose dos Componentes do Sistema

Componente	Diagnosticador
VA	*VA*
VB	*VB*
VD	*VD1, VD2*

As Figuras 47a, 47b e 47c mostram, respectivamente, a interface, o ECC e os algoritmos do ECC do bloco de função do módulo de chaveamento, obtido pelo sistema de síntese de lógica de controle. O conteúdo do arquivo *XML* do bloco de função do módulo de chaveamento do CTF é mostrado no Apêndice C. Com base nas informações apresentadas na entrada do Algoritmo 33, o desenvolvimento do bloco inicia pelas gerações dos seguintes eventos no Algoritmo 35:

1. Evento de entrada de inicialização do bloco de função (*INIT*);
2. Eventos de entrada que sinalizam diagnóstico (*SINAL_DIAG_VA, SINAL_DIAG_VB, SINAL_DIAG_VD1 e SINAL_DIAG_VD2*);
3. Evento de entrada que sinaliza recuperação de componente (*SINAL_REC*);
4. Evento de saída que indica que o bloco de função foi inicializado (*INITO*);
5. Evento de saída que realiza o chaveamento no CTF (*COM_HAB*).

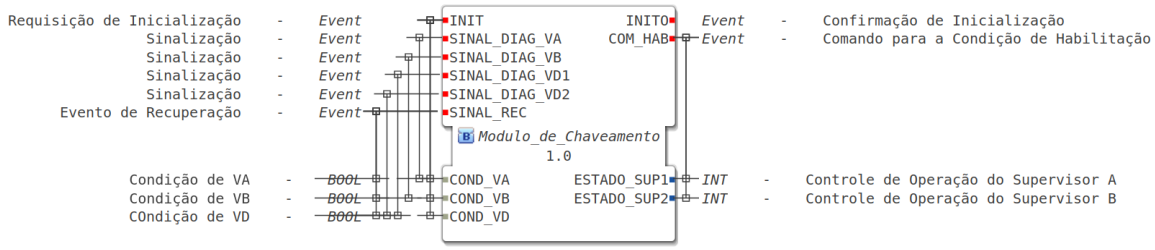
O desenvolvimento da interface finaliza no Algoritmo 36 com a geração das variáveis de entrada $COND\_VA$, $COND\_VB$ e $COND\_VD$ e das variáveis de saída $ESTADO\_SUP1$ e $ESTADO\_SUP2$. Na sequência, são gerados os estados que vão compor o ECC do bloco de função. O Algoritmo 38 gera os estados iniciais *START* e *INITO* e os seguites estados que representam o CTF:

1. O estado *UPDT* (estado de atualização das condições dos componentes do sistema e de definição de estratégia de supervisão);
2. Os estados *SUP1* e *SUP2* que representam a estratégia de controle que se encontra ativa no CTF;
3. Os estados de bloqueio *BLQ1* e *BLQ2*.

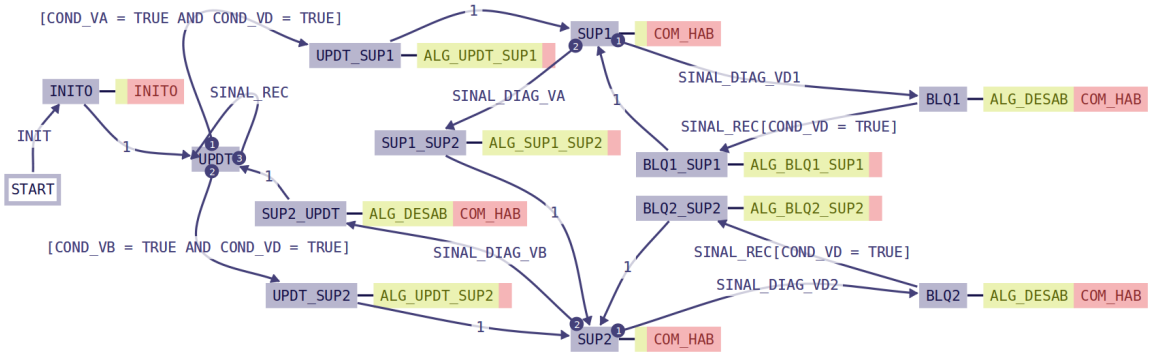
No Algoritmo 39, são gerados os estados transitórios que fazem parte de ações de chaveamento que têm o estado *UPDT* como origem, estados *UPDT_SUP1* e *UPDT_SUP2*. No Algoritmo 40, são gerados os estados transitórios que fazem parte de ações de chaveamento de uma estratégia de controle para outra estratégia de controle ou para o estado *UPDT*, estados *SUP1_SUP2* e *SUP2_UPDT*. Finalizando a criação dos estados no ECC, o Algoritmo 41 gera os estados transitórios que fazem parte de ações de chaveamento que têm como origem condições de bloqueio, estados *BLQ1_SUP1* e *BLQ2_SUP2*. Enquanto os estados do ECC estão sendo gerados, são obtidas informações referentes as suas transições e referentes as condições de atualização das estratégias de controle. Estas informações são incorporadas as listas TR_{lista} e ALG_{lista} respectivamente e servem para gerar as transições no Algoritmo 42 e os algoritmos do ECC no Algoritmo 43. São exemplos de informações incorporadas a lista TR_{lista} os dados (*SUP1*, *SUP1_SUP2*, *SINAL_DIAG_VA*) e (*SUP1_SUP2*, *SUP2*, "1"), que são produzidos na linha 15 do

Algoritmo 40 e que representam o chaveamento da estratégia de controle que adota o supervisor 1 para a estratégia de controle que adota o supervisor 2. Em relação à ALG_{lista}, um exemplo de informação incorporada a lista é o algoritmo denominado de *ALG_DESAB*, que tem seu conteúdo desenvolvido na linha 22 do Algoritmo 40.

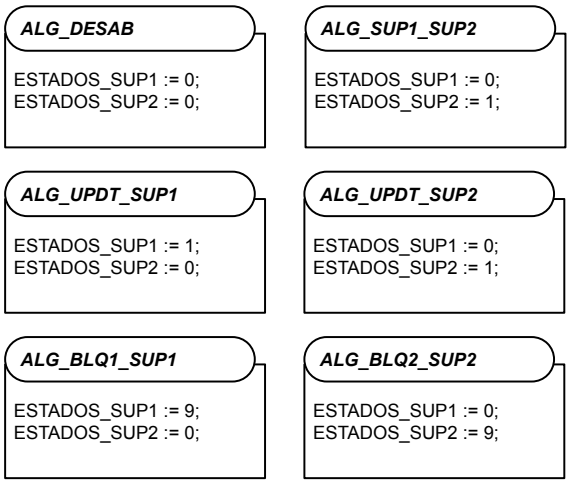
Figura 47 – Bloco de função do módulo de chavemaneto do CTF projetado para o processo de mistura da Figura 22.



(a) Interface do módulo de chaveamento.



(b) ECC do módulo de chaveamento.



(c) Algoritmos invocados pelo ECC do módulo de chaveamento.

Fonte: Elaborada pelo autor (2022).

Como forma de ilustrar a integração dos blocos do modelo de CTF projetado para o processo de mistura da Figura 22, a Figura 48 mostra como os blocos devem ser interligados na aplicação do projeto de controle na IEC 61499.

Algoritmo 33: Geração do Arquivo do BFB do Módulo de Chaveamento

Entrada: (S_{CTF}, BLQ_{lista}, *Dados de Diagnose dos Componentes do Sistema*, *Dados para Chaveamento*)

1 **início**

```
/* Arquivo xml                                                    */
```

2 escrever cabeçalho

3 escrever elemento inicial *FBType* (definição do bloco) acompanhado dos atributos *Comment* (comentário) e *Name* (nome do bloco)

4 escrever elemento vazio *Identification* acompanhado do atributo *Standard* para identificar a versão da norma à qual o bloco é compatível

5 escrever elemento vazio *VersionInfo* acompanhado dos atributos *Author* (autor), *Date* (data) e *Version* (versão) para registrar informações relacionadas à versão do arquivo

6 escrever elemento inicial *InterfaceList* para identificar os elementos que compõem a interface do bloco

```
/* Função para geração da interface do bloco de função do módulo
   de chaveamento (Algoritmo 34)                                  */
```

7 *variáveis de saída* ← *Geração da Interface do Bloco de Função do Módulo de Chaveamento (S_{CTF}, Dados de Diagnose dos Componentes do Sistema)*

8 escrever elemento final *InterfaceList*

9 escrever elemento inicial *BasicFB* para identificar a estrutura interna do bloco

10 escrever elemento inicial *ECC* para identificar o gráfico de controle de execução

```
/* Funções para geração da estrutura interna do bloco de função
   do módulo de chaveamento (Algoritmos 37, 42 e 43)              */
```

11 (TR_{lista},ALG_{lista}) ←*Geração dos Estados DO ECC do Bloco de Função do Módulo de Chaveamento (S_{CTF}, BLQ_{lista}, Dados para Chaveamento, variáveis de saída)*

12 *Geração das Transições do ECC do Bloco de Função do Módulo de Chaveamento (TR_{lista})*

13 escrever elemento final *ECC*

14 *Geração dos Algoritmos do Bloco de Função do Módulo de Chaveamento (ALG_{lista})*

15 escrever elemento final *BasicFB*

16 escrever elemento final *FBType*

Algoritmo 34: Geração da Interface do Bloco de Função do Módulo de Chaveamento

Entrada: (S_{CTF}, *Dados de Diagnose dos Componentes do Sistema*)

Saída: (*variáveis de saída*)

1 **início**

```
/* Função para Geração de Eventos na Interface (Algoritmo 35)    */
```

2 (*variáveis de entrada, variáveis de saída*) ← *Geração de Eventos na Interface (S_{CTF}, Dados de Diagnose dos Componentes do Sistema)*

```
/* Função para Geração de Variáveis na Interface (Algoritmo 36)
   */
```

3 *Geração de Variáveis na Interface (variáveis de entrada, variáveis de saída)*

Algoritmo 35: Geração de Eventos na Interface

Entrada: (S_{CTF}, *Dados de Diagnose dos Componentes do Sistema*)
Saída: (*variáveis de entrada*, *variáveis de saída*)

início
- escrever elemento inicial *EventInputs* para identificar os eventos de entrada
- *variáveis de entrada* $\leftarrow \emptyset$
- **para todo** *evento* $\in$ *{"INIT","SINAL_REC"}* **faça**
  - escrever elemento inicial *Event* acompanhado dos atributos *Comment* (comentário), *Name=evento* (evento de inicialização do bloco) e *Type* (tipo do evento) para definir a inicialização do bloco
  - **para todo** $(c, D_{lista}) \in$ *Dados de Diagnose dos Componentes do Sistema* **faça**
    - *componente* $\leftarrow$ *"COND_"* + $String(c)$
    - escrever elemento vazio *With* acompanhado do atributo *Var=componente*
    - *variáveis de entrada* $\leftarrow$ *variáveis de entrada* $\cup \{componente\}$
  - escrever elemento final *Event*
- **para todo** $(c, D_{lista}) \in$ *Dados de Diagnose dos Componentes do Sistema* **faça**
  - *componente* $\leftarrow$ *"COND_"* + $String(c)$
  - **para todo** $D \in D_{lista}$ **faça**
    - *diagnosticador* $\leftarrow$ *"SINAL_DIAG_"* $+ String(D)$
    - escrever elemento inicial *Event* acompanhado dos atributos *Comment* (comentário), *Name=diagnosticador* (evento de chaveamento proveniente do diagnosticador *D*) e *Type* (tipo do evento)
    - escrever elemento vazio *With* acompanhado do atributo *Var=componente*
    - escrever elemento final *Event*
- escrever elemento final *EventInputs*
- escrever elemento inicial *EventOutputs* para identificar os eventos de saída
- escrever elemento vazio *Event* acompanhado dos atributos *Comment* (comentário), *Name="INITO"* (evento de saída após a inicialização do bloco) e *Type* (tipo do evento)
- escrever elemento inicial *Event* acompanhado dos atributos *Comment* (comentário), *Name="COM_HAB"* (evento *COM_HAB*) e *Type* (tipo do evento) para comandar a habilitação ou a desabilitação dos supervisores e dos diagnosticadores do CTF
- criar vetor *variáveis de saída*
- *índice* $\leftarrow 0$
- **para todo** $S \in S_{CTF}$ **faça**
  - *supervisor* $\leftarrow$ *"ESTADO_"* $+ String(S)$
  - *índice* $\leftarrow$ *índice* $+1$
  - *variáveis de saída[índice]* $\leftarrow$ *supervisor*
  - escrever elemento vazio *With* acompanhado do atributo *Var=supervisor*
- escrever elemento final *Event*
- escrever elemento final *EventOutputs*

Algoritmo 36: Geração de Variáveis na Interface

Entrada: (*variáveis de entrada*, *variáveis de saída*)

1 **início**

2 escrever elemento inicial *InputVars* para identificar as variáveis de entrada

3 **para todo** *variável* $\in$ *variáveis de entrada* **faça**

4 escrever elemento vazio *VarDeclaration* acompanhado dos atributos *Comment* (comentário), *Name=variável* (condição de operação do componente) e *Type="BOOL"* (variável do tipo booleana)

5 escrever elemento final *InputVars*

6 escrever elemento inicial *OutputVars* para identificar as variáveis de saída

7 **para** $i \leftarrow 1$ **até** *o final do vetor variáveis de saída* **faça**

8 escrever elemento vazio *VarDeclaration* acompanhado dos atributos *Comment* (comentário), *Name=variável de saída[i]* (controle de operação do supervisor) e *Type="BOOL"* (variável do tipo booleana)

9 escrever elemento final *OutputVars*

Algoritmo 37: Geração dos Estados do Bloco de Função do Módulo de Chaveamento

Entrada: (S_{CTF}, BLQ_{lista}, *Dados para Chaveamento*, *variáveis de saída*)

Saída: (TR_{lista}, ALG_{lista})

1 **início**

`/* Função para geração dos estados iniciais e dos estados do sistema (Algoritmo 38) */`

2 $ALG_{lista} \leftarrow$ *Geração dos Estados Iniciais e dos Estados do Sistema* (S_{CTF}, BLQ_{lista}, *Dados para Chaveamento, variáveis de saída)*

`/* Lista de Transições` $TR_{lista} = \{(Origem, Destino, Transição), \ldots\}$ `*/`

3 $TR_{lista} \leftarrow$ {*("START", "INITO", "INIT")*, *("INITO", "UPDT", "1")*, *("UPDT", "UPDT", "SINAL_REC")*}

`/* Função para geração dos estados transitórios (Algoritmo 39) */`

4 (TR_{lista}, ALG_{lista}) $\leftarrow$*Geração dos Estados Transitórios* (S_{CTF}, BLQ_{lista}, *Dados para Chaveamento, variáveis de saída,* TR_{lista}, ALG_{lista})

Algoritmo 38: Geração dos Estados Iniciais e dos Estados do Sistema

Entrada: (S_{CTF}, BLQ_{lista}, *variáveis de saída*)
Saída: (ALG_{lista})

início
- escrever elemento inicial *ECState* acompanhado dos atributos *Comment* (comentário), *Name="START"* (estado inicial), *x* e *y* (coordenadas de posicionamento do estado)
- escrever elemento final *ECState*
- escrever elemento inicial *ECState* acompanhado dos atributos *Comment* (comentário), *Name="INITO"* (sinalização da inicialização do bloco), *x* e *y* (coordenadas de posicionamento do estado)
- escrever elemento vazio *ECAction* acompanhado do atributo *Output="INITO"* (evento para sinalização da inicialização)
- escrever elemento final *ECState*
- escrever elemento inicial *ECState* acompanhado dos atributos *Comment* (comentário), *Name="UPDT"* (estado de atualização das condições dos componentes), *x* e *y* (coordenadas de posicionamento do estado)
- escrever elemento final *ECState*
- **para todo** $S \in S_{CTF}$ **faça**
  - escrever elemento inicial *ECState* acompanhado dos atributos *Comment* (comentário), *Name=*$String(S)$ (supervisor *S* habilitado), *x* e *y* (coordenadas de posicionamento do estado)
  - escrever elemento vazio *ECAction* acompanhado do atributo *Output="COM_HAB"* (comando de habilitação)
  - escrever elemento final *ECState*

