Controle Supervisório Tolerante a Falhas de Sistemas a Eventos Discretos: Estudo de Caso ⋆

Bruno Mota * **Samuel Oliveira** **,*** **André B. Leal** ***

* *Grupo de Pesquisa em Automação de Sistemas e Robótica (GASR)*
Universidade do Estado de Santa Catarina, Joinville/SC
** *Departamento de Ciências Exatas e Tecnológicas*
Universidade Federal do Amapá, Macapá/AP
*** *Programa de Pós-Graduação em Engenharia Elétrica*
Universidade do Estado de Santa Catarina, Joinville/SC

Abstract: In this work, we address the problem of supervisory control tolerant to intermittent faults in manufacturing systems modeled as discrete event systems (DES). Two fault tolerance approaches were applied to a case study developed on a virtual plant using Factory IO software: active approach and fault-hiding. In both cases, a diagnoser automaton was used to detect fault occurrences. Modular supervisors were designed based on the supervisory control theory of DES, and the fault-tolerant control structure was implemented in a PLC, controlling the virtual plant via the OPC communication protocol. Different strategies for implementing the fault-tolerant control structure are proposed and applied to the case study, resulting in the proper functioning of the plant, even with the intermittent occurrence of faults.

Resumo: Neste trabalho trata-se do problema de controle supervisório tolerante a falhas intermitentes de sistemas de manufatura modelados como sistemas a eventos discretos (SEDs). Duas abordagens de tolerância a falhas foram aplicadas a um estudo de caso desenvolvido sobre uma planta virtual no *software* Factory IO: abordagem ativa e ocultação de falhas. Em ambos os casos foi utilizado um autômato diagnosticador para detectar a ocorrência de falhas. Supervisores modulares foram projetados com base na teoria de controle supervisório de SEDs e a estrutura de controle tolerante a falhas foi implementada em CLP, controlando a planta virtual via protocolo de comunicação OPC. Diferentes estratégias de implementação da estrutura de controle tolerante a falhas são propostas e aplicadas sobre o estudo de caso, obtendo-se com todas elas o correto funcionamento da planta, mesmo com a intermitente ocorrência de falha.

Keywords: Discrete event systems; fault-tolerant control; fault-hiding; Modular control; Virtual plant.

Palavras-chaves: Sistemas a eventos discretos; Controle tolerante a falhas; Ocultação de falhas; Controle modular; Planta virtual.

1. INTRODUÇÃO

Sistemas de automação têm sido amplamente utilizados em uma variedade de setores, como manufatura, transporte, e outros. Muitos destes sistemas são caracterizados como Sistemas a Eventos Discretos (SEDs), pois sua evolução dinâmica depende da ocorrência de eventos discretos. Em geral, os formalismos de Autômatos e Redes de Petri são utilizados para a modelagem e controle de SEDs (Cassandras e Lafortune, 2021).

De acordo com Sampath (1995), sistemas de automação estão sujeitos a falhas, que podem ser compreendidas como qualquer desvio de um sistema do seu comportamento pretendido (Maas et al., 2021). Em células produtivas, a operação em caso de falhas nos dispositivos pode levar a perdas monetárias devido à possível paralisação da produção, aos possíveis danos nos demais equipamentos do sistema e até mesmo, a riscos na segurança operacional. Nesse sentido, o desenvolvimento de uma estratégia de controle capaz de identificar e mitigar a ocorrência de falhas se torna de extrema importância (Moreira e Leal, 2020). Uma abordagem amplamente utilizada para lidar com esse problema no contexto de SEDs é conhecida como *controle tolerante a falhas* (Moor, 2016; Watanabe et al., 2022).

Para tratar de um controlador tolerante a falhas, é necessário que o sistema seja capaz de identificar a ocorrência de falhas em seus dispositivos (Leitão et al., 2020). Para tanto, utiliza-se, em geral, um autômato diagnosticador, como o proposto por Sampath et al. (1995), em que é possível tanto verificar a capacidade de diagnosticar uma falha a partir do modelo do sistema, quanto realizar a diagnose durante a execução da planta. Uma vez que é possível detectar a ocorrência da falha, pode-se aplicar alguma estratégia de controle tolerante a falhas, como a metodologia proposta por Paoli et al. (2011). Essa metodologia é chamada de controle tolerante a falhas ativo

⋆ O presente trabalho foi realizado com apoio parcial da Fundação de Amparo à Pesquisa e Inovação do Estado de Santa Catarina - FAPESC 2023TR000333.

