57º SBPO
SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
Gramado • RS
DE 5 A 9 DE OUTUBRO DE 2025

TADEM: Um Framework de Transformação de Modelos Ptolemy II para Simulink

Israel Lima Dias
Universidade Federal Rural do Semi-Árido
islimadias@gmail.com

Lenardo Chaves e Silva
Universidade Federal Rural do Semi-Árido
lenardo@ufersa.edu.br

Sebastião Emidio Alves Filho
Universidade do Estado do Rio Grande do Norte
sebastiaoalves@uern.br

RESUMO

Uma das principais técnicas utilizadas para o projeto de sistemas físico-cibernéticos é a construção de modelos de simulação. Esses modelos precisam ser testados em diferentes ferramentas para serem validados. Devido às mudanças no estilo de desenvolvimento e nas metodologias utilizadas, construir o mesmo modelo para diferentes ferramentas torna-se dispendioso em termos de tempo. A Transformação de Modelos facilita essa tarefa e torna o desenvolvimento de *software* mais eficiente. Desta forma, este trabalho propõe o TADEM, um *framework* que contribui com um mapeamento aprimorado para a transformação semiautomática de modelos. Para sua validação, foi realizado um experimento de mecanismo de caso único no qual os componentes de um modelo clínico criado no Ptolemy II foram mapeados semiautomaticamente para componentes com funcionalidades semelhantes do Simulink. A aplicação do *framework* permitiu reduzir o tempo na transformação do modelo, considerando o trabalho de refazê-lo manualmente para ser executado na segunda ferramenta.

PALAVRAS CHAVE. Sistemas Físico-Cibernéticos, Modelos de Simulação, Transformação de Modelo.
Tópicos. SIM - Simulação. OA – Outras aplicações em PO.

ABSTRACT

One of the main techniques used for the design of cyber-physical systems is the construction of simulation models. These models need to be tested on different tools to be validated. Due to changes in the development style and methodologies used, building the same model for different tools becomes costly in terms of time. Model Transformation facilitates this task and makes software development more efficient. In this way, this work proposes TADEM, a framework that contributes with an improved mapping for the semi-automatic transformation of models. For its validation, a single-case engine experiment was carried out in which the components of a clinical model created in Ptolemy II were semi-automatically mapped to components with similar functionalities of Simulink. The application of the framework allowed to reduce the model transformation time, considering the work of manually redoing it to be executed in the second tool.

KEYWORDS. Cyber-Physical Systems. Simulation Models. Model Transformation.
Paper topics. SIM - Simulation. OA - Other applications in PO.

DOI: 10.59254/sbpo-2025-212143
https://proceedings.science/p/212143?lang=pt-br

1. Introdução

Sistemas Físico-Cibernéticos (SFCs) são sistemas computacionais e físicos integrados que podem interagir com os seres humanos através de muitas modalidades [Carreira et al., 2020]. Por se tratarem de sistemas críticos, onde falhas podem significar perigos à vida humana, devem ser cuidadosamente projetados, desenvolvidos e validados para garantir confiabilidade e robustez.

Uma das principais técnicas utilizadas para o projeto de sistemas críticos é a construção de modelos [Klikovits e Buchs, 2021]. Para que esses modelos sejam validados, especialmente quando tratam-se de modelos de simulação, é importante testá-los em diferentes ferramentas de modelagem. No entanto, as ferramentas de modelagem existentes são muito especializadas, o que dificulta a transformação de um modelo construído em uma ferramenta para outra.