`/* Lista de Algoritmos` $ALG_{lista} = \{($ *`Nome do Algoritmo, Conteúdo`* $),\ldots\}$ `*/`

- *conteúdo* ← *" "*
- **para** *i* ← 1 **até** *o final do vetor variáveis de saída* **faça**
  - *conteúdo* ← *conteúdo + variáveis de saída[i] + ":= 0;"*
- $ALG_{lista} \leftarrow \{($*"ALG_DESAB", conteúdo*$)\}$
- **para todo** *estado de bloqueio* $\in BLQ_{lista}$ **faça**
  - *nome do estado* ← *String(estado de bloqueio)*
  - escrever elemento inicial *ECState* acompanhado dos atributos *Comment* (comentário), *Name=nome do estado*, *x* e *y* (coordenadas de posicionamento do estado)
  - escrever elemento vazio *ECAction* acompanhado dos atributos *Algorithm="ALG_DESAB"* (atualização dos estados dos supervisores e dos diagnosticadores) *Output="COM_HAB"* (comando de habilitação)
  - escrever elemento final *ECState*

Algoritmo 39: Geração dos Estados Transitórios

Entrada: (S_{CTF}, BLQ_{lista}, *Dados para Chaveamento*, *variáveis de saída*, TR_{lista}, ALG_{lista})
Saída: (TR_{lista}, ALG_{lista})

início
  índice ← 1
  para todo *(origem, destino, estados, sinal, vetor de componentes)* ∈ *Dados para Chaveamento* **faça**
    se *origem = UPDT* **então**
      estado transitório ← *String(origem)* + "_"+ *String(destino)*
      conteúdo ← " "
      para *i* ← 1 **até** *o final do vetor variáveis de saída* **faça**
        conteúdo ← *conteúdo* + *variáveis de saída[i]* + ":=" + *String(estados[i])* + ";"
      nome do algoritmo ← *"ALG_"* + *estado transitório*
      $ALG_{lista} \leftarrow ALG_{lista} \cup \{$(*nome do algoritmo, conteúdo*)$\}$
      escrever elemento inicial *ECState* acompanhado dos atributos *Comment* (comentário), *Name=estado transitório*, *x* e *y* (coordenadas de posicionamento do estado)
      escrever elemento vazio *ECAction* acompanhado do atributo *Algorithm=nome do algoritmo* (atualização dos estados dos supervisores e dos diagnosticadores)
      escrever elemento final *ECState*
      transição ← " "
      para *i* ← 1 **até** *o final do vetor de componentes* **faça**
        se *i* ≠ *tamanho do vetor de componentes* **então**
          transição ← *transição* + *"COND_"* + *String(vetor de componentes[i])* + *"= TRUE AND"*
        senão
          transição ← *transição* + *"COND_* + *String(vetor de componentes[i])* + *"= TRUE"*
      transição ← *"["* + *transição* + *"]"*
      $TR_{lista} \leftarrow TR_{lista} \cup \{$(*String(origem), estado transitório, transição*), (*estado transitório, String(destino), "1"*)$\}$
    `/* Função para análise de chaveamentos oriundos de supervisores (Algoritmo 40) */`
    TR_{lista}, ALG_{lista}, *índice* ← *Análise de Chaveamentos Oriundos de Supervisores* (S_{CTF}, BLQ_{lista}, *origem, destino, estados, sinal, variáveis de saída,* TR_{lista}, ALG_{lista}, *índice*)
    `/* Função para análise de chaveamentos oriundos de bloqueios (Algoritmo 41) */`
    TR_{lista}, ALG_{lista}, *índice* ← *Análise de Chaveamentos Oriundos de Bloqueios* (BLQ_{lista}, *origem, destino, estados, sinal, vetor de componentes, variáveis de saída,* TR_{lista}, ALG_{lista}, *índice*)

Algoritmo 40: Análise de Chaveamentos Oriundos de Supervisores

Entrada: (S_{CTF}, BLQ_{lista}, *origem, destino, estados, sinal*, *variáveis de saída*, TR_{lista}, ALG_{lista}, *índice*)
Saída: (TR_{lista}, ALG_{lista}, *índice*)

```
1  início
2      se origem ∈ S_CTF então
3          se destino ∈ S_CTF então
4              estado transitório ← String(origem) + "_" + String(destino) + "_" +
                 String(índice)
5              índice ← índice + 1
6              conteúdo ← "  "
7              para i ← 1 até o final do vetor variáveis de saída faça
8                  conteúdo ← conteúdo + variáveis de saída[i] + ":=" + String(estados[i])
                     + ";"
9              nome do algoritmo ← "ALG_" + estado transitório
10             ALG_lista ← ALG_lista ∪ {(nome do algoritmo, conteúdo)}
11             escrever elemento inicial ECState acompanhado dos atributos Comment
                 (comentário), Name=estado transitório, x e y (coordenadas de
                 posicionamento do estado)
12             escrever elemento vazio ECAction acompanhado do atributo
                 Algorithm=nome do algoritmo (atualização dos estados dos supervisores e
                 dos diagnosticadores)
13             escrever elemento final ECState
14             transição ← "SINAL_" + String(sinal)
15             TR_lista ← TR_lista ∪ {(String(origem), estado transitório, transição), (estado
                 transitório, String(destino), "1")}
16         senão se destino ∈ BLQ_lista então
17             transição ← "SINAL_" + String(sinal)
18             TR_lista ← TR_lista ∪ {(String(origem), String(destino), transição)}
19         senão
20             estado transitório ← String(origem) + "_" + String(destino)
21             escrever elemento inicial ECState acompanhado dos atributos Comment
                 (comentário), Name=estado transitório, x e y (coordenadas de
                 posicionamento do estado)
22             escrever elemento vazio ECAction acompanhado dos atributos
                 Algorithm="ALG_DESAB" (atualização dos estados dos supervisores e dos
                 diagnosticadores) Output="COM_HAB" (comando de habilitação)
23             escrever elemento final ECState
24             transição ← "SINAL_" + String(sinal)
25             TR_lista ← TR_lista ∪ {(String(origem), estado transitório, transição), (estado
                 transitório, String(destino), "1")}
```

Algoritmo 41: Análise de Chaveamentos Oriundos de Bloqueios

Entrada: (BLQ_{lista}, *origem, destino, estados, sinal, vetor de componentes*, *variáveis de saída*, TR_{lista}, ALG_{lista}, *índice*)
Saída: (TR_{lista}, ALG_{lista}, *índice*)

1 **início**
2 **se** *origem* $\in BLQ_{lista}$ **então**
3 *estado transitório* $\leftarrow$ *String(origem)* + "_" + *String(destino)* + "_" + *String(índice)*
4 *índice* $\leftarrow$ *índice* $+1$
5 *conteúdo* $\leftarrow$ " "
6 **para** $i \leftarrow 1$ **até** *o final do vetor variáveis de saída* **faça**
7 *conteúdo* $\leftarrow$ *conteúdo* + *String(variáveis de saída[i])* + ":=" + *String(estados[i])* + ";"
8 *nome do algoritmo* $\leftarrow$ "ALG_" + *estado transitório*
9 $ALG_{lista} \leftarrow ALG_{lista} \cup \{$(*nome do algoritmo, conteúdo*)$\}$
10 escrever elemento inicial *ECState* acompanhado dos atributos *Comment* (comentário), *Name=estado transitório*, *x* e *y* (coordenadas de posicionamento do estado)
11 escrever elemento vazio *ECAction* acompanhado do atributo *Algorithm=nome do algoritmo* (atualização dos estados dos supervisores e dos diagnosticadores)
12 escrever elemento final *ECState*
13 *transição* $\leftarrow$ "SINAL_" + *String(sinal)* + "[COND_" + *String(vetor de componentes)* + "= TRUE]"
14 $TR_{lista} \leftarrow TR_{lista} \cup \{$(*String(origem), estado transitório, transição*), (*estado transitório, String(destino), "1"*)$\}$

Algoritmo 42: Geração das Transições do Bloco de Função do Módulo de Chaveamento

Entrada: (TR_{lista})

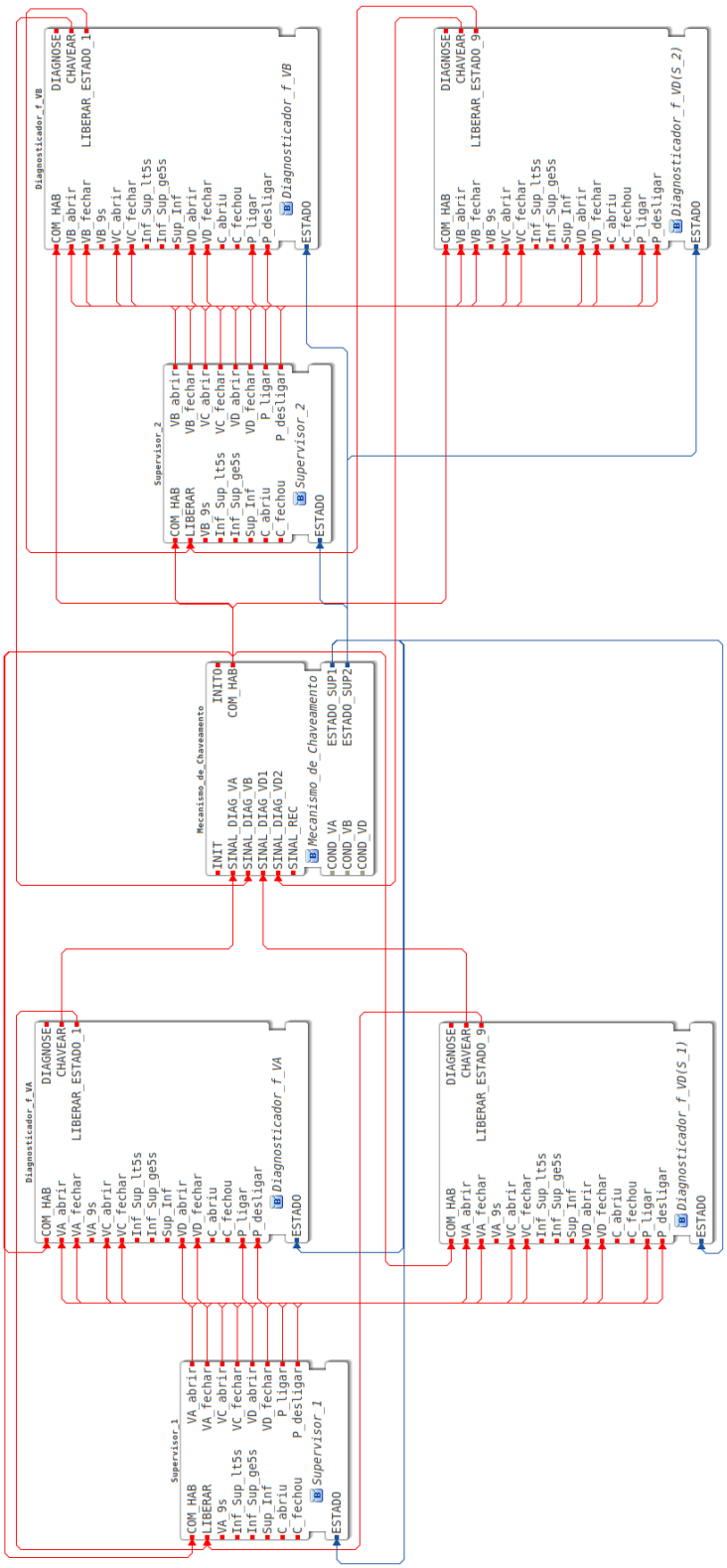
1 **início**
2 **para todo** *(estado de origem, estado de destino, transição)* $\in TR_{lista}$ **faça**
3 escrever elemento vazio *ECTransition* acompanhado dos atributos *Source=estado de origem*, *Destination=estado de destino*, *Condition=transição* (condição de transição), *Comment* (comentário), *x* e *y* (coordenadas de posicionamento)

Algoritmo 43: Geração dos Algoritmos do Bloco de Função do Módulo de Chaveamento

Entrada: (ALG_{lista})

1 **início**
2 **para todo** *(nome do algoritmo, conteúdo)* $\in ALG_{lista}$ **faça**
3 escrever elemento inicial *Algorithm* acompanhado dos atributos *Name=nome do algoritmo* e *Comment* (comentário)
4 *declaração do conteúdo* $\leftarrow$ *"<![CDATA[" + conteúdo + "]]>"*
5 escrever elemento inicial *ST* (algoritmo de linguagem de texto estruturado)
6 escrever *declaração do conteúdo*
7 escrever elemento final *ST*
8 escrever elemento final *Algorithm*

Figura 48 – Interligação dos blocos do CTF no projeto de controle do processo de mistura da Figura 22.



Fonte: Elaborada pelo autor (2022).

4.3 AVALIAÇÃO E VALIDAÇÃO DA METODOLOGIA DE CONTROLE TOLERANTE A FALHAS

Os algoritmos apresentados na Seção 4.2, que representam o sistema computacional originado da metodologia de CTF proposta na tese, foram implementados e testados em linguagem de programação Python. Uma cópia desses algoritmos se encontra disponível no repositório do GASR (UDESC, 2021). Para o teste, foi utilizado um modelo computacional do processo de mistura adotado como estudo de caso e apresentado na Seção 4.1, com as mesmas políticas de controle (supervisores), critério de mudança de controle e falhas nos componentes.

O sistema computacional foi modularizado em três subsistemas, conforme é proposto na metodologia de CTF: sistema de análise da diagnosticabilidade de falhas, sistema de síntese de controle tolerante a falhas e sistema de síntese do controle tolerante a falhas em blocos de função. A validação dos subsistemas foi feita por meio da avaliação das saídas, que foram obtidas para esses subsistemas, em função das entradas apresentadas, retratando o que foi apresentado nos Exemplos 1–5 na Seção 4.2. No final, obteve-se os blocos de função apresentados na Figura 48, que formam o CTF para o estudo de caso da Seção 4.1.

Para realizar uma avaliação prática do funcionamento do CTF obtido, foi desenvolvido um modelo computacional para o processo de mistura também por meio de uma rede de blocos de função da IEC 61499, gerando uma aplicação que simula a sua operação (Figura 49). Nessa aplicação, as válvulas e a bomba de sucção possuem blocos de função projetados especificamente para representar os seus funcionamentos: válvula *A* (bloco *VA*), válvula *B* (bloco *VB*), válvula *C* (bloco *VC*), válvula *D* (bloco *VD*) e bomba de sucção (bloco *PUMP*). Nestes atuadores, os eventos controláveis existentes no modelo do processo de mistura da Figura 22 representam eventos de *SET* e *RESET* nas entradas dos blocos, por exemplo, os eventos de *SET* e *RESET* na entrada do bloco de função da válvula *C*, são, respectivamente, os eventos controláveis *VC_abrir* e *VC_fechar*. Já para a representação dos elementos de sensoreamento existentes no processo (chave fim-de-curso e sensor de medição da concentração da mistura), foram utilizados um conjunto de blocos para cada elemento. Para a representação da chave fim-de-curso, foram utilizados os blocos de função *SW_VD*, *C_ABRIU* e *C_FECHOU*, sendo os dois últimos blocos responsáveis pela geração dos eventos não controláveis *C_abriu* e *C_fechou*, existentes no modelo do processo de mistura. A representação do sensor de medição da concentração da mistura foi feita pelos blocos *Sensor*, responsável pelo cálculo da concentração da mistura no processo, e *Sinal_Sensor*, responsável pela geração dos eventos não controláveis *Inf_Sup_ge5s*, *Inf_Sup_lt5s* e *Sup_Inf*.