(CTFA), em que são desenvolvidos: (i) um controlador para a operação nominal da planta; (ii) um controlador para operação da planta em falha, e (iii) um diagnosticador para detecção da falha, de maneira que a detecção da falha é usada como sinal para chavear o controlador, ou parte do controlador, como ilustrado na Figura 1.

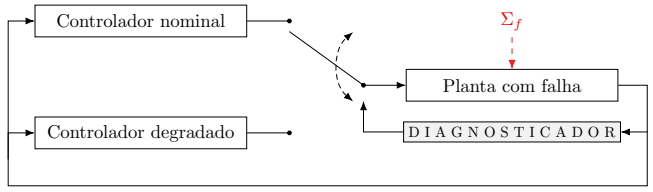


Figura 1. Arquitetura de um CTFA. Adaptado de Paoli et al. (2011).

Uma variação do controle ativo é a ocultação de falha, do Inglês, *fault-hiding* (Wittmann et al., 2013), que utiliza um bloco de reconfiguração que deve diagnosticar a ocorrência da falha e ajustar as entradas ou saídas para adaptar o controlador nominal da planta na situação de falha escondendo o efeito da falha. A Figura 2 ilustra essa arquitetura.

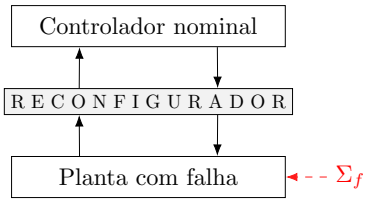


Figura 2. Arquitetura de controle tolerante a falhas com *Fault-hiding*. Adaptado de Wittmann et al. (2013).

Este trabalho tem como objetivo investigar a aplicação de metodologias para o controle tolerante a falhas em uma planta virtual de automação. A implementação dos códigos no Controlador Lógico Programável (CLP) foi realizada por meio da estrutura de controle modular local de acordo com de Queiroz e Cury (2000) e Vieira et al. (2017). Essa técnica utiliza métodos formais para a obtenção de supervisores locais que, em conjunto, garantem uma lógica de controle que atende às especificações de controle com restrições mínimas e sem bloqueios. Foi escolhido utilizar um CLP virtual e a comunicação entre o CLP e o *software* de simulação Factory IO foi feito utilizando comunicação OPC. Dessa maneira, as particularidades de desenvolvimento e aplicação, além de um comparativo de implementação entre as metodologias CTFA e ocultação de falhas são analisadas e demonstradas através de fluxogramas.

O presente artigo é organizado da seguinte forma: na Seção 2 são apresentados conceitos preliminares. A Seção 3 contém detalhes sobre a realização dos testes e simulações. A Seção 4 apresenta os resultados obtidos e, por fim, na Seção 5 são apresentadas as discussões e conclusões do trabalho.

2. CONCEITOS PRELIMINARES

2.1 Modelagem e controle de SEDs

Neste artigo, o formalismo de autômatos é utilizado para modelagem de SEDs. Segundo Cassandras e Lafortune (2021), um autômato pode ser representado por uma quíntupla $G = (X, \Sigma, f, x_0, X_m)$ em que X é um conjunto finito de estados; Σ representa o alfabeto de eventos; $f : X \times \Sigma \to X$ a função de transição, possivelmente parcial; x_0 é o estado inicial do autômato, e por fim, X_m é o conjunto de estados marcados ou finais, de maneira que $X_m \subseteq X$. Autômatos são utilizados para representar e manipular linguagens formais sobre o conjunto de eventos Σ. O Fecho de Kleene é uma operação denominada Σ^* que representa o conjunto de todas as sequências finitas de elementos de Σ, incluindo a palavra vazia ε. Assim, a função de transição $f : X \times \Sigma$ é estendida recursivamente para $f : X \times \Sigma^*$, de tal maneira que $f(x, \varepsilon) := x$ e $f(x, s\sigma) := f[f(x, s), \sigma]$ para $s \in \Sigma^*$ e $\sigma \in \Sigma$.

Um SED pode ser modelado por um autômato G que representa e manipula duas linguagens: a linguagem gerada por G e a linguagem marcada por G. A linguagem gerada por G é definida como $L(G) := \{s \in \Sigma^* | f(x_0, s)$ é definido$\}$. Já a linguagem marcada por G é definida como $L_m(G) := \{s \in L(G) | f(x_0, s) \in X_m\}$. Ou seja, a linguagem $L_m(G)$ é o subconjunto da linguagem $L(G)$ que contém todas as sequências cujas funções de transição levam a um estado final, representando o comportamento de G no qual as tarefas são concluídas.