Com este trabalho, busca-se facilitar o projeto de SFCs com uma solução para a transformação semiautomática de modelos entre duas ferramentas distintas bem conhecidas (Ptolemy II® e Simulink®). Para alcançar este objetivo, foi adotada a metodologia de pesquisa *Design Science*, que orienta a construção e avaliação de artefatos tecnológicos como forma de gerar conhecimento aplicável à solução de problemas práticos [Wieringa, 2014]. Conforme o contexto levantado e as recomendações da metodologia, foi abordada a seguinte questão de pesquisa: *"Como realizar o mapeamento semiautomático de modelos de SFCs entre diferentes ferramentas de modelagem computacional?"*

Para tratar a questão, foi proposto um *framework* composto por componentes mapeados e uma interface gráfica. Os componentes foram selecionados de dois modelos de simulação de um cenário clínico, construídos manualmente no Ptolemy II® e no Simulink®. A validação do *framework* ocorreu através de um experimento de mecanismo de caso único, onde os componentes do modelo no Ptolemy II® foram semiautomaticamente mapeados para componentes similares no Simulink®, reduzindo o tempo do processo em comparação ao método totalmente manual.

Após esta introdução, o artigo se organiza da seguinte forma: **Seção 2** apresenta os conceitos que fundamentam o *framework* e as tecnologias utilizadas. **Seção 3** descreve trabalhos relacionados. **Seção 4** detalha o cenário de aplicação, a arquitetura e a interface gráfica do *framework*. **Seção 5** discute a validação do *framework*, ameaças à sua validade e limitações. Por fim, a **Seção 6** conclui o artigo, destacando suas contribuições e futuras etapas da pesquisa.

2. Fundamentação Teórica

2.1. Sistemas Físico-Cibernéticos

O termo Sistema Físico-Cibernético (SFC) tem sido usado desde os anos 70 com o surgimento dos microprocessadores. Porém, apenas em 2006 foi utilizado para descrever sistemas que comunicam os mundos físico e digital [Liu et al., 2020]. A academia, a indústria e até mesmo vários governos têm se interessado pelos SFCs pelo seu potencial impacto significativo na economia e na sociedade [Bakirtzis et al., 2022].

Um SFC é representado por duas partes cruciais, sendo a primeira uma conectividade que assegura a comunicação de dados em tempo real do mundo físico, ao mesmo tempo em que há um *feedback* de informações do sistema cibernético, e a segunda, um gerenciamento otimizado de dados com uma análise computacional que constrói o sistema cibernético [Lee, 2015].

Os estudos sobre SFC são divididos em dois campos: os fundamentos teóricos e as aplicações, que são construídas para facilitar a implementação de sistemas de grande escala, com a possibilidade de melhorar autonomia, funcionalidade, adaptabilidade, confiabilidade, usabilidade e segurança [Bakirtzis et al., 2022].

Normalmente, não é possível garantir que todos os erros existentes em um sistema serão detectados após sua fase de testes. Assim, outras técnicas podem ser utilizadas em conjunto para reduzir falhas durante o seu processo de desenvolvimento. Uma técnica bem conhecida é a verificação

DOI: 10.59254/sbpo-2025-212143
https://proceedings.science/p/212143?lang=pt-br

de modelos, que permite explorar todo o espaço de estados do sistema a fim de encontrar o defeito ou falha ainda na fase de projeto [Sirjani et al., 2021].

2.2. Desenvolvimento Baseado em Modelos

Os modelos de *software* são componentes cruciais para uma simulação, pois permitem executar testes para cada componente, ou para um sistema como um todo [Klikovits e Buchs, 2021]. Dessa forma, sua utilização pode reduzir o custo de investimento em testes "em campo" desse sistema. Um "modelo", de maneira geral, é uma entidade conceitual ou física que se assemelha, descreve, prevê ou veicula informações sobre o comportamento de algum processo ou sistema com o benefício de poder explorar seu comportamento de uma forma econômica e segura.

O Desenvolvimento Orientado a Modelo (DOM) considera, dentro do desenvolvimento de software, a modelagem do problema mais importante que o desenvolvimento do código-fonte [Sirjani et al., 2021]. Com isto, a atenção fica mais concentrada em construir uma aplicação final que funcione de acordo com os requisitos especificados no modelo do *software*.