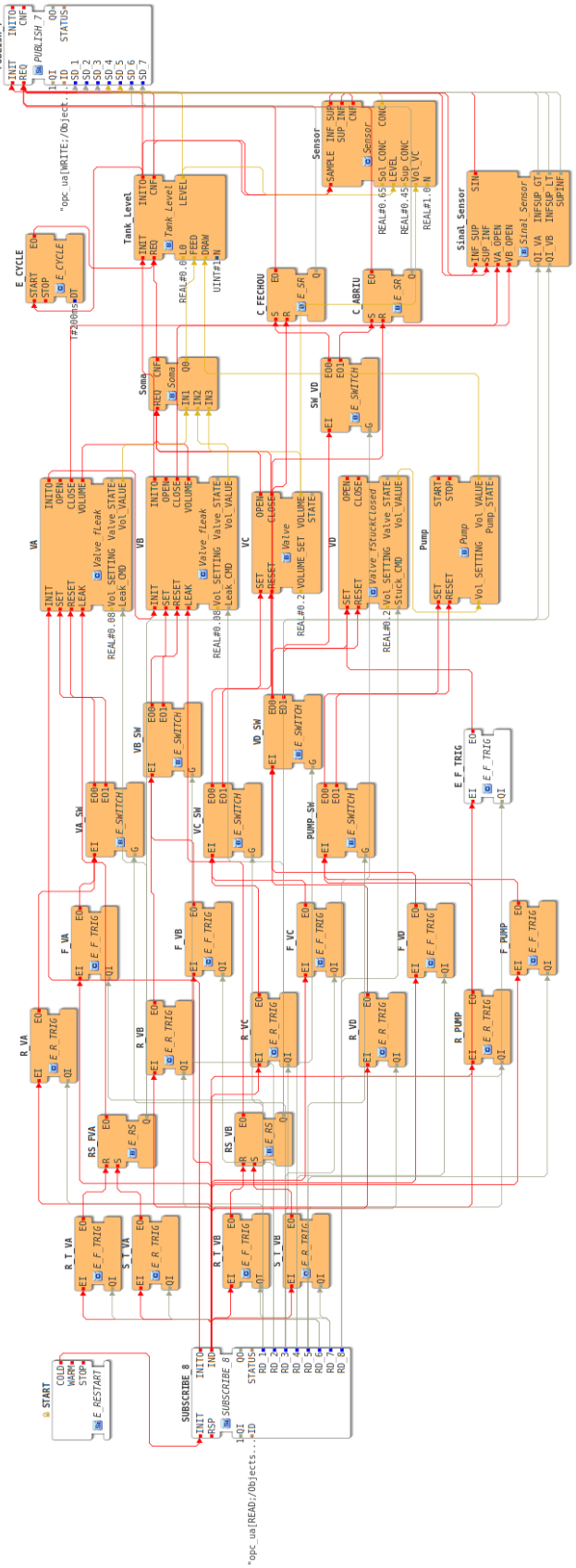
Os blocos de função, obtidos pelo Sistema de Síntese do Controle Tolerante a Falhas em Blocos de Função, foram interligados em uma aplicação na IEC 61499 desenvolvida para a operação do CTF (Figura 50). O fato do modelo computacional do processo de mistura também ter sido desenvolvido em uma aplicação na IEC 61499, propiciou com isso um ambiente unificado para a realização do teste e validação do CTF obtido. Tanto a aplicação do projeto do CTF como

a aplicação do modelo computacional do processo de mistura foram feitas na *Framework* 4diac e executadas em máquinas virtuais do tipo *fortePC* (*softPLC* disponibilizado no 4diac). Uma visão da vista de configuração do sistema é mostrada na Figura 51. As duas máquinas virtuais foram habilitadas para operar no mesmo computador, tendo como portas de gerenciamento as portas TCP 61510 (máquina virtual com o processo de mistura) e 61520 (ctf do processo de mistura).

A comunicação entre as máquinas virtuais foi feita por meio do protocolo OPC UA, baseando-se no modelo de configuração apresentado na documentação do 4diac (4DIAC, 2022). Nos testes, foi utilizado um cliente OPC *UaExpert*, tanto para monitorar as atualizações de eventos no sistema, quando o mesmo se encontra em malha fechada de controle, quanto para gerar as falhas nas válvulas *A*, *B* e *D* e sinalizar suas correções (Figura 52). A observação da ocorrência de eventos no sistema é feita por meio das variáveis inseridas na aba *Data Access View*. A representação dos eventos controláveis no OPC é feita por meio das variáveis que representam os atuadores válvula *A* (variável *VA*), válvula *B* (variável *VB*), válvula *C* (variável *VC*), válvula *D* (variável *VD*) e bomba de sucção (variável *P*), por exemplo, os eventos *VC_abrir* e *VC_fechar* são representados, respectivamente, pelos estados *True* e *False* da variável *VA*. A representação dos eventos não controláveis no OPC é feita diretamente por variáveis que representam estes eventos, por exemplo, o evento *Sup_Inf* é representado pela variável de mesmo nome (variável *SUP_INF*). As variáveis *Level* e *Concentration* servem, respectivamente, para monitorar o nível e a concentração da mistura no processo. Já as variáveis *F_VA*, *F_VB* e *F_VD* possuem dupla função nos testes, pois tanto servem para gerar, respectivamente, as falhas na válvulas *A*, *B* e *D*, como servem para sinalizar a recuperação destes componentes por meio de uma manutenção corretiva. Portanto, por exemplo, a variável *F_VA* quando se encontra na condição *True*, indica a existência da falha de vazamento na válvula *A* (f_{VA}), e quando se encontra no estado *False*, indica que não existe falha na válvula *A*.

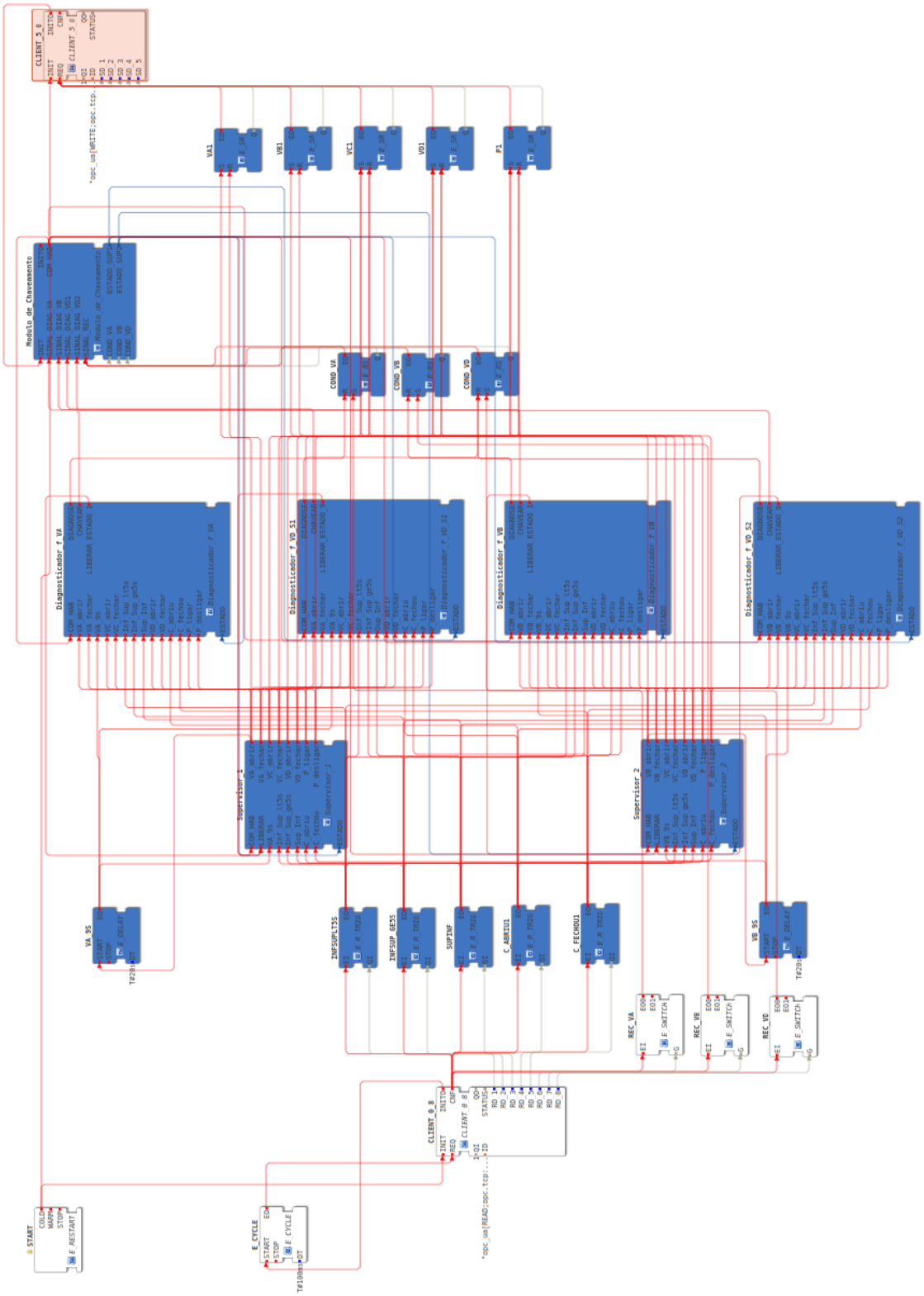
No teste, foram feitas avaliações da ocorrência das falhas nas diferentes condições de operação do sistema, podendo-se observar chaveamentos entre políticas de controle no CTF e até mesmo de parada do sistema, quando não era possível chavear para outra política de controle. Os resultados se mostraram efetivos, com o CTF conseguindo exercer sua ação sobre o processo de mistura e mantendo a operação do processo conforme era desejado. Um vídeo do teste se encontra disponível no canal do GASR no *YouTube* (UDESC, 2022).

Figura 49 – Rede de blocos de função da IEC 61499 que simula o funcionamento do processo de mistura da Figura 22.



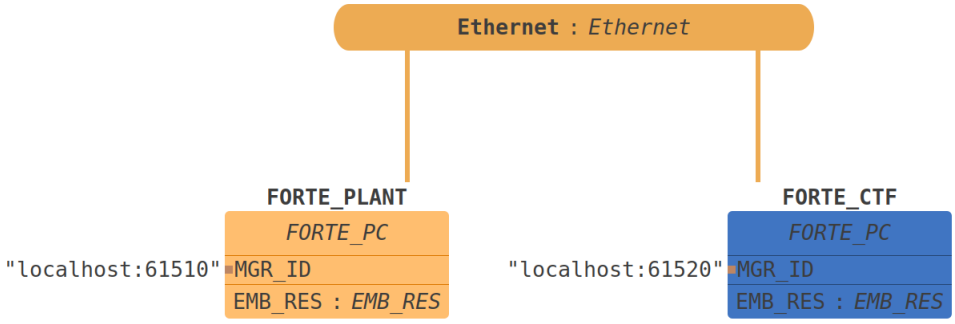
Fonte: Elaborada pelo autor (2022).

Figura 50 – Rede de blocos de função da IEC 61499 do projeto do controle tolerante a falhas para o processo de mistura da Figura 22.



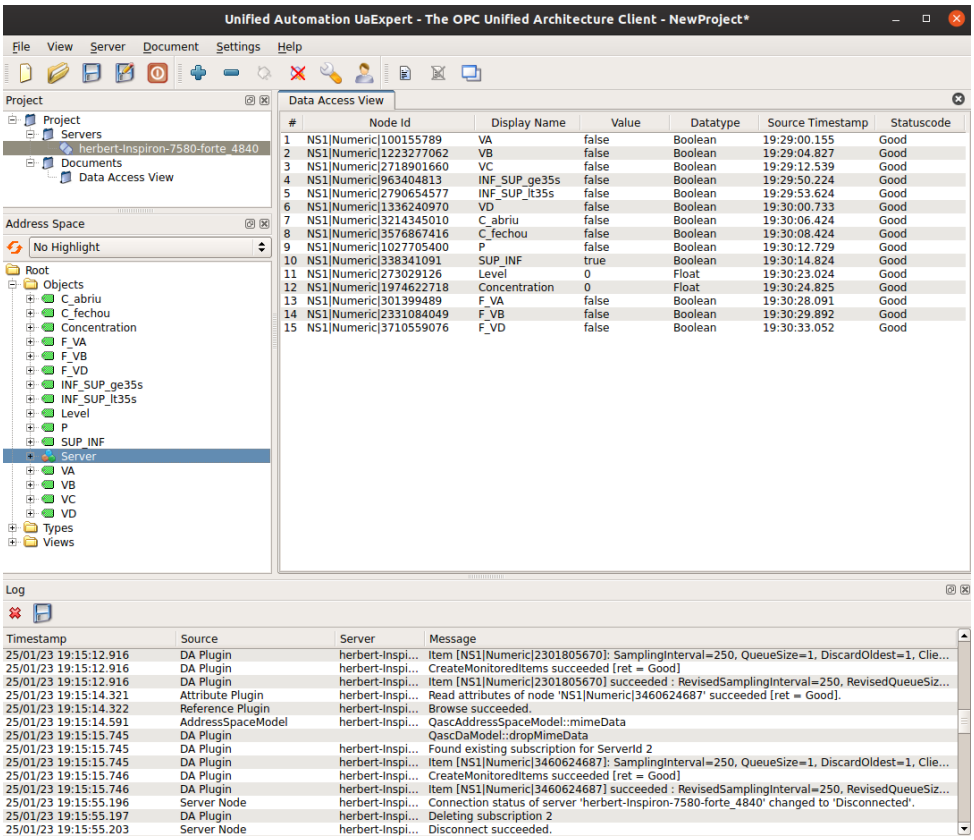
Fonte: Elaborada pelo autor (2022).

Figura 51 – Vista de configuração com as máquinas virtuais do tipo *fortePC* utilizadas para testar a ação do CTF no processo de mistura da Figura 22.



Fonte: Elaborada pelo autor (2022).

Figura 52 – Tela do cliente OPC *UaExpert* com monitoramento de eventos no sistema por meio da aba *Data Access View*.



Fonte: Elaborada pelo autor (2022).

4.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo foram abordados os temas diagnose falhas e controle tolerante a falhas em SEDs, apresentando as contribuições provenientes desta tese para estes assuntos.

A Seção 4.1 apresentou um estudo de caso baseado em um processo de mistura, que serviu para a avaliação da metodologia de CTF apresentada na Seção 4.2. Falhas de diferentes naturezas e com diferentes consequências para a operação do sistema foram consideradas em componentes do processo de mistura, fazendo com que diferentes ações de tolerância a estas falhas fossem utilizadas no CTF.

Na Seção 4.2, foi apresentada a principal contribuição da tese: uma metodologia de projeto de CTF para a arquitetura de reconfiguração RAFAH (LEITÃO et al., 2020). A análise e a descrição da metodologia de projeto do CTF foram realizadas levando em consideração suas três fases de desenvolvimento: análise de diagnosticabilidade de falhas, síntese do modelo do CTF e síntese do CTF em blocos de função.

Na Seção 4.3, foi relatado o teste realizado com o intuito de avaliar e validar a metodologia de CTF proposta na tese. No teste, foram desenvolvidas aplicações, na *framework* 4diac, para os blocos de função da IEC 61499, obtidos na síntese do CTF do processo de mistura, e para um modelo do processo de mistura. As aplicações foram executadas em máquinas do tipo *fortePC* (*softPLC* disponibilizado no 4diac) e, com isso, foram feitas a avaliação e a validação do CTF.

5 CONCLUSÃO E TRABALHOS FUTUROS

Nesta tese foi apresentado o desenvolvimento de uma metodologia de projeto de controle tolerante a falhas direcionada a aplicações em sistemas industriais. Baseando-se no uso de métodos formais em sistemas a eventos discretos, a metodologia sistematizou em um conjunto de atividades: a análise da diagnosticabilidade de falhas nos sistemas, a construção de modelos de controles tolerante a estas falhas e a síntese destes modelos em lógica de controle.