Na teoria de controle supervisório (TCS) de SEDs, o conjunto de eventos Σ é particionado como $\Sigma_o \cup \Sigma_{uo}$, e também como $\Sigma_c \cup \Sigma_{uc}$, os quais denotam os eventos observáveis, não observáveis, controláveis e não controláveis respectivamente. A projeção de uma sequência a partir do conjunto Σ^* no conjunto Σ_o^* é uma operação denotada por $P_o : \Sigma^* \to \Sigma_o^*$ e é definida como: $P_o(\varepsilon) = \varepsilon$; $P_o(\sigma) = \sigma$ se $\sigma \in \Sigma_o$, ou $P_o(\sigma) = \varepsilon$ se $\sigma \in \Sigma_{uo}$; $P_o(s\sigma) = P_o(s)P_o(\sigma)$, para $s \in \Sigma^*$ e $\sigma \in \Sigma$. Por outro lado, a projeção inversa é, por definição, $P_o^{-1}(t) = \{s \in \Sigma^* : P_o(s) = t\}$ (Cassandras e Lafortune, 2021).

Em um sistema de automação modelado como SED, caso os subsistemas que compõem a planta não compartilharem eventos, eles podem ser considerados como: *sistema produto (SP)*, e passam a ser controlados por um ou mais supervisores numa estrutura de malha fechada. Dessa forma, os supervisores possuem a capacidade de observar os eventos gerados na planta e, então, desabilitar eventos controláveis com o objetivo de restringir o comportamento da planta com base em especificações de controle predefinidas.

2.2 Implementação em CLP

Para implementar essa estrutura de controle, é necessário que o CLP atue como uma máquina de estados. Nesse sentido, este trabalho segue o modelo proposto por Leal et al. (2009), pois permite o processamento de múltiplos eventos em um único ciclo de varredura, mantendo a coerência nas ações de controle.

Um fluxograma que representa os passos de implementação de um CLP com base na estrutura de controle modular

local de SEDs pode ser visto na Figura 3. Foi escolhido um fluxograma, pois ele demonstra a ordem que cada bloco deve ser implementado, sendo que cada um desses blocos representam um passo necessário para o funcionamento do código.

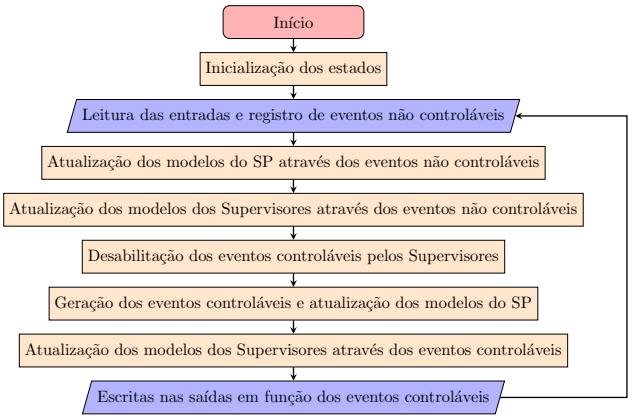


Figura 3. Fluxograma para implementação da estrutura de controle supervisório modular.

2.3 Diagnose de falhas

A diagnose de falhas consiste na detecção do mau funcionamento de um subsistema e a diagnosticabilidade desta falha consiste na propriedade que garante que, se a falha ocorrer, ela será detectada após um número finito de transições depois de sua ocorrência usando somente sequências de eventos observáveis (Zaytoon e Lafortune, 2013). Nesse sentido, para que uma linguagem L seja diagnosticável, sempre que o evento de falha σ_f ocorrer, ele deverá ser detectado após um número finito de observações de eventos.

Seja $\Sigma_f = \{\sigma_f\} \subseteq \Sigma_{uo}$ o conjunto cujo único evento σ_f modela uma falha do sistema e que L denote a linguagem gerada por G. Uma das maneiras de se verificar a diagnosticabilidade de uma linguagem é por meio do chamado diagnosticador. O diagnosticador é um autômato determinístico cujo conjunto de eventos é igual ao conjunto dos eventos observáveis de G e cujos estados são formados adicionando-se rótulos aos estados de G para indicar se o evento σ_f ocorreu ou não. Os rótulos indicam estados normais (certos de que a falha não ocorreu), estados incertos (não há como afirmar se a falha ocorreu ou não) e estados certos de falha. Caso o autômato diagnosticador permanecer indefinidamente em um ciclo formado somente por estados incertos, então não será possível diagnosticar a ocorrência do evento de falha σ_f (Basilio et al., 2010).

