XI CONGRESSO DE INICIAÇÃO CIENTÍFICA DO IFSP ITAPETININGA
Itapetininga, 27, 28 e 29 de maio de 2025
Instituto Federal de Educação, Ciência e Tecnologia de São Paulo
Campus Itapetininga

TESTES AUTOMATIZADOS PARA SISTEMA DE EXECUÇÃO DE BLOCOS FUNCIONAIS DEFINIDOS PELO USUÁRIO USANDO SIKULIX

Guilherme Santos da Silveira – ITI/IFSP[1]
Prof. Dr. Eduardo André Mossin - IFSP[2]
Prof. Dr. Rodrigo Palucci Pantoni - IFSP[3]

Introdução

O padrão O-PAS, desenvolvido pelo *The Open Group*, está em especificação desde 2016 e se destaca por atender às demandas da Indústria 4.0, englobando tecnologias como Inteligência Artificial, virtualização de dispositivos e Internet das Coisas (*Internet of Things* – IoT) (Pantoni et al., 2024), além de possibilitar a interoperabilidade entre diferentes fabricantes, de forma segura, flexível e aberta (Qamsane et al., 2022).

Blocos Funcionais (*Function Blocks* – FB) são elementos essenciais na construção de estratégias de controle em sistemas de automação industrial. Esses blocos encapsulam lógica de processamento, entradas, saídas e parâmetros de configuração, permitindo uma estrutura modular e reutilizável. No contexto do padrão O-PAS, além dos blocos funcionais padronizados, é possível criar Blocos Funcionais Definidos pelo Usuário (*User Defined Function Blocks* – UDFBs), que podem ser reutilizados em diferentes sistemas compatíveis, independentemente do fornecedor.

Como é comum em padrões ainda em desenvolvimento, a norma O-PAS apresenta lacunas em sua especificação. Visando contribuir para o preenchimento dessas lacunas, trabalhos anteriores se propuseram a transpor a especificação O-PAS em um mecanismo funcional capaz de criar UDFBs conforme a norma (Silveira; Pantoni, 2024), gerando os arquivos *Source Code AddData* e *NodeSet*.

Em continuidade ao projeto anterior, foi desenvolvido um mecanismo, chamado de UDFB *Engine*, capaz de executar os UDFBs, ou seja, receber os arquivos gerados no mecanismo anterior, interpretar a lógica, processar entradas e fornecer as saídas. O intuito final do mecanismo recentemente desenvolvido é possibilitar sua execução em plantas industriais dentro de dispositivos de controle e, caso haja defeitos em sua concepção, o impacto para a planta pode ser catastrófico (Myers, 2006).

Com o intuito de executar o mecanismo desenvolvido a fim de encontrar eventuais erros ou falhas (Myers, 2006), o presente projeto se propôs a realizar testes no mecanismo. Considerando que, mesmo em softwares pequenos, testar todas as possibilidades é inviável, este trabalho concentrou-se na realização de testes de caixa preta.

[1]Estudante do Curso de Engenharia Elétrica, IFSP - Sertãozinho/SP. silveira.s@aluno.ifsp.edu.br. Bolsa Captada Externamente (BCE) - APPDI - PD&I Nº 07/2024 - IFSP e Nova Smar S.A. https://orcid.org/0009-0005-6385-1739.
[2] Doutor. IFSP - Sertãozinho/SP. emossin@ifsp.edu.br. https://orcid.org/0000-0001-5144-518X.
[3] Doutor. IFSP - Sertãozinho/SP. rpantoni@ifsp.edu.br. https://orcid.org/0000-0001-8644-7118.

XI CONGRESSO DE INICIAÇÃO CIENTÍFICA DO IFSP ITAPETININGA
Itapetininga, 27, 28 e 29 de maio de 2025
Instituto Federal de Educação, Ciência e Tecnologia de São Paulo
Campus Itapetininga

Objetivo

O objetivo geral deste trabalho foi realizar testes de funcionalidade no mecanismo UDFB Engine de forma automática, utilizando o software SikuliX (2025).

Especificamente, buscou-se:

- Avaliar a robustez do UDFB por meio de testes de estresse em diferentes casos de teste;
- Avaliar a ocorrência de efeitos colaterais resultantes de modificações na lógica interna do UDFB, analisando se tais alterações comprometem a consistência da saída esperada do bloco.