De acordo com o que foi observado na literatura, as arquiteturas de CTF propostas não atendiam ou atendiam parcialmente: à possibilidade de atuação em um cenário com mais de uma falha, inclusive permitindo trocas entre políticas de controle de um comportamento degradado para outro; à condição de recuperação da falha no sistema; à condição de controlabilidade segura do sistema por meio da diagnose da falha; à condição de ter seu desenvolvimento executado de forma não assistida; e à possibilidade de implementação em lógica de controle. Além disso, também foi incorporada ao CTF a possibilidade de adotar estratégias de controle diferentes no início da operação do sistema. Este conjunto de requisitos tem importância no desenvolvimento do CTF, pois, além de simplificar sua aplicação, torna-o mais adaptável e flexível. Sendo assim, a metodologia de projeto de CTF apresentada nesta tese visou preencher esta lacuna.

A partir do que foi estruturado na metodologia de projeto de CTF apresentada na tese, foi proposto o desenvolvimento de um sistema computacional, que é parte integrante de um sistema de reconfiguração para tratamento de falhas denominado de RAFAH (*Reconfiguration Architecture for Fault Handling*, arquitetura de reconfiguração para tratamento de falhas). Na tese, foram desenvolvidas três etapas da arquitetura: a etapa de análise da diagnosticabilidade de falhas, composta pelo sistema de análise da diagnosticabilidade de falhas (FAS, *Fault Analysis System*); a etapa de modelagem do controle tolerante a falhas, composta pelo sistema de síntese de controle tolerante a falhas (FTCSS, *Fault Tolerant Control Synthesis System*); e a etapa de síntese de lógica de controle, responsável pela implementação do modelo do controle tolerante a falhas em blocos de função. O fato da arquitetura de reconfiguração ter uma estrutura formada por etapas distintas, sendo estas etapas compostas por diferentes ações que sequencialmente contribuem para o desenvolvimento do CTF, permite que diferentes estratégias, seja de análise de diagnosticabilidade de falhas, de modelagem de CTF ou de síntese de lógica de controle, possam ser exploradas futuramente na arquitetura.

O sistema computacional, que derivou da metodologia de projeto de CTF, foi implementado em linguagem de programação Python, sendo os algoritmos que o representam apresentados no Capítulo 4. Seu desenvolvimento foi feito de forma modularizada, tendo como composição conjuntos de códigos para cada etapa da RAFAH. Especificamente para a etapa de síntese em lógica de controle, existiu internamente uma nova modularização em sua implementação, tendo cada elemento do CTF (supervisor, diagnosticador e módulo de chaveamento) uma metodologia específica de implementação em bloco de função. Este processo de modularização faz com que as etapas da RAFAH operem de forma independente, possibilitando que possam ser utilizadas de

forma individualizada ou incorporadas a outros projetos.

Um aspecto importante presente na síntese do modelo do CTF em blocos de função da IEC 61499 é que cada um de seus elementos é implementado de forma autocontida em um bloco de função. Isto facilita a identificação desses componentes, o monitoramento de suas ações na aplicação, a análise gráfica do CTF e o seu uso em sistemas distribuídos. Outro aspecto relevante da síntese é que os supervisores e os diagnosticadores têm representações nos gráficos de controle de execução dos blocos de função muito parecidas aos seus modelos em autômatos, facilitando suas interpretações no projeto implementado.

Na tese, a metodologia de controle tolerante a falhas foi testada e validada por meio da utilização de um estudo de caso baseado em um processo de mistura que representa a dissolução de um produto químico genérico em água. Para o teste, foi definido como critério de mudança de controle que o sistema operasse até o último estado onde fosse possível exercer a ação de controlabilidade segura. O teste foi executado em uma aplicação desenvolvida na *Framework* 4diac, sendo observadas a ação do sistema perante falhas e a consequente reação do CTF. Seus resultados se mostraram efetivos, indicando que a aplicação da RAFAH em sistemas industriais é promissora.

Como trabalhos futuros, existem diversas possibilidades a serem exploradas em relação ao que foi desenvolvido na tese. Uma delas é o desenvolvimento do restante da RAFAH, que depende da finalização da etapa de síntese de lógica de controle, produzindo uma aplicação completa na IEC 61499, e do desenvolvimento da etapa de reconfiguração dinâmica. Está sendo feita a portabilidade dos algoritmos que foram produzidos na tese para o *software* Nadzoru 2 (GASR-UDESC, 2020), com isso as etapas da arquitetura passarão a ter integração direta com outras ferramentas computacionais da área de SEDs, terão uma melhor usabilidade e poderão ser exploradas de forma mais efetiva em outros trabalhos. Como as etapas de desenvolvimento do CTF encontram-se modularizadas, existe a possibilidade de padronizar em arquivos com linguagem de marcação do tipo *Extensible Markup Language* (XML) suas entradas e saídas. Isto já é feito na saída da etapa de síntese do modelo do CTF em blocos de função, já que os blocos de função são arquivos deste tipo. Na tese, foi utilizada uma arquitetura de CTF baseada em supervisores monolíticos, entretanto, existe uma tendência para que os sistemas industriais adotem controles que possuam arquiteturas distribuídas. Sendo assim, é de grande relevância que a arquitetura de reconfiguração possua outras metodologias de desenvolvimento de CTF que futuramente adotem outras técnicas de controle, como, por exemplo, o controle supervisório modular local (QUEIROZ, 2004) e o controle por localização do supervisor (CAI; WONHAM, 2010). Sobre o ponto de vista do projeto do CTF, a arquitetura de reconfiguração considera apenas a diagnose da falha no modelo. Pretende-se futuramente expandir sua ação incorporando ao modelo informações referentes à prognose das falhas (WATANABE et al., 2021).

REFERÊNCIAS

4DIAC. *Open Source PLC Framework for Industrial Automation & Control*. 2022. Disponível em: <www.eclipse.org/4diac/>. Citado 2 vezes nas páginas 41 e 127.

ACAR, Ayse Nur; SCHMIDT, Klaus Werner. Discrete event supervisor design and application for manufacturing systems with arbitrary faults and repairs. In: IEEE. **2015 IEEE International Conference on Automation Science and Engineering (CASE)**. [S.l.], 2015. p. 825–830. Citado na página 44.

ANGLUIN, Dana. Learning regular sets from queries and counterexamples. **Information and computation**, Elsevier, v. 75, n. 2, p. 87–106, 1987. Citado na página 50.

BASILE, Francesco; CHIACCHIO, Pasquale; GERBASIO, Diego. On the implementation of industrial automation systems based on plc. **IEEE Transactions on Automation Science and Engineering**, IEEE, v. 10, n. 4, p. 990–1003, 2012. Citado na página 21.

BLANKE, Mogens et al. **Diagnosis and fault-tolerant control**. [S.l.]: Springer, 2016. v. 2. Citado na página 19.

CAI, Kai; WONHAM, W Murray. Supervisor localization: a top-down approach to distributed control of discrete-event systems. **IEEE Transactions on Automatic Control**, IEEE, v. 55, n. 3, p. 605–618, 2010. Citado na página 133.

CARVALHO, Lilian K; BASILIO, João C; MOREIRA, Marcos V. Robust diagnosis of discrete event systems against intermittent loss of observations. **Automatica**, Elsevier, v. 48, n. 9, p. 2068–2078, 2012. Citado 2 vezes nas páginas 31 e 33.

CASSANDRAS, Christos G; LAFORTUNE, Stephane. **Introduction to discrete event systems**. [S.l.]: Springer Science & Business Media, 2009. Citado 5 vezes nas páginas 23, 25, 26, 27 e 28.

CENGIC, Goran; AKESSON, Knut. On formal analysis of IEC 61499 applications, part a: Modeling. **IEEE Transactions on Industrial Informatics**, IEEE, v. 6, n. 2, p. 136–144, 2010. Citado na página 41.

CENGIC, Goran et al. Implementation of full synchronous composition using IEC 61499 function blocks. In: IEEE. **IEEE International Conference on Automation Science and Engineering, 2005.** Edmonton, AB, Canadá, 2005. p. 267–272. Citado na página 21.

DAI, Jin; KARIMODDINI, Ali; LIN, Hai. Achieving fault-tolerance and safety of discrete-event systems through learning. In: IEEE. **2016 American Control Conference (ACC)**. [S.l.], 2016. p. 4835–4840. Citado 5 vezes nas páginas 9, 45, 50, 52 e 57.

DARABI, Houshang; JAFARI, Mohsen A; BUCZAK, Anna L. A control switching theory for supervisory control of discrete event systems. **IEEE Transactions on Robotics and Automation**, IEEE, v. 19, n. 1, p. 131–137, 2003. Citado na página 44.

DERBEL, Haithem et al. Diagnosis of a class of timed discrete event systems. In: IEEE. **2006 8th International Workshop on Discrete Event Systems**. Ann Arbor, MI, Estados Unidos, 2006. p. 256–261. Citado na página 58.

DUBININ, Victor et al. Implementation of state transition models in IEC 61499 and its use for recognition and selection of sequences of events and objects. In: IEEE. **2019 IEEE 17th International Conference on Industrial Informatics (INDIN)**. Helsinki, Finland, 2019. v. 1, p. 466–469. Citado na página 21.

FABIAN, Martin; HELLGREN, Anders. PLC-based implementation of supervisory control for discrete event systems. In: IEEE. **Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No. 98CH36171)**. Tampa, FL, Estados Unidos, 1998. v. 3, p. 3305–3310. Citado 2 vezes nas páginas 86 e 88.

FOYTH, Jonas O. **Metodologia de implementação de código aderente à norma IEC 61499 baseada na Teoria de Controle Supervisório de Sistemas a Eventos Discretos.** 2018. Monografia (Bacharel em Engenharia Elétrica), UDESC (Universidade do Estado de Santa Catarina), Joinville, Brazil. Citado na página 21.

FRITZ, Raphael; ZHANG, Ping. Overview of fault-tolerant control methods for discrete event systems. **IFAC-PapersOnLine**, Elsevier, v. 51, n. 24, p. 88–95, 2018. Citado 3 vezes nas páginas 20, 43 e 44.

GASR-UDESC. **Nadzoru 2**. 2020. Disponível em: <www.udesc.br/cct/gasr/softwares/nadzoru>. Citado na página 133.

GATWAZA, Fabrice et al. Fault-tolerant control of energy production devices using discrete event systems formalisms. In: IEEE. **2020 International Conference on Control, Automation and Diagnosis (ICCAD)**. Paris, France, 2020. p. 1–6. Citado 4 vezes nas páginas 9, 53, 54 e 57.

HOLOBLOC. **Function Block Development Kit (FBDK)**. 2008. Disponível em: <www.holobloc.org>. Citado na página 41.

IEC. **61499-1: Function Blocks – Part 1: Architecture**. Genebra, Suíça, 2012. Citado 2 vezes nas páginas 19 e 36.

IEC. **61499-2: Function Blocks – Part 2: Software tool requirements**. Genebra, Suíça, 2012. Citado 3 vezes nas páginas 19, 36 e 37.

IEC. **61499-4: Function blocks – Part 4: Rules for compliance profiles**. Genebra, Suíça, 2013. Citado 2 vezes nas páginas 19 e 36.

IEC. **IEC 61131–3: 2013**. Genebra, Suíça, 2013. Citado 2 vezes nas páginas 19 e 40.

KARIMADINI, Mohammad; KARIMODDINI, Ali; HOMAIFAR, Abdollah. A survey on fault-tolerant supervisory control. In: IEEE. **2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)**. Windsor, ON, Canadá, 2018. p. 733–738. Citado na página 43.

KUMAR, Ratnesh; TAKAI, Shigemasa. A framework for control-reconfiguration following fault-detection in discrete event systems. **IFAC Proceedings Volumes**, Elsevier, v. 45, n. 20, p. 848–853, 2012. Citado na página 45.

LEITÃO, Herbert A S et al. Fault handling in discrete event systems applied to IEC 61499. In: IEEE. **2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)**. Viena, Austria, 2020. v. 1, p. 1039–1042. Citado 4 vezes nas páginas 20, 58, 64 e 131.

LIN, Feng. Robust and adaptive supervisory control of discrete event systems. **IEEE Transactions on Automatic Control**, IEEE, v. 38, n. 12, p. 1848–1852, 1993. Citado 2 vezes nas páginas 20 e 43.

MAJDZIK, Paweł et al. A fault-tolerant approach to the control of a battery assembly system. **Control Engineering Practice**, Elsevier, v. 55, p. 139–148, 2016. Citado na página 45.

MAJDZIK, Paweł; WITCZAK, Marcin; LIPIEC, Bogdan. Feasible schedule-based predictive fault-tolerant control for an automated packing system. In: IEEE. **2019 4th Conference on Control and Fault Tolerant Systems (SysTol)**. Casablanca, Marrocos, 2019. p. 165–170. Citado na página 45.

MENDONÇA, Rodrigo. **DES4DC: Uma metodologia para a implementação da estrutura de controle supervisório modular local usando blocos de função da IEC 61499.** 2019. Monografia (Bacharel em Engenharia Elétrica), UDESC (Universidade do Estado de Santa Catarina), Joinville, Brazil. Citado na página 21.

MOOR, Thomas. Fault-tolerant supervisory control. **IFAC-PapersOnLine**, Elsevier, v. 48, n. 7, p. 124–131, 2015. Citado na página 29.

MOOR, Thomas. A discussion of fault-tolerant supervisory control in terms of formal languages. **Annual Reviews in Control**, Elsevier, v. 41, p. 159–169, 2016. Citado 3 vezes nas páginas 43, 44 e 45.

MOOR, Thomas; SCHMIDT, Klaus Werner. Fault-tolerant control of discrete-event systems with lower-bound specifications. **IFAC-PapersOnLine**, Elsevier, v. 48, n. 7, p. 161–166, 2015. Citado 3 vezes nas páginas 44, 48 e 57.

MOREIRA, Benjamin Grando; LEAL, André Bittencourt. A proposal for an active diagnoser for safe fault-tolerant control of discrete event systems. **IFAC-PapersOnLine**, Elsevier, v. 53, n. 4, p. 282–287, 2020. Citado na página 45.

NIGUEZ, Julien; AMARI, Saïd; FAURE, Jean-Marc. Active fault-tolerant control of timed automata with guards. **IFAC-PapersOnLine**, Elsevier, v. 50, n. 1, p. 13648–13653, 2017. Citado 3 vezes nas páginas 45, 53 e 57.

NKE, Yannick; LUNZE, Jan. Fault-tolerant control of nondeterministic input/output automata subject to actuator faults. **IFAC Proceedings Volumes**, Elsevier, v. 43, n. 12, p. 350–355, 2010. Citado na página 45.

NKE, Yannick; LUNZE, Jan. A fault modeling approach for input/output automata. **IFAC Proceedings Volumes**, Elsevier, v. 44, n. 1, p. 8657–8662, 2011. Citado na página 45.

NKE, Yannick; LUNZE, Jan. Control reconfiguration based on unfolding of input/output automata. **IFAC Proceedings Volumes**, Elsevier, v. 45, n. 20, p. 866–873, 2012. Citado na página 45.

PANG, Cheng et al. A portability study of IEC 61499: Semantics and tools. In: IEEE. **2014 12th IEEE International Conference on Industrial Informatics (INDIN)**. Porto Alegre, Brazil, 2014. p. 440–445. Citado na página 36.

PANG, Cheng; VYATKIN, Valeriy. Automatic model generation of IEC 61499 function block using net condition/event systems. In: IEEE. **2008 6th IEEE International Conference on Industrial Informatics**. Daejeon, Coréia do Sul, 2008. p. 1133–1138. Citado na página 21.

PAOLI, Andrea; LAFORTUNE, StéPhane. Safe diagnosability for fault-tolerant supervision of discrete-event systems. **Automatica**, Elsevier, v. 41, n. 8, p. 1335–1347, 2005. Citado 2 vezes nas páginas 34 e 35.

PAOLI, Andrea; SARTINI, Matteo; LAFORTUNE, Stéphane. A fault tolerant architecture for supervisory control of discrete event systems. **IFAC Proceedings Volumes**, Elsevier, v. 41, n. 2, p. 6542–6547, 2008. Citado 2 vezes nas páginas 21 e 44.

PAOLI, Andrea; SARTINI, Matteo; LAFORTUNE, Stéphane. Active fault tolerant control of discrete event systems using online diagnostics. **Automatica**, Elsevier, v. 47, n. 4, p. 639–649, 2011. Citado 12 vezes nas páginas 9, 19, 21, 22, 29, 43, 44, 46, 47, 57, 65 e 66.