3. TESTES E SIMULAÇÕES

3.1 Planta e controladores

Os testes das metodologias de controle tolerante a falhas foram realizados em uma modificação de uma planta disponível no *software* Factory IO, ilustrada na Figura 4 e referida, aqui, como Estação de Separação. A lista de subsistemas é apresentada a seguir.

(1) Entrada de peças;
(2) Esteira de entrada;

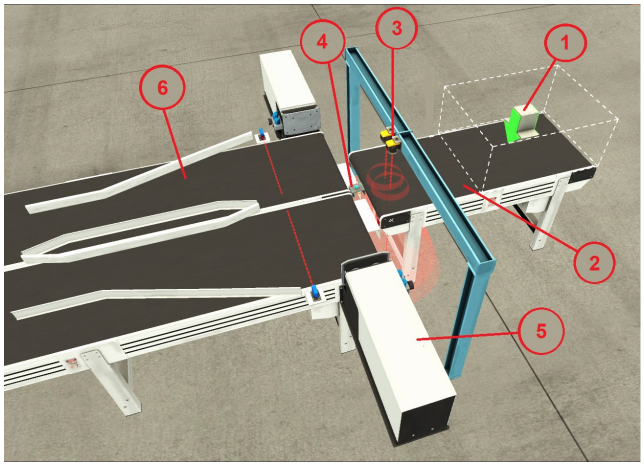


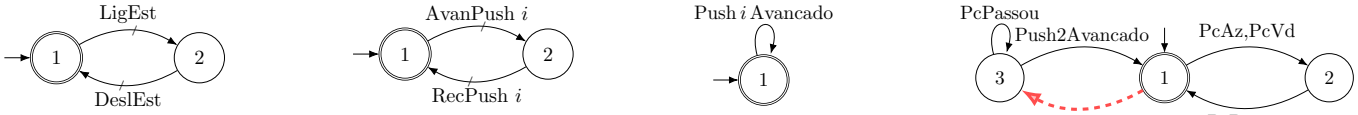
Figura 4. Estação de Separação.

(3) Sensor de visão;
(4) Sensor de passagem;
(5) Atuador pneumático com sensores de fim de curso (estendido e recolhido);
(6) Esteiras de saída.

O sistema opera nominalmente da seguinte maneira: dois tipos de peças, azuis e verdes, são alimentados na esteira de entrada. As peças entram uma de cada vez, em ordem aleatória e com um espaçamento mínimo entre elas. Um sensor de visão identifica o tipo de peça, enquanto um sensor de presença indica quando a peça sai da esteira de entrada. Nesse momento, um evento é gerado, ativando um dos atuadores pneumáticos. O atuador é acionado de forma a direcionar as peças verdes para uma das esteiras de saída e as peças azuis para outra. Por questões de simplicidade, considera-se que as esteiras de saída estão sempre em funcionamento.

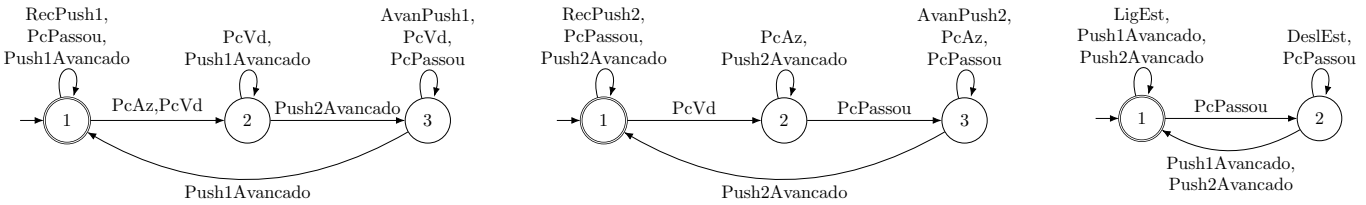
Na Figura 5 são apresentados os autômatos que modelam os subsistemas da estação de separação. Especificamente, a Figura 5a apresenta o autômato que modela esteira de entrada, no qual os eventos `LigEst` e `DeslEst` representam os sinais de ativação e desativação, respectivamente. O autômato apresentado na Figura 5b modela o comportamento dos dois atuadores pneumáticos, sendo que o evento `RecPush`i faz o atuador retornar para a posição recolhida e `AvanPush`i faz o atuador começar o movimento de avanço. O autômato da Figura 5c representa a ativação dos sensores que indicam que o atuador está completamente avançado, por meio do evento `Push`i`Avancado`. Por fim, o autômato 5d representa a combinação dos sinais de dois sensores: `PcAz` e `PcVd`, que estão ligados ao sensor de visão na identificação do tipo de peça. Já o evento `PcPassou` representa que uma peça deixou de estar na frente do sensor de presença na saída da esteira de entrada. Os dois sensores foram modelados juntos para que se criasse uma ordem entre o acontecimento de seus eventos. Ou seja, após o sensor de visão identificar um tipo de peça, ele só poderá enviar outro evento de tipo de peça após o item já ter saído da esteira de entrada.