Para a construção de um SFC confiável, eficaz e seguro é imprescindível a inserção da modelagem no centro do processo de desenvolvimento do software [Carreira et al., 2020]. Assim, por meio de simulações ou diagramas, o comportamento de um sistema pode ser concebido de acordo com a entrada de dados, processamento e consequentemente uma saída. É importante que um modelo seja testado em ferramentas distintas, pois elas possuem diferentes funcionalidades e nuances que podem ser úteis para o projeto de SFCs. No entanto, o trabalho de refazer um modelo em outra ferramenta é custoso em relação ao tempo e pode gerar diferenças conceituais.

2.3. Ferramentas de Modelagem

Neste trabalho é proposto um *framework* para a transformação semiautomática de modelos entre Ptolemy II® [1] e Simulink® [2]. Ambos os ambientes de modelagem se caracterizam como um conjunto de ferramentas utilizadas para simular sistemas de controle e também para gerar códigos.

Ptolemy II® é uma ferramenta de código aberto que utiliza do conceito de projeto orientado a atores. Nela, os componentes são executados simultaneamente e se comunicam por portas interconectadas, com a finalidade de construir sistemas informatizados experimentais [Xiong et al., 2005].

Simulink® é uma extensão do Matlab®, que viabiliza a construção de modelos computacionais dinâmicos utilizando notação de diagrama de blocos. Ela possibilita uma fácil modelagem de sistemas não lineares complexos. O Simulink® permite demonstrar, com animações gráficas, o comportamento da simulação, acrescentando significativamente o entendimento do comportamento do sistema [Dabney e Harman, 2004].

2.4. Transformação de Modelos

A transformação de modelos é um importante quesito dentro da Engenharia Dirigida por Modelos (EDM), que visa tornar mais eficiente o processo de desenvolvimento de *software*, incluindo atualizações e manutenção. Seu ponto central é prover artefatos primários que possam guiar todo o processo [Drave et al., 2022].

Para uma transformação ser efetivada, deve-se definir quais elementos serão transformados e qual o mapeamento entre os modelos. A partir dos elementos do modelo fonte, o mapeamento indicará quais elementos serão criados no modelo destino [Liu et al., 2020]. Na transformação, os elementos dos respectivos modelos fonte e destino são sub-tipos de modelos genéricos existentes e a transformação é especificada com base nos tipos dos modelos genéricos.

O **Modelo Específico de Plataforma (MEP)** é um modelo genérico de uma tecnologia, ou sistema de organização, que é intrinsecamente dependente de uma tecnologia específica. MEPs

[1] Disponível em: `https://ptolemy.berkeley.edu/PtolemyII/index.html`
[2] Disponível em: `https://mathworks.com/products/simulink.html`

DOI: 10.59254/sbpo-2025-212143
https://proceedings.science/p/212143?lang=pt-br

têm sido indispensáveis na implantação de sistemas tecnicamente complexos [Xiao e Fan, 2012]. Isso se deve ao fato de que qualquer informação, dado, conhecimento e funcionalidade devem ser processados e comunicados de maneira consistente com as tecnologias específicas que estão sendo usadas. Assim, um MEP é um modelo que representa o sistema em relação às escolhas implementadas em uma determinada plataforma [Bévan et al., 2012].

O **Modelo Independente de Plataforma (MIP)**, também chamado de modelo genérico na Engenharia de Software, é um modelo simples e reduzido de um sistema de software ou aplicativo. Um MIP não depende de nenhuma plataforma tecnológica específica para apoiar os processos e não deve depender de nenhum componente do sistema de origem [Kherraf et al., 2008].