PINTO, Leandro Israel. **Um Método de Reconfiguração Dinâmica Segura e sua Aplicação em Sistemas Aderentes à IEC 61499**. Tese (Doutorado) — Universidade do Estado de Santa Catarina, 2019. Citado na página 39.

PINTO, Leandro I; LEAL, André B; ROSSO, Roberto SU. Safe dynamic reconfiguration through supervisory control in iec 61499 compliant systems. In: IEEE. **2017 IEEE 15th International Conference on Industrial Informatics (INDIN)**. Emden, Alemanha, 2017. p. 753–758. Citado na página 21.

PINTO, Leandro Israel et al. Icaru-fb: An iec 61499 compliant multiplatform software infrastructure. **IEEE Transactions on Industrial Informatics**, IEEE, v. 12, n. 3, p. 1074–1083, 2016. Citado na página 42.

PRADO, Rafael B. S. **Desenvolvimento e implementação de algoritmos para a conversão automática da estrutura de controle supervisório de SEDs em blocos de função aderentes à IEC 61499 seguindo a metodologia DES4DC.** 2019. Monografia (Bacharel em Engenharia Elétrica), UDESC (Universidade do Estado de Santa Catarina), Joinville, Brazil. Citado na página 21.

QUEIROZ, Max Hering de. **Controle supervisório modular e multitarefa de sistemas compostos**. Tese (Doutorado) — Universidade Federal de Santa Catarina, 2004. Citado na página 133.

RADEL, Simon; MULAHUWAISH, Aos; LEDUC, Ryan J. Fault tolerant controllability. In: IEEE. **2015 American Control Conference (ACC)**. Chicago, IL, Estados Unidos, 2015. p. 1603–1610. Citado na página 44.

RAMADGE, Peter J; WONHAM, W Murray. Supervisory control of a class of discrete event processes. **SIAM journal on control and optimization**, SIAM, v. 25, n. 1, p. 206–230, 1987. Citado na página 19.

REIJNEN, Ferdie et al. Synthesized fault-tolerant supervisory controllers, with an application to a rotating bridge. **Computers in Industry**, Elsevier, v. 130, p. 103473, 2021. Citado 4 vezes nas páginas 9, 54, 55 e 57.

REIJNEN, Ferdie et al. Structured synthesis of fault-tolerant supervisory controllers. **IFAC-PapersOnLine**, Elsevier, v. 51, n. 24, p. 894–901, 2018. Citado 2 vezes nas páginas 54 e 57.

SAMPATH, Meera et al. Diagnosability of discrete-event systems. **IEEE Transactions on automatic control**, IEEE, v. 40, n. 9, p. 1555–1575, 1995. Citado 4 vezes nas páginas 29, 30, 31 e 33.

SAMPATH, Meera et al. Failure diagnosis using discrete-event models. **IEEE transactions on control systems technology**, IEEE, v. 4, n. 2, p. 105–124, 1996. Citado na página 30.

SCHMIDT, Melanie; LUNZE, Jan. A framework for active fault-tolerant control of deterministic i/o automata. **IFAC Proceedings Volumes**, Elsevier, v. 47, n. 2, p. 446–452, 2014. Citado 3 vezes nas páginas 45, 49 e 57.

SCHUH, Melanie; LUNZE, Jan. Fault-tolerant control for deterministic discrete event systems with measurable state. In: IEEE. **2016 American Control Conference (ACC)**. Boston, MA, Estados Unidos, 2016. p. 7516–7522. Citado 2 vezes nas páginas 52 e 57.

SCHUH, Melanie; LUNZE, Jan. Fault-tolerant control of deterministic I/O automata using active diagnosis. In: IEEE. **2016 European Control Conference (ECC)**. Aalborg, Dinamarca, 2016. p. 2359–2365. Citado 2 vezes nas páginas 52 e 57.

SCHUH, Melanie; LUNZE, Jan. Fault-tolerant control of deterministic I/O automata with ambiguous diagnostic result. In: IEEE. **2016 13th International Workshop on Discrete Event Systems (WODES)**. Xi'an, China, 2016. p. 251–257. Citado 4 vezes nas páginas 9, 52, 53 e 57.

SCHUH, M; ZGORZELSKI, M; LUNZE, J. Experimental evaluation of an active fault–tolerant control method. **Control Engineering Practice**, Elsevier, v. 43, p. 1–11, 2015. Citado 4 vezes nas páginas 9, 50, 51 e 57.

SHU, Shaolong; LIN, Feng. Fault-tolerant control for safety of discrete-event systems. **IEEE Transactions on Automation Science and Engineering**, IEEE, v. 11, n. 1, p. 78–89, 2013. Citado 6 vezes nas páginas 9, 19, 45, 49, 50 e 57.

SÜLEK, Ayse Nur; SCHMIDT, Klaus Werner. Computation of fault-tolerant supervisors for discrete event systems. **IFAC Proceedings Volumes**, Elsevier, v. 46, n. 22, p. 115–120, 2013. Citado na página 44.

SWARTJES, Lennart; BEEK, Dirk A van; RENIERS, Michel A. Towards the removal of synchronous behavior of events in automata. **IFAC Proceedings Volumes**, Elsevier, v. 47, n. 2, p. 188–194, 2014. Citado na página 55.

TAHIRI, Imane et al. Timed synthesis control approach for tolerant-fault control of discrete event systems (des). In: IEEE. **2018 International Conference on Control, Automation and Diagnosis (ICCAD)**. Marraquexe, Marrocos, 2018. p. 1–6. Citado 4 vezes nas páginas 9, 53, 54 e 57.

UDESC. **GASR-FBE**. 2014. Disponível em: <https://sourceforge.net/projects/gasrfbe/>. Citado na página 41.

UDESC. **Github GASR UDESC**. 2021. Disponível em: <https://github.com/GASR-UDESC/FaultTolerantControl>. Citado na página 126.

UDESC. **Canal do YouTube do GASR UDESC**. 2022. Disponível em: <https://www.youtube.com/watch?v=1syeg3JsxiU>. Citado na página 127.

VIEIRA, Agnelo Denis et al. A method for plc implementation of supervisory control of discrete event systems. **IEEE Transactions on Control Systems Technology**, IEEE, v. 25, n. 1, p. 175–191, 2016. Citado na página 88.

WATANABE, Ana Teruko Yokomizo. **Controlabilidade segura de sistemas a eventos discretos utilizando diagnose e prognose online**. Tese (Doutorado) — Universidade do Estado de Santa Catarina, 2019. Citado 3 vezes nas páginas 32, 35 e 36.

WATANABE, Ana Teruko Yokomizo et al. Combining online diagnosis and prognosis for safe controllability. **IEEE Transactions on Automatic Control**, IEEE, v. 1, n. 1, p. 1–1, 2021. Citado 3 vezes nas páginas 20, 44 e 133.

WEN, Qin; KUMAR, Ratnesh; HUANG, Jing. Framework for optimal fault-tolerant control synthesis: Maximize prefault while minimize post-fault behaviors. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, IEEE, v. 44, n. 8, p. 1056–1066, 2013. Citado na página 44.

WEN, Qin et al. A framework for fault-tolerant control of discrete event systems. **IEEE Transactions on Automatic Control**, IEEE, v. 53, n. 8, p. 1839–1849, 2008. Citado 2 vezes nas páginas 46 e 57.

WITTMANN, Thomas; RICHTER, Jan; MOOR, Thomas. Fault-tolerant control of discrete event systems based on fault-accommodating models. **IFAC Proceedings Volumes**, Elsevier, v. 45, n. 20, p. 854–859, 2012. Citado 5 vezes nas páginas 44, 45, 47, 48 e 57.

WITTMANN, Th; RICHTER, JH; MOOR, Thomas. Fault-hiding control reconfiguration for a class of discrete event systems. **IFAC Proceedings Volumes**, Elsevier, v. 46, n. 22, p. 49–54, 2013. Citado na página 45.

ZAYTOON, Janan; RIERA, Bernard. Synthesis and implementation of logic controllers–a review. **Annual reviews in control**, Elsevier, v. 43, p. 152–168, 2017. Citado na página 19.

ZOITL, Alois. **Real-time Execution for IEC 61499**. [S.l.]: ISA, 2008. Citado 2 vezes nas páginas 38 e 39.

ZOITL, Alois; LEWIS, Robert. **Modelling control systems using IEC 61499**. [S.l.]: IET, 2014. v. 95. Citado 5 vezes nas páginas 20, 37, 38, 39 e 41.

APÊNDICE A – ARQUIVOS DOS BLOCOS DE FUNÇÃO DOS SUPERVISORES

```
<?xml version="1.0" encoding="UTF-8"?>
<FBType Name="Supervisor_1" Comment="Template for a simple Basic Function Block Type">
    <Identification Standard="61499-2">
    </Identification>
    <VersionInfo Version="1.0" Author="herbert" Date="2021-10-14">
    </VersionInfo>
    <InterfaceList>
        <EventInputs>
            <Event Name="COM_HAB" Type="Event" Comment="Comando de Habilitação">
                <With Var="ESTADO"/>
            </Event>
            <Event Name="LIBERAR" Type="Event" Comment="Liberação de Evento Controlável">
            </Event>
            <Event Name="VA_9s" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="Inf_Sup_lt5s" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="Inf_Sup_ge5s" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="Sup_Inf" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="C_abriu" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="C_fechou" Type="Event" Comment="Evento Não Controlável">
            </Event>
        </EventInputs>
        <EventOutputs>
            <Event Name="VA_abrir" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VA_fechar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VC_abrir" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VC_fechar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VD_abrir" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VD_fechar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="P_ligar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="P_desligar" Type="Event" Comment="Evento Controlável">
            </Event>
        </EventOutputs>
        <InputVars>
            <VarDeclaration Name="ESTADO" Type="INT" Comment="Identificação do Estado do
Supervisor"/>
        </InputVars>
    </InterfaceList>
    <BasicFB>
        <ECC>
            <ECState Name="Desabilitado" Comment="Initial State" x="1882.3529411764705"
y="117.6470588235294">
            </ECState>
            <ECState Name="Estado_1" Comment="Initialization" x="1294.1176470588234"
y="1058.8235294117646">
                <ECAction Output="VA_abrir"/>
            </ECState>
            <ECState Name="Liberar_Estado_1" Comment="Normal execution" x="588.2352941176471"
y="588.2352941176471">
            </ECState>
            <ECState Name="Estado_2" Comment="" x="235.2941176470588" y="1294.1176470588234">
            </ECState>
            <ECState Name="Estado_3" Comment="" x="1294.1176470588234" y="1529.4117647058822">
                <ECAction Output="VA_fechar"/>
            </ECState>
            <ECState Name="Estado_4" Comment="" x="235.2941176470588" y="1882.3529411764705">
                <ECAction Output="VC_abrir"/>
            </ECState>
            <ECState Name="Estado_5" Comment="" x="823.5294117647059" y="2352.9411764705883">
            </ECState>
            <ECState Name="Estado_6" Comment="" x="1882.3529411764705" y="2588.235294117647">
                <ECAction Output="VC_fechar"/>
            </ECState>
            <ECState Name="Estado_7" Comment="" x="2941.176470588235" y="2352.9411764705883">
                <ECAction Output="VD_abrir"/>
            </ECState>
            <ECState Name="Estado_8" Comment="" x="3764.705882352941" y="1882.3529411764705">
```

```
            </ECState>
            <ECState Name="Liberar_Estado_9" Comment="" x="2470.5882352941176"
y="1529.4117647058822">
            </ECState>
            <ECState Name="Estado_10" Comment="" x="3764.705882352941" y="705.8823529411765">
            </ECState>
            <ECState Name="Estado_11" Comment="" x="4117.647058823529" y="235.2941176470588">
                <ECAction Output="P_desligar"/>
            </ECState>
            <ECState Name="Estado_12" Comment="" x="2823.529411764706" y="352.94117647058823">
                <ECAction Output="VD_fechar"/>
            </ECState>
            <ECState Name="Estado_9" Comment="" x="3058.8235294117644" y="1058.8235294117646">
                <ECAction Output="P_ligar"/>
            </ECState>
            <ECTransition Source="Desabilitado" Destination="Liberar_Estado_1"
Condition="COM_HAB[ESTADO = 1]" Comment="" x="1188.235294117647" y="311.7647058823529"/>
            <ECTransition Source="Liberar_Estado_1" Destination="Desabilitado"
Condition="COM_HAB[ESTADO = 0]" Comment="" x="1729.4117647058822" y="452.9411764705882"/>
            <ECTransition Source="Liberar_Estado_1" Destination="Estado_1"
Condition="LIBERAR" Comment="" x="1217.6470588235293" y="882.3529411764705"/>
            <ECTransition Source="Estado_1" Destination="Estado_2" Condition="1" Comment=""
x="976.4705882352941" y="1211.764705882353"/>
            <ECTransition Source="Estado_2" Destination="Estado_3" Condition="VA_9s"
Comment="" x="917.6470588235294" y="1570.5882352941176"/>
            <ECTransition Source="Estado_3" Destination="Estado_4" Condition="1" Comment=""
x="970.5882352941176" y="1770.5882352941176"/>
            <ECTransition Source="Estado_4" Destination="Estado_5" Condition="1" Comment=""
x="670.5882352941176" y="2235.2941176470586"/>
            <ECTransition Source="Estado_5" Destination="Estado_6" Condition="Inf_Sup_ge5s"
Comment="" x="1717.6470588235293" y="2264.705882352941"/>
            <ECTransition Source="Estado_5" Destination="Estado_6" Condition="Inf_Sup_lt5s"
Comment="" x="1294.1176470588234" y="2741.176470588235"/>
            <ECTransition Source="Estado_6" Destination="Estado_7" Condition="1" Comment=""
x="2576.4705882352941" y="2447.0588235294117"/>
            <ECTransition Source="Estado_7" Destination="Estado_8" Condition="1" Comment=""
x="3511.7647058823527" y="2158.8235294117644"/>
            <ECTransition Source="Estado_8" Destination="Liberar_Estado_9"
Condition="C_abriu" Comment="" x="3670.5882352941176" y="1670.5882352941176"/>
            <ECTransition Source="Estado_8" Destination="Liberar_Estado_9"
Condition="C_fechou" Comment="" x="3382.3529411764703" y="1870.5882352941176"/>
            <ECTransition Source="Liberar_Estado_9" Destination="Desabilitado"
Condition="COM_HAB[ESTADO = 0]" Comment="" x="2323.529411764706" y="941.1764705882352"/>
            <ECTransition Source="Liberar_Estado_9" Destination="Estado_9"
Condition="LIBERAR" Comment="" x="3111.7647058823527" y="1358.8235294117646"/>
            <ECTransition Source="Estado_9" Destination="Estado_10" Condition="1" Comment=""
x="3494.1176470588234" y="917.6470588235294"/>
            <ECTransition Source="Estado_10" Destination="Estado_11" Condition="Sup_Inf"
Comment="" x="4252.941176470588" y="523.5294117647059"/>
            <ECTransition Source="Estado_11" Destination="Estado_12" Condition="1" Comment=""
x="3682.3529411764703" y="252.94117647058823"/>
            <ECTransition Source="Estado_12" Destination="Liberar_Estado_1" Condition="1"
Comment="" x="2005.8823529411764" y="629.4117647058823"/>
            <ECTransition Source="Desabilitado" Destination="Liberar_Estado_9"
Condition="COM_HAB[ESTADO = 9]" Comment="" x="2811.7647058823527" y="811.7647058823529"/>
        </ECC>
    </BasicFB>
</FBType>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<FBType Name="Supervisor_2" Comment="Template for a simple Basic Function Block Type">
    <Identification Standard="61499-2">
    </Identification>
    <VersionInfo Version="1.0" Author="herbert" Date="2021-10-14">
    </VersionInfo>
    <InterfaceList>
        <EventInputs>
            <Event Name="COM_HAB" Type="Event" Comment="Comando de Habilitação">
                <With Var="ESTADO"/>
            </Event>
            <Event Name="LIBERAR" Type="Event" Comment="Liberação de Evento Controlável">
            </Event>
            <Event Name="VB_9s" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="Inf_Sup_lt5s" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="Inf_Sup_ge5s" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="Sup_Inf" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="C_abriu" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="C_fechou" Type="Event" Comment="Evento Não Controlável">
            </Event>
        </EventInputs>
        <EventOutputs>
            <Event Name="VB_abrir" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VB_fechar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VC_abrir" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VC_fechar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VD_abrir" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VD_fechar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="P_ligar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="P_desligar" Type="Event" Comment="Evento Controlável">
            </Event>
        </EventOutputs>
        <InputVars>
            <VarDeclaration Name="ESTADO" Type="INT" Comment="Identificação do Estado do
Supervisor"/>
        </InputVars>
    </InterfaceList>
    <BasicFB>
        <ECC>
            <ECState Name="Desabilitado" Comment="Initial State" x="1882.3529411764705"
y="117.6470588235294">
            </ECState>
            <ECState Name="Estado_1" Comment="Initialization" x="1294.1176470588234"
y="1058.8235294117646">
                <ECAction Output="VB_abrir"/>
            </ECState>
            <ECState Name="Liberar_Estado_1" Comment="Normal execution" x="588.2352941176471"
y="588.2352941176471">
            </ECState>
            <ECState Name="Estado_2" Comment="" x="235.2941176470588" y="1294.1176470588234">
            </ECState>
            <ECState Name="Estado_3" Comment="" x="1294.1176470588234" y="1529.4117647058822">
                <ECAction Output="VB_fechar"/>
            </ECState>
            <ECState Name="Estado_4" Comment="" x="235.2941176470588" y="1882.3529411764705">
                <ECAction Output="VC_abrir"/>
            </ECState>
            <ECState Name="Estado_5" Comment="" x="823.5294117647059" y="2352.9411764705883">
            </ECState>
            <ECState Name="Estado_6" Comment="" x="1882.3529411764705" y="2588.235294117647">
                <ECAction Output="VC_fechar"/>
            </ECState>
            <ECState Name="Estado_7" Comment="" x="2941.176470588235" y="2352.9411764705883">
                <ECAction Output="VD_abrir"/>
            </ECState>
            <ECState Name="Estado_8" Comment="" x="3764.705882352941" y="1882.3529411764705">
```

```
            </ECState>
            <ECState Name="Liberar_Estado_9" Comment="" x="2470.5882352941176"
y="1529.4117647058822">
            </ECState>
            <ECState Name="Estado_10" Comment="" x="3764.705882352941" y="705.8823529411765">
            </ECState>
            <ECState Name="Estado_11" Comment="" x="4117.647058823529" y="235.2941176470588">
                <ECAction Output="P_desligar"/>
            </ECState>
            <ECState Name="Estado_12" Comment="" x="2823.529411764706" y="352.94117647058823">
                <ECAction Output="VD_fechar"/>
            </ECState>
            <ECState Name="Estado_9" Comment="" x="3058.8235294117644" y="1058.8235294117646">
                <ECAction Output="P_ligar"/>
            </ECState>
            <ECTransition Source="Desabilitado" Destination="Liberar_Estado_1"
Condition="COM_HAB[ESTADO = 1]" Comment="" x="1188.235294117647" y="311.7647058823529"/>
            <ECTransition Source="Liberar_Estado_1" Destination="Desabilitado"
Condition="COM_HAB[ESTADO = 0]" Comment="" x="1729.4117647058822" y="452.9411764705882"/>
            <ECTransition Source="Liberar_Estado_1" Destination="Estado_1"
Condition="LIBERAR" Comment="" x="1217.6470588235293" y="882.3529411764705"/>
            <ECTransition Source="Estado_1" Destination="Estado_2" Condition="1" Comment=""
x="976.4705882352941" y="1211.764705882353"/>
            <ECTransition Source="Estado_2" Destination="Estado_3" Condition="VB_9s"
Comment="" x="917.6470588235294" y="1570.5882352941176"/>
            <ECTransition Source="Estado_3" Destination="Estado_4" Condition="1" Comment=""
x="970.5882352941176" y="1770.5882352941176"/>
            <ECTransition Source="Estado_4" Destination="Estado_5" Condition="1" Comment=""
x="670.5882352941176" y="2235.2941176470586"/>
            <ECTransition Source="Estado_5" Destination="Estado_6" Condition="Inf_Sup_ge5s"
Comment="" x="1717.6470588235293" y="2264.705882352941"/>
            <ECTransition Source="Estado_5" Destination="Estado_6" Condition="Inf_Sup_lt5s"
Comment="" x="1294.1176470588234" y="2741.176470588235"/>
            <ECTransition Source="Estado_6" Destination="Estado_7" Condition="1" Comment=""
x="2576.4705882352944" y="2447.0588235294117"/>
            <ECTransition Source="Estado_7" Destination="Estado_8" Condition="1" Comment=""
x="3511.7647058823527" y="2158.8235294117644"/>
            <ECTransition Source="Estado_8" Destination="Liberar_Estado_9"
Condition="C_abriu" Comment="" x="3670.5882352941176" y="1670.5882352941176"/>
            <ECTransition Source="Estado_8" Destination="Liberar_Estado_9"
Condition="C_fechou" Comment="" x="3382.3529411764703" y="1870.5882352941176"/>
            <ECTransition Source="Liberar_Estado_9" Destination="Desabilitado"
Condition="COM_HAB[ESTADO = 0]" Comment="" x="2323.529411764706" y="941.1764705882352"/>
            <ECTransition Source="Liberar_Estado_9" Destination="Estado_9"
Condition="LIBERAR" Comment="" x="3111.7647058823527" y="1358.8235294117646"/>
            <ECTransition Source="Estado_9" Destination="Estado_10" Condition="1" Comment=""
x="3494.1176470588234" y="917.6470588235294"/>
            <ECTransition Source="Estado_10" Destination="Estado_11" Condition="Sup_Inf"
Comment="" x="4252.941176470588" y="523.5294117647059"/>
            <ECTransition Source="Estado_11" Destination="Estado_12" Condition="1" Comment=""
x="3682.3529411764703" y="252.94117647058823"/>
            <ECTransition Source="Estado_12" Destination="Liberar_Estado_1" Condition="1"
Comment="" x="2005.8823529411764" y="629.4117647058823"/>
            <ECTransition Source="Desabilitado" Destination="Liberar_Estado_9"
Condition="COM_HAB[ESTADO = 9]" Comment="" x="2811.7647058823527" y="811.7647058823529"/>
        </ECC>
    </BasicFB>
</FBType>
```