Note que o modelo do sensor de visão (Fig. 5d) já contém o evento de falha que leva do estado 1 ao estado 3. Note também que este modelo se assemelha a um tipo de falha intermitente (Carvalho et al., 2017), uma vez que, o sensor



(a) Esteira (b) Atuador $i = \{1, 2\}$ (c) Sensor de fim de curso do Atuador $i = \{1, 2\}$ (d) Sensor de Visão

Figura 5. Autômatos para os subsistemas da planta.



(a) Atuador 1 (A1) (b) Atuador 2 (A2) (c) Atuador 2 (A2)

Figura 6. Autômatos para os supervisores da planta.

pode deixar e voltar a funcionar de forma irregular e o controlador manterá o funcionamento da planta.

Para fazer a planta seguir seu comportamento nominal, foram criadas especificações de controle que resultaram nos supervisores reduzidos ilustrados na Figura 6, através de operações de redução realizadas pelo *software* TCT Wonham (2005), de acordo com a síntese de supervisores modulares locais. Os autômatos apresentados nas Figuras 6a e 6b se referem aos supervisores reduzidos que controlam o avanço e recuo dos atuadores pneumáticos. Já a Figura 6c apresenta o supervisor reduzido que define restrições de controle para a esteira de entrada. Vale destacar que os supervisores modulares reduzidos apresentados na Figura 6 são iguais às especificações de controle projetadas. Utilizando os conceitos de diagnose de falhas foi obtido um autômato diagnosticador para permitir a identificação da ocorrência do evento de falha. A modelagem dos autômatos, da planta e de controle, bem como os cálculos necessários para obtenção dos supervisores foram realizados através do *software* Nadzoru (Pinheiro et al., 2015).

3.2 Diagnose da Falha

Considera-se, aqui, que a falha no sensor de visão está relacionada com a impossibilidade de identificar peças na cor azul. Para tanto, foram utilizados dois sensores de visão no simulador, sendo um para identificar peças verdes e outro para identificar peças azuis. A Figura 7 apresenta o diagnosticador utilizado para identificação da falha.

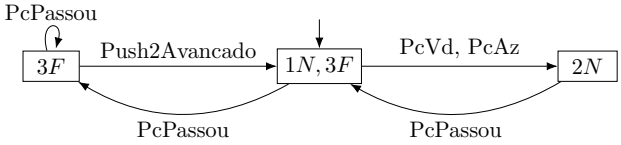


Figura 7. Diagnosticador utilizado para identificação da falha.

No comportamento nominal, o diagnosticador ficará trocando entre dois estados normais de funcionamento, $(1N, 3F)$ e $(2N)$. O primeiro representa um estado de incerteza da ocorrência da falha e o segundo um estado de certeza de que a falha não ocorreu. Caso ocorra o evento `PcPassou` sem que antes tivesse ocorrido o evento que indica o tipo da peça, o sistema é levado ao estado de certeza de falha, identificado como $(3F)$. Ao final, na ocorrência do evento `Push2Avancado` o modelo do diagnosticador retorna para o estado $(1N, 3F)$. Ou seja, caso o sistema se recupere da falha, imediatamente será retomado o controle nominal.