3. Trabalhos Relacionados

A transformação de modelos entre ferramentas distintas de modelagem e simulação, como Ptolemy II® e Simulink®, continua sendo um desafio significativo na engenharia de sistemas ciber-físicos (CPS). Estudos recentes têm abordado esses desafios sob diferentes perspectivas:

- [Zhan et al., 2024]: Propuseram uma ferramenta que transforma modelos do Simulink (Stateflow) e AADL em processos HCSP, permitindo simulação, verificação formal e geração de código para sistemas ciber-físicos.
- [Lehner et al., 2025]: Realizaram um mapeamento sistemático sobre a aplicação de técnicas de automação EDM, como transformação de modelos e geração de código, no contexto de gêmeos digitais. O estudo destaca a aplicabilidade das técnicas EDM em diversos domínios, incluindo a área médica, especialmente na modelagem de sistemas complexos como os SFCs.
- [Xinquan et al., 2023] : Propuseram uma abordagem de transformação de modelos que converte modelos híbridos desenvolvidos em SysML para o framework DEVS, permitindo a simulação e validação de sistemas ciber-físicos complexos sem a necessidade de conhecimento aprofundado da teoria DEVS.
- [Melo et al., 2015]: Descreveram a utilização dos ambientes Ptolemy II® e Simulink® como ferramentas de aprendizagem na implementação de modelos de simulação, para que sejam discutidas e avaliadas por estudantes de engenharia, representando o sistema de controle *multi-loop* descentralizado, usado para controlar processo em sistemas que precisam de mais de uma malha de controle dentro de um único sistema.

Esses estudos reforçam a complexidade envolvida na transformação de modelos entre ferramentas heterogêneas e a necessidade de frameworks especializados que considerem as particularidades semânticas e operacionais de cada ambiente. Diante disso, este trabalho propõe um *framework* semiautomático voltado especificamente para a transformação de modelos entre Ptolemy II® e Simulink®, selecionadas por sua ampla adoção acadêmica e industrial na modelagem e simulação de sistemas físico-cibernéticos.

4. *Framework* para TrAnsformação DE Modelos (TADEM)

Com o objetivo de reduzir tempo e esforço na transformação de modelos de SFCs, minimizando divergências entre eles, este trabalho propõe um *framework* denominado TADEM para TrAnsformação DE Modelos. O cenário de aplicação do *framework* é apresentado na **Figura 1**.

O processo é dividido em duas fases: a fase de análise e a fase de síntese. O ponto inicial corresponde aos dados obtidos através de um arquivo, cujo o conteúdo é composto pelo modelo de entrada que está configurado de acordo com as particularidades referentes ao *software* de modelagem primário. Após todas as etapas, é concebido um arquivo de saída contendo o modelo configurado conforme a ferramenta de modelagem destino.

DOI: 10.59254/sbpo-2025-212143
https://proceedings.science/p/212143?lang=pt-br

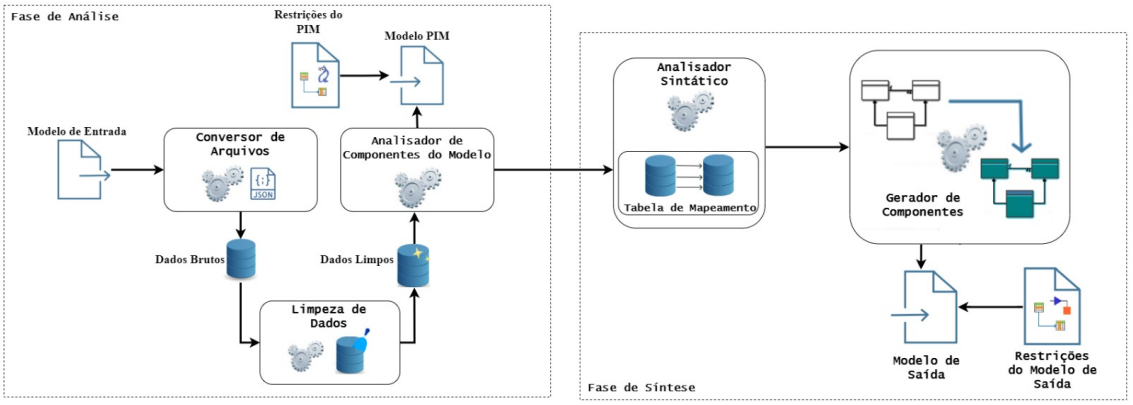


Figura 1: Cenário de aplicação do *framework*. FONTE: Autoria Própria (2023).