APÊNDICE B – ARQUIVOS DOS BLOCOS DE FUNÇÃO DOS DIAGNOSTICADORES

```
<?xml version="1.0" encoding="UTF-8"?>
<FBType Name="Diagnosticador_f_VA" Comment="Template for a simple Basic Function Block Type">
    <Identification Standard="61499-2">
    </Identification>
    <VersionInfo Version="1.0" Author="herbert" Date="2021-10-14">
    </VersionInfo>
    <InterfaceList>
        <EventInputs>
            <Event Name="COM_HAB" Type="Event" Comment="Comando de Habilitação">
                <With Var="ESTADO"/>
            </Event>
            <Event Name="VA_abrir" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VA_fechar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VA_9s" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="VC_abrir" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VC_fechar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="Inf_Sup_lt5s" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="Inf_Sup_ge5s" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="Sup_Inf" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="VD_abrir" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VD_fechar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="C_abriu" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="C_fechou" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="P_ligar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="P_desligar" Type="Event" Comment="Evento Controlável">
            </Event>
        </EventInputs>
        <EventOutputs>
            <Event Name="DIAGNOSE" Type="Event" Comment="Diagnóstico da Falha">
            </Event>
            <Event Name="CHAVEAR" Type="Event" Comment="Sinalização para Chaveamento">
            </Event>
            <Event Name="LIBERAR_ESTADO_1" Type="Event" Comment="Liberação da Ocorrência de
Evento Controlável">
            </Event>
        </EventOutputs>
        <InputVars>
            <VarDeclaration Name="ESTADO" Type="INT" Comment="Identificação do Estado do
Supervisor"/>
        </InputVars>
    </InterfaceList>
    <BasicFB>
        <ECC>
            <ECState Name="Desabilitado" Comment="Initial State" x="2235.2941176470586"
y="352.94117647058823">
            </ECState>
            <ECState Name="Estado_1NnB" Comment="Initialization" x="3999.9999999999995"
y="352.94117647058823">
                <ECAction Output="LIBERAR_ESTADO_1"/>
            </ECState>
            <ECState Name="Estado_2NnB_2YnB" Comment="" x="2823.529411764706"
y="941.1764705882352">
            </ECState>
            <ECState Name="Estado_3NnB_3YnB" Comment="" x="4117.647058823529"
y="1294.1176470588234">
            </ECState>
            <ECState Name="Estado_4NnB_4YnB" Comment="" x="3058.8235294117644"
y="1529.4117647058822">
            </ECState>
            <ECState Name="Estado_5NnB_5YnB" Comment="" x="4117.647058823529"
y="1882.3529411764705">
            </ECState>
            <ECState Name="Estado_6NnB" Comment="" x="5411.764705882352"
y="1882.3529411764705">
            </ECState>
```

```
            <ECState Name="Estado_7NnB" Comment="" x="6588.235294117647"
y="1882.3529411764705">
            </ECState>
            <ECState Name="Estado_8NnB" Comment="" x="5411.764705882352"
y="1411.764705882353">
            </ECState>
            <ECState Name="Estado_10NnB" Comment="" x="5411.764705882352"
y="705.8823529411765">
            </ECState>
            <ECState Name="Estado_11NnB" Comment="" x="6588.235294117647"
y="470.5882352941176">
            </ECState>
            <ECState Name="Estado_12NnB" Comment="" x="5411.764705882352"
y="117.6470588235294">
            </ECState>
            <ECState Name="Estado_9NnB" Comment="" x="6588.235294117647"
y="1058.8235294117646">
            </ECState>
            <ECState Name="Estado_6YnB" Comment="" x="3882.3529411764703"
y="2352.9411764705883">
                <ECAction Output="DIAGNOSE"/>
            </ECState>
            <ECState Name="Estado_7YnB" Comment="" x="5411.764705882352"
y="2235.2941176470586">
            </ECState>
            <ECState Name="Estado_8YnB" Comment="" x="6588.235294117647"
y="2470.5882352941176">
            </ECState>
            <ECState Name="Estado_9YnB" Comment="" x="5411.764705882352"
y="2823.529411764706">
            </ECState>
            <ECState Name="Estado_10YnB" Comment="" x="4117.647058823529"
y="2705.882352941176">
            </ECState>
            <ECState Name="Estado_11YnB" Comment="" x="2823.529411764706"
y="2470.5882352941176">
            </ECState>
            <ECState Name="Estado_12YnB" Comment="" x="2470.5882352941176"
y="1882.3529411764705">
            </ECState>
            <ECState Name="Estado_1YnB" Comment="" x="2235.2941176470586"
y="1176.4705882352941">
                <ECAction Output="CHAVEAR"/>
            </ECState>
            <ECTransition Source="Estado_1NnB" Destination="Estado_2NnB_2YnB"
Condition="VA_abrir" Comment="" x="3611.7647058823527" y="517.6470588235294"/>
            <ECTransition Source="Estado_2NnB_2YnB" Destination="Estado_3NnB_3YnB"
Condition="VA_9s" Comment="" x="4076.4705882352937" y="1141.1764705882351"/>
            <ECTransition Source="Estado_3NnB_3YnB" Destination="Estado_4NnB_4YnB"
Condition="VA_fechar" Comment="" x="3776.4705882352937" y="1382.3529411764705"/>
            <ECTransition Source="Estado_4NnB_4YnB" Destination="Estado_5NnB_5YnB"
Condition="VC_abrir" Comment="" x="4152.941176470588" y="1741.1764705882351"/>
            <ECTransition Source="Estado_5NnB_5YnB" Destination="Estado_6NnB"
Condition="Inf_Sup_ge5s" Comment="" x="5152.941176470588" y="1735.2941176470588"/>
            <ECTransition Source="Estado_6NnB" Destination="Estado_7NnB"
Condition="VC_fechar" Comment="" x="6311.764705882352" y="2141.176470588235"/>
            <ECTransition Source="Estado_7NnB" Destination="Estado_8NnB" Condition="VD_abrir"
Comment="" x="6170.588235294117" y="1741.1764705882351"/>
            <ECTransition Source="Estado_9NnB" Destination="Estado_10NnB" Condition="P_ligar"
Comment="" x="6382.35294117647" y="864.7058823529411"/>
            <ECTransition Source="Estado_10NnB" Destination="Estado_11NnB"
Condition="Sup_Inf" Comment="" x="6200.0" y="570.5882352941176"/>
            <ECTransition Source="Estado_11NnB" Destination="Estado_12NnB"
Condition="P_desligar" Comment="" x="6452.941176470587" y="317.6470588235294"/>
            <ECTransition Source="Desabilitado" Destination="Estado_1NnB"
Condition="COM_HAB[ESTADO = 1]" Comment="" x="3441.176470588235" y="252.9411764705882"/>
            <ECTransition Source="Estado_8NnB" Destination="Estado_9NnB" Condition="C_fechou"
Comment="" x="6105.882352941176" y="1247.0588235294117"/>
            <ECTransition Source="Estado_8NnB" Destination="Estado_9NnB" Condition="C_abriu"
Comment="" x="6511.764705882352" y="1423.5294117647059"/>
            <ECTransition Source="Estado_12NnB" Destination="Estado_1NnB"
Condition="VD_fechar" Comment="" x="4929.411764705882" y="247.05882352941174"/>
            <ECTransition Source="Estado_5NnB_5YnB" Destination="Estado_6YnB"
Condition="Inf_Sup_lt5s" Comment="" x="4252.941176470588" y="2170.5882352941176"/>
            <ECTransition Source="Estado_6YnB" Destination="Estado_7YnB"
Condition="VC_fechar" Comment="" x="4905.882352941176" y="2241.176470588235"/>
            <ECTransition Source="Estado_7YnB" Destination="Estado_8YnB" Condition="VD_abrir"
Comment="" x="6352.941176470587" y="2341.176470588235"/>
```

```
            <ECTransition Source="Estado_8YnB" Destination="Estado_9YnB" Condition="C_fechou"
Comment="" x="6158.823529411764" y="2617.6470588235293"/>
            <ECTransition Source="Estado_8YnB" Destination="Estado_9YnB" Condition="C_abriu"
Comment="" x="6452.941176470587" y="2835.2941176470586"/>
            <ECTransition Source="Estado_9YnB" Destination="Estado_10YnB" Condition="P_ligar"
Comment="" x="5111.764705882352" y="2705.882352941176"/>
            <ECTransition Source="Estado_10YnB" Destination="Estado_11YnB"
Condition="Sup_Inf" Comment="" x="3682.3529411764703" y="2776.470588235294"/>
            <ECTransition Source="Estado_11YnB" Destination="Estado_12YnB"
Condition="P_desligar" Comment="" x="3064.705882352941" y="2241.176470588235"/>
            <ECTransition Source="Estado_12YnB" Destination="Estado_1YnB"
Condition="VD_fechar" Comment="" x="2576.470588235294" y="1599.9999999999998"/>
            <ECTransition Source="Estado_1YnB" Destination="Desabilitado"
Condition="COM_HAB[ESTADO = 0]" Comment="" x="2558.8235294117644" y="852.9411764705882"/>
            <ECTransition Source="Estado_9NnB" Destination="Desabilitado"
Condition="COM_HAB[ESTADO = 0]" Comment="" x="4505.882352941177" y="805.8823529411765"/>
            <ECTransition Source="Desabilitado" Destination="Estado_9NnB"
Condition="COM_HAB[ESTADO = 9]" Comment="" x="4182.35294117647" y="941.1764705882352"/>
        </ECC>
    </BasicFB>
</FBType>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<FBType Name="Diagnosticador_f_VB" Comment="Template for a simple Basic Function Block Type">
    <Identification Standard="61499-2">
    </Identification>
    <VersionInfo Version="1.0" Author="herbert" Date="2021-10-14">
    </VersionInfo>
    <InterfaceList>
        <EventInputs>
            <Event Name="COM_HAB" Type="Event" Comment="Comando de Habilitação">
                <With Var="ESTADO"/>
            </Event>
            <Event Name="VB_abrir" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VB_fechar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VB_9s" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="VC_abrir" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VC_fechar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="Inf_Sup_lt5s" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="Inf_Sup_ge5s" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="Sup_Inf" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="VD_abrir" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VD_fechar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="C_abriu" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="C_fechou" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="P_ligar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="P_desligar" Type="Event" Comment="Evento Controlável">
            </Event>
        </EventInputs>
        <EventOutputs>
            <Event Name="DIAGNOSE" Type="Event" Comment="Diagnóstico da Falha">
            </Event>
            <Event Name="CHAVEAR" Type="Event" Comment="Sinalização para Chaveamento">
            </Event>
            <Event Name="LIBERAR_ESTADO_1" Type="Event" Comment="Liberação da Ocorrência de
Evento Controlável">
            </Event>
        </EventOutputs>
        <InputVars>
            <VarDeclaration Name="ESTADO" Type="INT" Comment="Identificação do Estado do
Supervisor"/>
        </InputVars>
    </InterfaceList>
    <BasicFB>
        <ECC>
            <ECState Name="Desabilitado" Comment="Initial State" x="2235.2941176470586"
y="352.94117647058823">
            </ECState>
            <ECState Name="Estado_1NnB" Comment="Initialization" x="3999.9999999999995"
y="352.94117647058823">
                <ECAction Output="LIBERAR_ESTADO_1"/>
            </ECState>
            <ECState Name="Estado_2NnB_2YnB" Comment="" x="2823.529411764706"
y="941.1764705882352">
            </ECState>
            <ECState Name="Estado_3NnB_3YnB" Comment="" x="4117.647058823529"
y="1294.1176470588234">
            </ECState>
            <ECState Name="Estado_4NnB_4YnB" Comment="" x="3058.8235294117644"
y="1529.4117647058822">
            </ECState>
            <ECState Name="Estado_5NnB_5YnB" Comment="" x="4117.647058823529"
y="1882.3529411764705">
            </ECState>
            <ECState Name="Estado_6NnB" Comment="" x="5411.764705882352"
y="1882.3529411764705">
            </ECState>
```

```
            <ECState Name="Estado_7NnB" Comment="" x="6588.235294117647"
y="1882.3529411764705">
            </ECState>
            <ECState Name="Estado_8NnB" Comment="" x="5411.764705882352"
y="1411.764705882353">
            </ECState>
            <ECState Name="Estado_10NnB" Comment="" x="5411.764705882352"
y="705.8823529411765">
            </ECState>
            <ECState Name="Estado_11NnB" Comment="" x="6588.235294117647"
y="470.5882352941176">
            </ECState>
            <ECState Name="Estado_12NnB" Comment="" x="5411.764705882352"
y="235.2941176470588">
            </ECState>
            <ECState Name="Estado_9NnB" Comment="" x="6588.235294117647"
y="1058.8235294117646">
            </ECState>
            <ECState Name="Estado_6YnB" Comment="" x="3882.3529411764703"
y="2352.9411764705883">
                <ECAction Output="DIAGNOSE"/>
            </ECState>
            <ECState Name="Estado_7YnB" Comment="" x="5411.764705882352"
y="2235.2941176470586">
            </ECState>
            <ECState Name="Estado_8YnB" Comment="" x="6588.235294117647"
y="2470.5882352941176">
            </ECState>
            <ECState Name="Estado_9YnB" Comment="" x="5411.764705882352"
y="2823.529411764706">
            </ECState>
            <ECState Name="Estado_10YnB" Comment="" x="4117.647058823529"
y="2705.882352941176">
            </ECState>
            <ECState Name="Estado_11YnB" Comment="" x="2823.529411764706"
y="2470.5882352941176">
            </ECState>
            <ECState Name="Estado_12YnB" Comment="" x="2470.5882352941176"
y="1882.3529411764705">
            </ECState>
            <ECState Name="Estado_1YnB" Comment="" x="2235.2941176470586"
y="1176.4705882352941">
                <ECAction Output="CHAVEAR"/>
            </ECState>
            <ECTransition Source="Estado_1NnB" Destination="Estado_2NnB_2YnB"
Condition="VB_abrir" Comment="" x="3611.7647058823527" y="517.6470588235294"/>
            <ECTransition Source="Estado_2NnB_2YnB" Destination="Estado_3NnB_3YnB"
Condition="VB_9s" Comment="" x="4076.4705882352937" y="1141.1764705882351"/>
            <ECTransition Source="Estado_3NnB_3YnB" Destination="Estado_4NnB_4YnB"
Condition="VB_fechar" Comment="" x="3776.4705882352937" y="1382.3529411764705"/>
            <ECTransition Source="Estado_4NnB_4YnB" Destination="Estado_5NnB_5YnB"
Condition="VC_abrir" Comment="" x="4152.941176470588" y="1741.1764705882351"/>
            <ECTransition Source="Estado_5NnB_5YnB" Destination="Estado_6NnB"
Condition="Inf_Sup_ge5s" Comment="" x="5152.941176470588" y="1735.2941176470588"/>
            <ECTransition Source="Estado_6NnB" Destination="Estado_7NnB"
Condition="VC_fechar" Comment="" x="6311.764705882352" y="2141.176470588235"/>
            <ECTransition Source="Estado_7NnB" Destination="Estado_8NnB" Condition="VD_abrir"
Comment="" x="6170.588235294117" y="1741.1764705882351"/>
            <ECTransition Source="Estado_9NnB" Destination="Estado_10NnB" Condition="P_ligar"
Comment="" x="6382.35294117647" y="864.7058823529411"/>
            <ECTransition Source="Estado_10NnB" Destination="Estado_11NnB"
Condition="Sup_Inf" Comment="" x="6200.0" y="570.5882352941176"/>
            <ECTransition Source="Estado_11NnB" Destination="Estado_12NnB"
Condition="P_desligar" Comment="" x="6452.941176470587" y="317.6470588235294"/>
            <ECTransition Source="Desabilitado" Destination="Estado_1NnB"
Condition="COM_HAB[ESTADO = 1]" Comment="" x="3441.176470588235" y="252.9411764705882"/>
            <ECTransition Source="Estado_8NnB" Destination="Estado_9NnB" Condition="C_fechou"
Comment="" x="6105.882352941176" y="1247.0588235294117"/>
            <ECTransition Source="Estado_8NnB" Destination="Estado_9NnB" Condition="C_abriu"
Comment="" x="6511.764705882352" y="1423.5294117647059"/>
            <ECTransition Source="Estado_12NnB" Destination="Estado_1NnB"