3.3 Metodologias de Controle Tolerante a Falhas

CTFA. Para que o CTFA seja implementado no sistema, é necessário desabilitar o autômato que comanda o subsistema que apresenta a falha. Especificamente, sensor de visão deixa de gerar o evento `PcAz` em caso de falha. Dessa forma, o autômato de controle do segundo atuador pneumático fica naturalmente desligado, travado no primeiro estado. Em seguida, deve-se chavear o novo autômato de controle para a situação de falha. Nesse caso o autômato chaveado é demonstrado pela Figura 8.

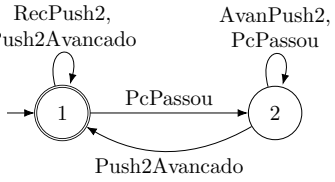


Figura 8. Autômato de controle do atuador pneumático após a ocorrência da falha.

Ao final dessa sequência criada pelo autômato chaveado, o diagnosticador é reiniciado, e portanto, desabilita o controlador para falha. Isso é feito para que, caso entre uma peça da cor verde o controlador não habilite a ativação do Atuador 2. Além disso, caso o sensor de peças azuis volte a funcionar, o sistema passa a ser controlado pelos seus supervisores nominais sem a necessidade de parar ou reconfigurar a linha.

Fault-hiding. No controle tolerante a falhas através da técnica de *Fault-hiding*, oculta-se a falha do sensor de visão da seguinte forma: quando o diagnosticador detecta a falha, é criado um evento com o mesmo nome do evento que deveria ser criado pelo sensor defeituoso, ou seja, é criado um evento `PcAz`. Dessa forma o supervisor de controle nominal da planta age como se a falha nunca tivesse acontecido, mantendo o funcionamento normal do sistema. Como o controle nominal nunca deixa de atuar na planta, caso o sensor volte a funcionar o sistema não precisará ser parado ou reiniciado.

Para ambas as abordagens, os autômatos das plantas, supervisores, diagnosticadores e controladores pós falha foram implementados em um CLP virtual com comunicação OPC para o *software* de simulação Factory IO. A metodologia de implementação tem como base o modelo proposto por Leal et al. (2009) de implementação de estruturas de controle supervisório modular. As propostas de modificação dessa abordagem são descritas na Seção 4.

4. RESULTADOS

A Figura 9 demonstra os fluxogramas, propostos pelo artigo, para a implementação em CLP do supervisor nominal da planta em junção com o controlador tolerante a falhas. Assim como o modelo de implementação de Leal et al. (2009), nessa proposta também é permitido o processamento de múltiplos eventos em um único ciclo de varredura, mantendo a coerência nas ações de controle. Os blocos realçados em vermelho demonstram as posições onde forma feitas modificações na teoria original.

Para o controle com ocultação de falhas, como se trata da criação de um evento não controlável que será utilizado no decorrer do código, é notório que o novo bloco do diagnosticador e controlador deve estar localizado entre o bloco de leitura das entradas e registro dos eventos não controláveis e o bloco de atualização dos modelos do sistema produto através dos eventos não controláveis. A Figura 9a representa a proposta de fluxograma para a implementação modular utilizando o controle de ocultação de falhas.

Já o controle tolerante a falhas ativo possui um bloco de diagnose e controle com comportamento de um sequenciador de tarefas. Dessa forma, a única restrição é que ele seja inserido após o bloco de leitura das entradas e registro dos eventos não controláveis. Por motivos de organização, decidiu-se posicioná-lo após a escrita nas saídas em função dos eventos controláveis. O fluxograma que representa essa proposta de implementação é apresentado na Figura 9b.

Estudou-se também a possibilidade de implementação do diagnosticador e do controlador em blocos separados. Nesse caso o controlador não precisaria ser projetado como um sequenciador, mas normalmente como um autômato na implementação modular. Com isso o bloco do diagnosticador deve ficar após o registro dos eventos não controláveis, mas antes das atualizações dos supervisores através dos eventos não controláveis. Já o modelo do controlador poderia ser aplicado juntamente com os blocos de atualização dos supervisores, dessa forma é criado um bloco novo para o diagnosticador e por conta do CTFA estar separado ele acaba modificando diversos blocos do modelo original. Essa proposta foi exemplificada pelo fluxograma da Figura 9c.

Os fluxogramas propostos neste trabalho são uma hipótese de implementação do CTFA e ocultação de falhas, sendo que ainda se fazem necessários novos estudos para validação da estrutura descrita. Para esta aplicação em específico, com apenas uma falha em um sensor, a metodologia *fault hiding* se mostrou mais vantajosa, devido a sua maior simplicidade de implementação. Entretanto, a avaliação de diferentes estudos de caso se faz necessária a fim de definir vantagens e desvantagens entre o CTFA e *fault hiding*, bem como validar os fluxogramas propostos.