Metodologia

Conforme mencionado nos objetivos, para a realização dos testes foi adotado o software SikuliX (2025), que utiliza reconhecimento de imagens por meio da plataforma OpenCV para identificar componentes em interfaces gráficas. Além disso, o SikuliX é capaz de reconhecer textos utilizando a tecnologia Tesseract e oferece suporte a diversas linguagens de programação, como Python, Ruby e JavaScript. Essas linguagens permitem a criação de *scripts* que simulam a interação humana com a interface gráfica a ser testada (SikuliX, 2025).

UDFBs são, em sua maioria, compostos por sinais de entrada e saída definidos pela norma O-PAS, além de parâmetros de configuração em conformidade com a IEC 61131-3 (nternational Electrotechnical Commission, 2013). Com o intuito de analisar a aplicabilidade do SikuliX no contexto do padrão O-PAS, foi desenvolvido um UDFB responsável por implementar a fórmula de Bhaskara, cuja lógica em texto estruturado (*Strucutred Text* - ST) é apresentada no código 1. Este bloco funcional, ao receber como entrada um conjunto de coeficientes reais, retorna como saída as raízes da equação do segundo grau correspondente.

Uma vez definida a lógica a ser testada, foram desenvolvidos *scripts* em Python no SikuliX. Foram considerados dois cenários de teste distintos. O primeiro consistiu na realização de um teste de estresse, com o objetivo de avaliar a robustez do UDFB. O segundo cenário teve como propósito identificar possíveis efeitos colaterais decorrentes de alterações na lógica interna do UDFB por parte do desenvolvedor, verificando se tais modificações impactaram negativamente o comportamento ou a saída esperada do bloco.

Como parte dos procedimentos metodológicos, o primeiro teste foi realizado com a submissão do UDFB a uma execução contínua pelo período de 6 horas, com o objetivo de avaliar sua robustez e estabilidade. Durante o teste, o SikuliX enviou ao bloco, como entrada, diferentes conjuntos de coeficientes inteiros, entre -10 e 10, e analisou suas saídas.

Código 1 - Lógica em st do bloco que realiza a operação de Bhaskara.

```
PROGRAM program0
VAR_INPUT
   input1 : REAL;
   input2 : REAL;
   input3 : REAL;
END_VAR

VAR_OUTPUT
   x1_real : REAL;
   x1_imag : REAL;
   x2_real : REAL;
   x2_imag : REAL;
END_VAR
```

```
...
ELSIF (input2 * input2 - 4.0 * input1 * input3)
= 0.0 THEN
   x1_real := -input2 / (2.0 * input1);
   x2_real := x1_real;
   x1_imag := 0.0;
   x2_imag := 0.0;

ELSE
```

Para a realização do segundo teste, foram utilizados como dados de entrada os valores apresentados na tabela 1 (seção de resultados), que também apresenta os respectivos valores esperados como saída. Inicialmente, o SikuliX executou uma batelada de testes com um UDFB cuja implementação da fórmula de Bhaskara estava correta. Em um segundo momento, com o objetivo de simular uma modificação na lógica do bloco e verificar possíveis efeitos colaterais, a expressão referente ao denominador da fórmula — originalmente representada por *2a* — foi intencionalmente alterada para *3a*. Em seguida, esse novo UDFB modificado foi submetido ao mesmo conjunto de testes através do SikuliX . Os resultados obtidos a partir dessa comparação são apresentados na próxima seção.

Resultados

O primeiro teste foi conduzido utilizando o bloco funcional que implementa a lógica da fórmula de Bhaskara, sendo executado continuamente durante um período de 6 horas. Para garantir a precisão dos testes, o *script* desenvolvido no SikuliX incluía uma validação dos valores lidos: caso a ferramenta não conseguisse reconhecer corretamente o resultado, a rodada era descartada e uma nova execução era iniciada. Dessa forma, apenas leituras confiáveis foram consideradas. Em todas as iterações válidas, o mecanismo processou corretamente os coeficientes fornecidos e retornou as raízes esperadas das equações, sem apresentar falhas ou inconsistências.