Condition="VD_fechar" Comment="" x="4929.411764705882" y="247.05882352941174"/>
            <ECTransition Source="Estado_5NnB_5YnB" Destination="Estado_6YnB"
Condition="Inf_Sup_lt5s" Comment="" x="4252.941176470588" y="2170.5882352941176"/>
            <ECTransition Source="Estado_6YnB" Destination="Estado_7YnB"
Condition="VC_fechar" Comment="" x="4900.0" y="2241.176470588235"/>
            <ECTransition Source="Estado_7YnB" Destination="Estado_8YnB" Condition="VD_abrir"
Comment="" x="6352.941176470587" y="2341.176470588235"/>
```

```
            <ECTransition Source="Estado_8YnB" Destination="Estado_9YnB" Condition="C_fechou"
Comment="" x="6158.823529411764" y="2617.6470588235293"/>
            <ECTransition Source="Estado_8YnB" Destination="Estado_9YnB" Condition="C_abriu"
Comment="" x="6452.941176470587" y="2835.2941176470586"/>
            <ECTransition Source="Estado_9YnB" Destination="Estado_10YnB" Condition="P_ligar"
Comment="" x="5111.764705882352" y="2705.882352941176"/>
            <ECTransition Source="Estado_10YnB" Destination="Estado_11YnB"
Condition="Sup_Inf" Comment="" x="3682.3529411764703" y="2776.470588235294"/>
            <ECTransition Source="Estado_11YnB" Destination="Estado_12YnB"
Condition="P_desligar" Comment="" x="3064.705882352941" y="2241.176470588235"/>
            <ECTransition Source="Estado_12YnB" Destination="Estado_1YnB"
Condition="VD_fechar" Comment="" x="2576.470588235294" y="1599.9999999999998"/>
            <ECTransition Source="Estado_1YnB" Destination="Desabilitado"
Condition="COM_HAB[ESTADO = 0]" Comment="" x="2558.8235294117644" y="852.9411764705882"/>
            <ECTransition Source="Estado_9NnB" Destination="Desabilitado"
Condition="COM_HAB[ESTADO = 0]" Comment="" x="4505.882352941177" y="805.8823529411765"/>
            <ECTransition Source="Desabilitado" Destination="Estado_9NnB"
Condition="COM_HAB[ESTADO = 9]" Comment="" x="4182.35294117647" y="941.1764705882352"/>
        </ECC>
    </BasicFB>
</FBType>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<FBType Name="Diagnosticador_f_VD(S_1)" Comment="Template for a simple Basic Function Block
Type">
    <Identification Standard="61499-2">
    </Identification>
    <VersionInfo Version="1.0" Author="herbert" Date="2021-10-14">
    </VersionInfo>
    <InterfaceList>
        <EventInputs>
            <Event Name="COM_HAB" Type="Event" Comment="Comando de Habilitação">
                <With Var="ESTADO"/>
            </Event>
            <Event Name="VA_abrir" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VA_fechar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VA_9s" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="VC_abrir" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VC_fechar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="Inf_Sup_lt5s" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="Inf_Sup_ge5s" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="Sup_Inf" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="VD_abrir" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VD_fechar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="C_abriu" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="C_fechou" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="P_ligar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="P_desligar" Type="Event" Comment="Evento Controlável">
            </Event>
        </EventInputs>
        <EventOutputs>
            <Event Name="DIAGNOSE" Type="Event" Comment="Diagnóstico da Falha">
            </Event>
            <Event Name="CHAVEAR" Type="Event" Comment="Sinalização para Chaveamento">
            </Event>
            <Event Name="LIBERAR_ESTADO_9" Type="Event" Comment="Liberação da Ocorrência de
Evento Controlável">
            </Event>
        </EventOutputs>
        <InputVars>
            <VarDeclaration Name="ESTADO" Type="INT" Comment="Identificação do Estado do
Supervisor"/>
        </InputVars>
    </InterfaceList>
    <BasicFB>
        <ECC>
            <ECState Name="Desabilitado" Comment="Initial State" x="2235.2941176470586"
y="352.94117647058823">
            </ECState>
            <ECState Name="Estado_1NnB_1YnB" Comment="Initialization" x="3999.9999999999995"
y="352.94117647058823">
            </ECState>
            <ECState Name="Estado_2NnB_2YnB" Comment="" x="3176.4705882352937"
y="941.1764705882352">
            </ECState>
            <ECState Name="Estado_3NnB_3YnB" Comment="" x="4117.647058823529"
y="1294.1176470588234">
            </ECState>
            <ECState Name="Estado_4NnB_4YnB" Comment="" x="3058.8235294117644"
y="1529.4117647058822">
            </ECState>
            <ECState Name="Estado_5NnB_5YnB" Comment="" x="4117.647058823529"
y="1882.3529411764705">
            </ECState>
            <ECState Name="Estado_6NnB_6YnB" Comment="" x="5411.764705882352"
y="1882.3529411764705">
            </ECState>
```

```
            <ECState Name="Estado_7NnB_7YnB" Comment="" x="5764.7058823529405"
y="1411.764705882353">
            </ECState>
            <ECState Name="Estado_8NnB_8YnB" Comment="" x="6470.588235294117"
y="2352.9411764705883">
            </ECState>
            <ECState Name="Estado_9YnB" Comment="" x="2235.2941176470586"
y="2352.9411764705883">
                <ECAction Output="DIAGNOSE"/>
                <ECAction Output="CHAVEAR"/>
            </ECState>
            <ECState Name="Estado_12NnB" Comment="" x="5411.764705882352"
y="235.2941176470588">
            </ECState>
            <ECState Name="Estado_11NnB" Comment="" x="6588.235294117647"
y="470.5882352941176">
            </ECState>
            <ECState Name="Estado_10NnB" Comment="" x="5411.764705882352"
y="705.8823529411765">
            </ECState>
            <ECState Name="Estado_9NnB" Comment="" x="6588.235294117647"
y="1058.8235294117646">
                <ECAction Output="LIBERAR_ESTADO_9"/>
            </ECState>
            <ECTransition Source="Estado_1NnB_1YnB" Destination="Estado_2NnB_2YnB"
Condition="VA_abrir" Comment="" x="4017.6470588235293" y="794.1176470588234"/>
            <ECTransition Source="Estado_2NnB_2YnB" Destination="Estado_3NnB_3YnB"
Condition="VA_9s" Comment="" x="4076.4705882352937" y="1141.1764705882351"/>
            <ECTransition Source="Estado_3NnB_3YnB" Destination="Estado_4NnB_4YnB"
Condition="VA_fechar" Comment="" x="3776.4705882352937" y="1382.3529411764705"/>
            <ECTransition Source="Estado_4NnB_4YnB" Destination="Estado_5NnB_5YnB"
Condition="VC_abrir" Comment="" x="4152.941176470588" y="1741.1764705882351"/>
            <ECTransition Source="Estado_5NnB_5YnB" Destination="Estado_6NnB_6YnB"
Condition="Inf_Sup_ge5s" Comment="" x="5152.941176470588" y="1735.2941176470588"/>
            <ECTransition Source="Estado_9NnB" Destination="Estado_10NnB" Condition="P_ligar"
Comment="" x="6382.35294117647" y="864.7058823529411"/>
            <ECTransition Source="Estado_10NnB" Destination="Estado_11NnB"
Condition="Sup_Inf" Comment="" x="6200.0" y="570.5882352941176"/>
            <ECTransition Source="Estado_11NnB" Destination="Estado_12NnB"
Condition="P_desligar" Comment="" x="6452.941176470587" y="317.6470588235294"/>
            <ECTransition Source="Desabilitado" Destination="Estado_1NnB_1YnB"
Condition="COM_HAB[ESTADO = 1]" Comment="" x="3441.176470588235" y="252.9411764705882"/>
            <ECTransition Source="Estado_12NnB" Destination="Estado_1NnB_1YnB"
Condition="VD_fechar" Comment="" x="4929.411764705882" y="247.05882352941174"/>
            <ECTransition Source="Estado_7NnB_7YnB" Destination="Estado_8NnB_8YnB"
Condition="VD_abrir" Comment="" x="6511.764705882352" y="1888.2352941176468"/>
            <ECTransition Source="Estado_8NnB_8YnB" Destination="Estado_9YnB"
Condition="C_abriu" Comment="" x="4576.470588235294" y="2135.2941176470586"/>
            <ECTransition Source="Estado_5NnB_5YnB" Destination="Estado_6NnB_6YnB"
Condition="Inf_Sup_lt5s" Comment="" x="5135.294117647059" y="1541.1764705882351"/>
            <ECTransition Source="Estado_6NnB_6YnB" Destination="Estado_7NnB_7YnB"
Condition="VC_fechar" Comment="" x="6000.0" y="1694.1176470588234"/>
            <ECTransition Source="Estado_8NnB_8YnB" Destination="Estado_9NnB"
Condition="C_fechou" Comment="" x="7047.058823529412" y="1782.3529411764705"/>
            <ECTransition Source="Estado_9YnB" Destination="Desabilitado"
Condition="COM_HAB[ESTADO = 0]" Comment="" x="2841.176470588235" y="1158.8235294117646"/>
            <ECTransition Source="Estado_1NnB_1YnB" Destination="Desabilitado"
Condition="COM_HAB[ESTADO = 0]" Comment="" x="3447.0588235294117" y="623.5294117647059"/>
            <ECTransition Source="Desabilitado" Destination="Estado_9YnB"
Condition="COM_HAB[ESTADO = 9]" Comment="" x="2300.0" y="1341.1764705882351"/>
        </ECC>
    </BasicFB>
</FBType>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<FBType Name="Diagnosticador_f_VD(S_2)" Comment="Template for a simple Basic Function Block
Type">
    <Identification Standard="61499-2">
    </Identification>
    <VersionInfo Version="1.0" Author="herbert" Date="2021-10-14">
    </VersionInfo>
    <InterfaceList>
        <EventInputs>
            <Event Name="COM_HAB" Type="Event" Comment="Comando de Habilitação">
                <With Var="ESTADO"/>
            </Event>
            <Event Name="VB_abrir" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VB_fechar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VB_9s" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="VC_abrir" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VC_fechar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="Inf_Sup_lt5s" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="Inf_Sup_ge5s" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="Sup_Inf" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="VD_abrir" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="VD_fechar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="C_abriu" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="C_fechou" Type="Event" Comment="Evento Não Controlável">
            </Event>
            <Event Name="P_ligar" Type="Event" Comment="Evento Controlável">
            </Event>
            <Event Name="P_desligar" Type="Event" Comment="Evento Controlável">
            </Event>
        </EventInputs>
        <EventOutputs>
            <Event Name="DIAGNOSE" Type="Event" Comment="Diagnóstico da Falha">
            </Event>
            <Event Name="CHAVEAR" Type="Event" Comment="Sinalização para Chaveamento">
            </Event>
            <Event Name="LIBERAR_ESTADO_9" Type="Event" Comment="Liberação da Ocorrência de
Evento Controlável">
            </Event>
        </EventOutputs>
        <InputVars>
            <VarDeclaration Name="ESTADO" Type="INT" Comment="Identificação do Estado do
Supervisor"/>
        </InputVars>
    </InterfaceList>
    <BasicFB>
        <ECC>
            <ECState Name="Desabilitado" Comment="Initial State" x="2235.2941176470586"
y="352.94117647058823">
            </ECState>
            <ECState Name="Estado_1NnB_1YnB" Comment="Initialization" x="3999.9999999999995"
y="352.94117647058823">
            </ECState>
            <ECState Name="Estado_2NnB_2YnB" Comment="" x="3176.4705882352937"
y="941.1764705882352">
            </ECState>
            <ECState Name="Estado_3NnB_3YnB" Comment="" x="4117.647058823529"
y="1294.1176470588234">
            </ECState>
            <ECState Name="Estado_4NnB_4YnB" Comment="" x="3058.8235294117644"
y="1529.4117647058822">
            </ECState>
            <ECState Name="Estado_5NnB_5YnB" Comment="" x="4117.647058823529"
y="1882.3529411764705">
            </ECState>
            <ECState Name="Estado_6NnB_6YnB" Comment="" x="5411.764705882352"
y="1882.3529411764705">
            </ECState>
```

```
            <ECState Name="Estado_7NnB_7YnB" Comment="" x="5764.7058823529405"
y="1411.764705882353">
            </ECState>
            <ECState Name="Estado_8NnB_8YnB" Comment="" x="6470.588235294117"
y="2352.9411764705883">
            </ECState>
            <ECState Name="Estado_9YnB" Comment="" x="2235.2941176470586"
y="2352.9411764705883">
                <ECAction Output="DIAGNOSE"/>
                <ECAction Output="CHAVEAR"/>
            </ECState>
            <ECState Name="Estado_12NnB" Comment="" x="5411.764705882352"
y="235.2941176470588">
            </ECState>
            <ECState Name="Estado_11NnB" Comment="" x="6588.235294117647"
y="470.5882352941176">
            </ECState>
            <ECState Name="Estado_10NnB" Comment="" x="5411.764705882352"
y="705.8823529411765">
            </ECState>
            <ECState Name="Estado_9NnB" Comment="" x="6588.235294117647"
y="1058.8235294117646">
                <ECAction Output="LIBERAR_ESTADO_9"/>
            </ECState>
            <ECTransition Source="Estado_1NnB_1YnB" Destination="Estado_2NnB_2YnB"
Condition="VB_abrir" Comment="" x="4017.6470588235293" y="794.1176470588234"/>
            <ECTransition Source="Estado_2NnB_2YnB" Destination="Estado_3NnB_3YnB"
Condition="VB_9s" Comment="" x="4076.4705882352937" y="1141.1764705882351"/>
            <ECTransition Source="Estado_3NnB_3YnB" Destination="Estado_4NnB_4YnB"
Condition="VB_fechar" Comment="" x="3776.4705882352937" y="1382.3529411764705"/>
            <ECTransition Source="Estado_4NnB_4YnB" Destination="Estado_5NnB_5YnB"
Condition="VC_abrir" Comment="" x="4152.941176470588" y="1741.1764705882351"/>
            <ECTransition Source="Estado_5NnB_5YnB" Destination="Estado_6NnB_6YnB"
Condition="Inf_Sup_ge5s" Comment="" x="5152.941176470588" y="1735.2941176470588"/>
            <ECTransition Source="Estado_9NnB" Destination="Estado_10NnB" Condition="P_ligar"
Comment="" x="6382.35294117647" y="864.7058823529411"/>
            <ECTransition Source="Estado_10NnB" Destination="Estado_11NnB"
Condition="Sup_Inf" Comment="" x="6200.0" y="570.5882352941176"/>
            <ECTransition Source="Estado_11NnB" Destination="Estado_12NnB"
Condition="P_desligar" Comment="" x="6452.941176470587" y="317.6470588235294"/>
            <ECTransition Source="Desabilitado" Destination="Estado_1NnB_1YnB"
Condition="COM_HAB[ESTADO = 1]" Comment="" x="3441.176470588235" y="252.9411764705882"/>
            <ECTransition Source="Estado_12NnB" Destination="Estado_1NnB_1YnB"
Condition="VD_fechar" Comment="" x="4929.411764705882" y="247.05882352941174"/>
            <ECTransition Source="Estado_7NnB_7YnB" Destination="Estado_8NnB_8YnB"
Condition="VD_abrir" Comment="" x="6511.764705882352" y="1888.2352941176468"/>
            <ECTransition Source="Estado_8NnB_8YnB" Destination="Estado_9YnB"
Condition="C_abriu" Comment="" x="4576.470588235294" y="2135.2941176470586"/>
            <ECTransition Source="Estado_5NnB_5YnB" Destination="Estado_6NnB_6YnB"
Condition="Inf_Sup_lt5s" Comment="" x="5135.294117647059" y="1541.1764705882351"/>
            <ECTransition Source="Estado_6NnB_6YnB" Destination="Estado_7NnB_7YnB"
Condition="VC_fechar" Comment="" x="6000.0" y="1694.1176470588234"/>
            <ECTransition Source="Estado_8NnB_8YnB" Destination="Estado_9NnB"
Condition="C_fechou" Comment="" x="7047.058823529412" y="1782.3529411764705"/>
            <ECTransition Source="Estado_9YnB" Destination="Desabilitado"
Condition="COM_HAB[ESTADO = 0]" Comment="" x="2841.176470588235" y="1158.8235294117646"/>
            <ECTransition Source="Estado_1NnB_1YnB" Destination="Desabilitado"
Condition="COM_HAB[ESTADO = 0]" Comment="" x="3447.0588235294117" y="623.5294117647059"/>
            <ECTransition Source="Desabilitado" Destination="Estado_9YnB"
Condition="COM_HAB[ESTADO = 9]" Comment="" x="2300.0" y="1341.1764705882351"/>
        </ECC>
    </BasicFB>
</FBType>
```