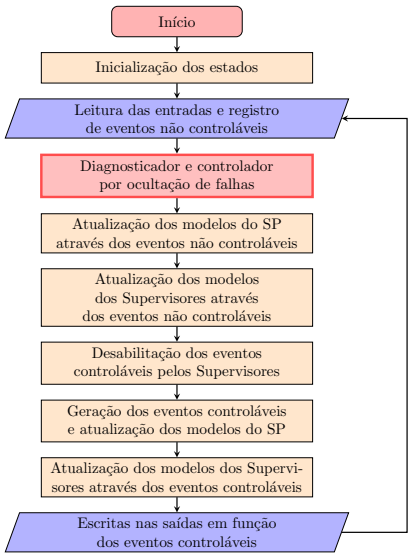
Para demonstração e validação dos testes, foram gravados e disponibilizados na internet dois vídeos do funcionamento da planta. O primeiro vídeo [1] demonstra o comportamento da planta antes e após a ocorrência da falha, porém, sem um sistema de detecção ou de controle tolerante a falhas. Com a planta em seu estado normal as peças são separadas perfeitamente, contudo após forçar uma falha no sensor de visão, o segundo atuador pneumático deixa de ser acionado e as peças azuis seguem um caminho indeterminado. Já no segundo vídeo [2], é demonstrado o comportamento da planta utilizando a técnica de controle de ocultação de falhas. Antes da falha acontecer o sistema funciona normalmente, no entanto, após forçar a falha no sensor o sistema de diagnose e controle são ativados e a planta continua funcionando conforme o projetado. No vídeo, ainda é ilustrada a situação em que a falha é corrigida, sendo possível verificar que o sistema continua separando as peças corretamente. Foram realizados os testes e gravado um vídeo com o funcionamento do controle tolerante a falhas ativo, porém não será apresentado, já que ele é visualmente idêntico ao vídeo que demonstra a ocultação de falhas.

5. CONCLUSÃO

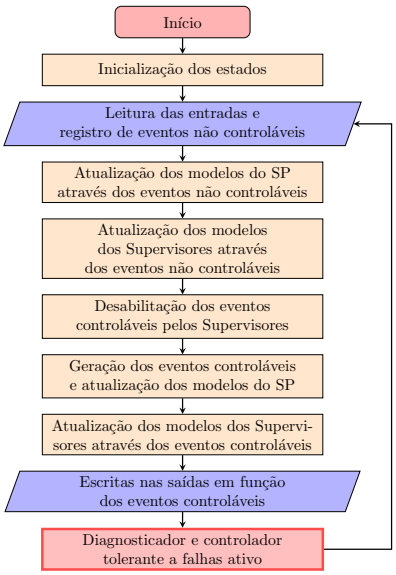
A principal contribuição deste artigo foi a demonstração da utilização das técnicas de controle tolerante a falhas CTFA e *fault-hiding*, validando sua funcionalidade através de um estudo de caso baseado em uma planta de manufatura, em que foram aplicados testes e simulações de maneira virtual no software Factory IO.

Destaca-se a importância de serem utilizadas abordagens formais no projeto de controle supervisório e na implementação de CLP's, tendo em vista a crescente demanda por resiliência contra falhas em sistemas de automação modelados como SEDs e a complexidade dos problemas de controle. Neste trabalho, a abordagem modular local para a síntese de supervisores foi utilizada, tanto na fase de modelagem, quanto na implementação de CLP's. Dessa maneira, foi possível identificar e sugerir hipóteses de possíveis posicionamentos do bloco diagnosticador e controlador na metodologia de implementação da estrutura de controle modular local em sistemas de controle tolerante a falhas, sendo necessário para trabalhos futuros a validação dessas propostas através da análise de diferentes estudos de caso.