O segundo teste, que buscou simular uma modificação na lógica do bloco e verificar possíveis efeitos colaterais, foi realizado em duas etapas. Primeiramente, os testes foram realizados com a versão original do UDFB, que implementava corretamente a fórmula de Bhaskara. Neste caso, os resultados apresentaram valores plenamente coerentes com os valores esperados indicados na tabela 1, validando a precisão do bloco funcional. No entanto, após a modificação intencional da expressão *2a* para *3a*, os testes repetidos com o UDFB alterado revelaram discrepâncias sistemáticas nas saídas geradas. Com exceção de um caso, as raízes calculadas divergiram dos valores esperados, evidenciando que a alteração na lógica impactou diretamente o comportamento do bloco. Com o objetivo de ilustrar este cenário, tem-se a tabela 1. Esses resultados demonstram a eficácia do uso do SikuliX na detecção automatizada de efeitos colaterais provocados por modificações na lógica interna de UDFBs, reforçando seu potencial como ferramenta de apoio à verificação funcional em ambientes compatíveis com o padrão O-PAS. Nota-se que, no último exemplo

da tabela 1, os valores esperados e obtidos são os mesmos. Isso aconteceu pois os coeficientes b e c são zeros, o que faz com que as raízes do sistema sejam zero, inclusive ao alterar essa equação.

Tabela 1 - Coeficientes utilizados e resultados para o segundo cenário de teste.

a	b	c	Resultado esperado	Resultado obtido	
1	-5	6	{2,3}	{2, 1.33}	✗
1	0	-4	{2, -2}	{1.33, -1.33}	✗
1	-4	4	{2}	{1.33, 1.33}	✗
2	-5	2	{2, 0.5}	{1.33, 0.33}	✗
3	7	-6	{⅔, -3}	{0.44, -2}	✗
1	0	0	{0, 0}	{0, 0}	☑

Fonte: autoria própria

Conclusão

Durante a execução dos testes de caixa preta utilizando a ferramenta SikuliX, foi possível validar o comportamento da interface do software a partir da simulação de interações visuais, como cliques em botões, preenchimento de campos e verificação de mensagens exibidas na tela. A maioria dos casos de teste apresentou resultados satisfatórios, indicando que o sistema responde corretamente às entradas fornecidas pelo usuário.

Entretanto, algumas falhas observadas se devem à sensibilidade do SikuliX a pequenas variações na interface gráfica, como mudanças de resolução, contraste ou posicionamento de elementos. Somado a isso, o fato do SikuliX utilizar algoritmos de reconhecimento óptico de caracteres (*Optical Character Recognition* - OCR) para identificar textos, pode gerar leituras imprecisas, ocasionando erros de interpretação nos testes. Dessa forma, é essencial que os *scripts* sejam preparados para lidar com possíveis variações e que se tenha consciência de que um erro apontado pode não representar, de fato, uma falha do sistema, mas sim uma limitação na leitura visual feita pela ferramenta. Essas observações reforçam a necessidade de se realizar testes em ambientes controlados para garantir a confiabilidade dos resultados.

De modo geral, os testes demonstraram a efetividade da abordagem baseada em imagens para validar funcionalidades do sistema sob a perspectiva do usuário final, reforçando a aplicabilidade do SikuliX como ferramenta de apoio na automação de testes funcionais.

Ao alterar a lógica para verificar se os resultados deixariam de estar corretos, o *script* conseguiu, com sucesso, identificar todos os casos incorretos, com exceção de um. Em um

cenário prático, é importante estar ciente da lógica implementada, pois, mesmo que alterações sejam aplicadas, alguns resultados podem aparentar estar corretos, mas isso não necessariamente os valida. Dessa maneira, a análise dos testes deve considerar tanto o comportamento da ferramenta quanto o entendimento criterioso do domínio da aplicação testada.

Agradecimentos

Os autores agradecem à empresa Nova Smar pela colaboração e apoio no desenvolvimento desse projeto (APPDI - PD&I Nº 07/2024 - IFSP e Nova Smar S/A).

Referências

International Electrotechnical Commission. *IEC 61131-3: Industrial automation systems - Programmable controllers - Part 3: Programming languages*. [S.l.], 2013. Disponível em: https://webstore.iec.ch/publication/6182.

MYERS, Glenford J. *The art of software testing*. Hoboken: John Wiley & Sons, 2006.

PANTONI, R. P. et al. *Design and implementation of o-pas user-defined function blocks*. Journal of Electrical Systems and Information Technology, v. 11, p. 55, 2024. Disponível em: https://link.springer.com/article/10.1186/s43067-024-00183-9.

QAMSANE, Y. et al. *Open process automation- and digital twin-based performance monitoring of a process manufacturing system*. IEEE Access, v. 10, p. 60823–60835, 2022.

SIKULIX. Sikulix.com. Disponível em: http://www.sikulix.com/. Acesso em: 26 abr. 2025.

SILVEIRA, G. S.; PANTONI, R. P. *Mecanismo de implementação de blocos funcionais o-pas definidos pelo usuário*. In: 15° Congresso de Inovação, Ciência e Tecnologia do IFSP, 2024. São Paulo: Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, 2024.