APÊNDICE C – ARQUIVOS DO BLOCO DE FUNÇÃO DO MECANISMO DE CHAVEAMENTO

```
<?xml version="1.0" encoding="UTF-8"?>
<FBType Name="Mecanismo_de_Chaveamento" Comment="Template for a simple Basic Function Block
Type">
    <Identification Standard="61499-2">
    </Identification>
    <VersionInfo Version="1.0" Author="herbert" Date="2021-09-13">
    </VersionInfo>
    <InterfaceList>
        <EventInputs>
            <Event Name="INIT" Type="Event" Comment="Initialization Request">
                <With Var="COND_VA"/>
                <With Var="COND_VB"/>
                <With Var="COND_VD"/>
            </Event>
            <Event Name="SINAL_DIAG_VA" Type="Event" Comment="Sinalização">
                <With Var="COND_VA"/>
            </Event>
            <Event Name="SINAL_DIAG_VB" Type="Event" Comment="">
                <With Var="COND_VB"/>
            </Event>
            <Event Name="SINAL_DIAG_VD1" Type="Event" Comment="">
                <With Var="COND_VD"/>
            </Event>
            <Event Name="SINAL_DIAG_VD2" Type="Event" Comment="">
                <With Var="COND_VD"/>
            </Event>
            <Event Name="SINAL_REC" Type="Event" Comment="Evento de Recuperação">
                <With Var="COND_VA"/>
                <With Var="COND_VB"/>
                <With Var="COND_VD"/>
            </Event>
        </EventInputs>
        <EventOutputs>
            <Event Name="INITO" Type="Event" Comment="Initialization Confirm">
            </Event>
            <Event Name="COM_HAB" Type="Event" Comment="Comando para a Condição de
Habilitação">
                <With Var="ESTADO_SUP1"/>
                <With Var="ESTADO_SUP2"/>
            </Event>
        </EventOutputs>
        <InputVars>
            <VarDeclaration Name="COND_VA" Type="BOOL" Comment="Condição de VA"/>
            <VarDeclaration Name="COND_VB" Type="BOOL" Comment="Condição de VB"/>
            <VarDeclaration Name="COND_VD" Type="BOOL" Comment="COndição de VD"/>
        </InputVars>
        <OutputVars>
            <VarDeclaration Name="ESTADO_SUP1" Type="INT" Comment="Controle de Operação do
Supervisor A"/>
            <VarDeclaration Name="ESTADO_SUP2" Type="INT" Comment="Controle de Operação do
Supervisor B"/>
        </OutputVars>
    </InterfaceList>
    <BasicFB>
        <ECC>
            <ECState Name="START" Comment="Initial State" x="235.2941176470588"
y="823.5294117647059">
            </ECState>
            <ECState Name="UPDT" Comment="" x="1247.0588235294117" y="676.470588235294">
                <ECAction Output="INITO"/>
            </ECState>
            <ECState Name="SUP1" Comment="" x="3294.1176470588234" y="0.0">
                <ECAction Output="COM_HAB"/>
            </ECState>
            <ECState Name="BLQ1" Comment="" x="5176.470588235294" y="352.94117647058823">
                <ECAction Algorithm="ALG_BLQ" Output="COM_HAB"/>
            </ECState>
            <ECState Name="UPDT_SUP2" Comment="" x="1647.0588235294117" y="1411.764705882353">
                <ECAction Algorithm="ALG_UPDT_SUP2"/>
            </ECState>
            <ECState Name="BLQ2" Comment="" x="5176.470588235294" y="1411.764705882353">
                <ECAction Algorithm="ALG_BLQ" Output="COM_HAB"/>
            </ECState>
            <ECState Name="UPDT_SUP1" Comment="" x="1882.3529411764705" y="117.6470588235294">
                <ECAction Algorithm="ALG_UPDT_SUP1"/>
            </ECState>
            <ECState Name="BLQ1_SUP1" Comment="" x="3647.0588235294117" y="705.8823529411765">
                <ECAction Algorithm="ALG_BLQ1_SUP1"/>
```

```
            </ECState>
            <ECState Name="SUP2" Comment="" x="3294.1176470588234" y="1647.0588235294117">
                <ECAction Output="COM_HAB"/>
            </ECState>
            <ECState Name="BLQ2_SUP2" Comment="" x="3647.0588235294117" y="941.1764705882352">
                <ECAction Algorithm="ALG_BLQ1_SUP2"/>
            </ECState>
            <ECState Name="SUP1_SUP2" Comment="" x="2235.2941176470586" y="588.2352941176471">
                <ECAction Algorithm="ALG_SUP1_SUP2"/>
            </ECState>
            <ECTransition Source="START" Destination="UPDT" Condition="INIT" Comment=""
x="905.8823529411764" y="799.9999999999999"/>
            <ECTransition Source="SUP1" Destination="BLQ1" Condition="SINAL_DIAG_VD1"
Comment="" x="4205.882352941177" y="305.88235294117646"/>
            <ECTransition Source="UPDT" Destination="UPDT_SUP1" Condition="[COND_VA = TRUE
AND COND_VD = TRUE]" Comment="" x="1247.0588235294117" y="-11.76470588235294"/>
            <ECTransition Source="UPDT" Destination="UPDT_SUP2" Condition="[COND_VB = TRUE
AND COND_VD = TRUE]" Comment="" x="1288.235294117647" y="1223.5294117647059"/>
            <ECTransition Source="UPDT" Destination="UPDT" Condition="SINAL_REC" Comment=""
x="1564.705882352941" y="388.235294117647"/>
            <ECTransition Source="UPDT_SUP1" Destination="SUP1" Condition="1" Comment=""
x="2776.470588235294" y="-58.8235294117647"/>
            <ECTransition Source="BLQ1" Destination="BLQ1_SUP1" Condition="SINAL_REC[COND_VD
= TRUE]" Comment="" x="4482.35294117647" y="570.5882352941176"/>
            <ECTransition Source="BLQ1_SUP1" Destination="SUP1" Condition="1" Comment=""
x="3547.0588235294117" y="429.4117647058823"/>
            <ECTransition Source="BLQ2" Destination="BLQ2_SUP2" Condition="SINAL_REC[COND_VD
= TRUE]" Comment="" x="4552.941176470588" y="1247.0588235294117"/>
            <ECTransition Source="BLQ2_SUP2" Destination="SUP2" Condition="1" Comment=""
x="3605.882352941176" y="1329.4117647058822"/>
            <ECTransition Source="UPDT_SUP2" Destination="SUP2" Condition="1" Comment=""
x="2811.7647058823527" y="1647.0588235294117"/>
            <ECTransition Source="SUP2" Destination="UPDT" Condition="SINAL_DIAG_VB"
Comment="" x="2594.1176470588234" y="1123.5294117647059"/>
            <ECTransition Source="SUP2" Destination="BLQ2" Condition="SINAL_DIAG_VD2"
Comment="" x="4417.647058823529" y="1423.5294117647059"/>
            <ECTransition Source="SUP1" Destination="SUP1_SUP2" Condition="SINAL_DIAG_VA"
Comment="" x="2964.705882352941" y="405.88235294117646"/>
            <ECTransition Source="SUP1_SUP2" Destination="SUP2" Condition="1" Comment=""
x="3129.411764705882" y="1088.235294117647"/>
        </ECC>
        <Algorithm Name="ALG_UPDT_SUP1" Comment="new algorithm">
            <ST><![CDATA[ESTADO_SUP1 := 1;ESTADO_SUP2 := 0;]]></ST>
        </Algorithm>
        <Algorithm Name="ALG_BLQ" Comment="new algorithm">
            <ST><![CDATA[ESTADO_SUP1 := 0;ESTADO_SUP2 := 0;]]></ST>
        </Algorithm>
        <Algorithm Name="ALG_BLQ1_SUP1" Comment="new algorithm">
            <ST><![CDATA[ESTADO_SUP1 := 9;ESTADO_SUP2 := 0;]]></ST>
        </Algorithm>
        <Algorithm Name="ALG_UPDT_SUP2" Comment="new algorithm">
            <ST><![CDATA[ESTADO_SUP1 := 0;ESTADO_SUP2 := 1;]]></ST>
        </Algorithm>
        <Algorithm Name="ALG_BLQ1_SUP2" Comment="new algorithm">
            <ST><![CDATA[ESTADO_SUP1 := 0;ESTADO_SUP2 := 9;]]></ST>
        </Algorithm>
        <Algorithm Name="ALG_SUP1_SUP2" Comment="new algorithm">
            <ST><![CDATA[ESTADO_SUP1 := 0;ESTADO_SUP2 := 1;]]></ST>
        </Algorithm>
    </BasicFB>
</FBType>
```