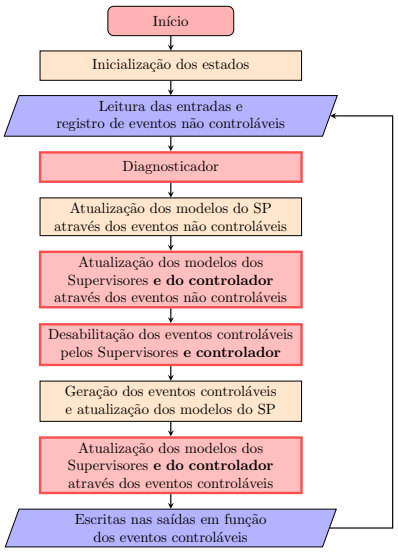
[1] Disponível em: <https://youtu.be/ShapifeArag>
[2] Disponível em: <https://youtu.be/2IE7r-TTDfo>



(a) Fluxograma de implementação modular proposto para o controle de ocultação de falhas.



(b) Fluxograma de implementação modular proposto para o controle tolerante a falhas ativo, considerando um bloco diagnosticador e controlador.



(c) Fluxograma de implementação modular proposto para o controle tolerante a falhas ativo, considerando diagnosticador e controlador separados.

Figura 9. Fluxogramas de implementação de estruturas de controle tolerante a falhas.

REFERÊNCIAS

Basilio, J.C., Carvalho, L.K., e Moreira, M.V. (2010). Diagnose de falhas em sistemas a eventos discretos modelados por autômatos finitos. *Sba: Controle & Automação - Sociedade Brasileira de Automatica*, 21(5), 510–533.

Carvalho, L.K., Moreira, M.V., e Basilio, J.C. (2017). Diagnosability of intermittent sensor faults in discrete event systems. *Automatica*, 79, 315–325.

Cassandras, C.G. e Lafortune, S. (2021). *Introduction to Discrete Event Systems*. Springer Cham, 3 edition.

de Queiroz, M.H. e Cury, J.E.R. (2000). Modular supervisory control of large scale discrete event systems. In R. Boel e G. Stremersch (eds.), *Discrete Event Systems: Analysis and Control*, 103–110. Springer US, Boston, MA.

Leal, A.B., da Cruz, D.L.L., e da S. Hounsell, M. (2009). Supervisory control implementation into programmable logic controllers. *IEEE Conference on Emerging Technologies Factory Automation*, 1–7.

Leitão, H.A.S., Rosso, R.S.U., Leal, A.B., e Zoitl, A. (2020). Fault handling in discrete event systems applied to IEC 61499. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, 1039–1042.

Maas, D., Sebem, R., e Leal, A.B. (2021). Multilayer architecture for fault diagnosis of embedded systems. *International Journal of Prognostics and Health Management*, 12(2), 1–10.

Moor, T. (2016). A discussion of fault-tolerant supervisory control in terms of formal languages. *Annual Reviews in Control*, 41, 159–169.

Moreira, B.G. e Leal, A.B. (2020). A proposal for an active diagnoser for safe fault-tolerant control of discrete event systems. *IFAC-PapersOnLine*, 53(4), 282–287.

Paoli, A., Sartini, M., e Lafortune, S. (2011). Active fault tolerant control of discrete event systems using online diagnostics. *Automatica*, 47(4), 639–649.

Pinheiro, L.P., Lopes, Y.K., Leal, A.B., e Rosso Junior, R.S.U. (2015). Nadzoru: A software tool for supervisory control of discrete event systems. *IFAC-PapersOnLine*, 48(7), 182–187.

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., e Teneketzis, D.C. (1995). Diagnosticability of discrete-event models. *IEE Transactions on Automatic Control*, 1555–1575.

Sampath, M. (1995). *A Discrete Event Systems Approach to Failure Diagnosis*. Ph.D. thesis, The University of Michigan, Michigan, EUA.

Vieira, A.D., Santos, E.A.P., de Queiroz, M.H., Leal, A.B., de Paula Neto, A.D., e Cury, J.E.R. (2017). A method for plc implementation of supervisory control of discrete event systems. *IEEE Transactions on Control Systems Technology*, 25(1), 175–191.

Watanabe, A.T.Y., Leal, A.B., Cury, J.E.R., e Queiroz, M.H.d. (2022). Combining online diagnosis and prognosis for safe controllability. *IEEE Transactions on Automatic Control*, 67(10), 5563–5569.

Wittmann, T., Richter, J., e Moor, T. (2013). Fault-hiding control reconfiguration for a class of discrete event systems. *IFAC Proceedings Volumes*, 46(22), 49–54.

Wonham, W. (2005). *Supervisory Control of Discrete-Event Systems and Design Software*. Dept. of Electrical and Computer Engineering, University of Toronto.

Zaytoon, J. e Lafortune, S. (2013). Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 37(2), 308–320.