4.1. Fase de Análise

O ponto central desta fase é o modelo de entrada, base para todas as etapas. O conteúdo desse modelo é analisado, extraindo strings, caracteres e valores. Na fase de análise, o conteúdo é lido e analisado. A seguir, são descritos os componentes do *framework* envolvidos nesta fase.

4.1.1. Modelo de Entrada

As ferramentas de modelagem de *software* salvam seus modelos na máquina do usuário. Esses arquivos devem estar em formato aberto para permitir visualização e edição por outros *softwares*. O arquivo deve disponibilizar dados do modelo (componentes, nomes de variáveis, valores) para extração e leitura pelo *framework*. As regras de composição do arquivo precisam estar em texto com significado em uma linguagem formal, permitindo abstrair sua semântica.

4.1.2. Análise de Componentes e Atributos

Este processo permite ao *framework* remover componentes redundantes e indesejados. A verificação de caracteres especiais, símbolos e letras é crucial, pois seu uso pode causar erros de codificação e dificultar a implementação. O analisador examina palavras-chave lexicais e estruturais no código do modelo para corresponder componentes aos *tokens* corretos.

4.1.3. Construção do Modelo Independente de Plataforma

A transformação do MEP em MIP é guiada por regras de transformação, diretrizes que facilitam a tradução entre formatos. É crucial considerar todas as restrições de configuração do modelo independente, incluindo elementos hierárquicos e meta-dados como títulos e *charsets*.

Utiliza-se uma regra comum de transformar um modelo de origem em modelos de destino que possuem restrições dependentes. Essas cláusulas condicionais permitem expressar um nível mais alto de dependência nos modelos de saída, sem reduzir sua complexidade geral. O MIP ainda teria características e nomenclaturas referentes ao modelo de origem.

O MIP é útil para futuras transformações visando atualizar ou construir novos modelos conceituais base. Isso reduz o tempo e esforço de desenvolvimento para as plataformas do *framework* e outras ferramentas de modelagem, bastando desenvolver a tabela de mapeamento.

4.2. Fase de Síntese

Na fase de síntese do *framework* é criado um grupo de representações, que podem conter qualquer tipo de informação necessária para mapear as saídas finais para os componentes do modelo. A questão-chave a ser observada é que essas representações normalmente são baseadas em *tokens*, que terão que aderir ao formato especificado pela plataforma destino.

Na fase de síntese do *framework*, cria-se um grupo de representações para mapear as saídas finais aos componentes do modelo. Essas representações, baseadas em *tokens*, devem aderir ao formato da plataforma de destino.

DOI: 10.59254/sbpo-2025-212143
https://proceedings.science/p/212143?lang=pt-br

4.2.1. Tabela de Mapeamento

A tabela de mapeamento relaciona atores do Ptolemy II a blocos (ou combinações) no Simulink. O *framework* consulta essa tabela durante a síntese para instanciar o conjunto-alvo, preservando função/comportamento. Quando não há equivalência 1:1, o mapeamento ocorre por *padrões* (grupos de blocos) previamente especificados.

4.2.2. Gerador de Componentes

Ao identificar um componente de origem mapeável, o TADEM cria o(s) bloco(s) equivalente(s) e suas conexões, respeitando dependências (ordem e hierarquia). A alocação é incremental sobre um esqueleto MDL padrão do Simulink.

4.3. Construção de um Modelo Dependente de Plataforma

Com todos os componentes alocados, estrutura-se o arquivo necessário para a execução do modelo na ferramenta destino, posicionando os componentes e identificando seus comportamentos. Modelos criados por uma ferramenta compartilham dados comuns, que não se referem aos componentes essenciais, mas permitem à ferramenta obter informações sobre a máquina onde está instalada. Assim, essas informações comuns podem ser previamente atribuídas ao modelo de saída no arquivo gerado.

4.4. Arquitetura do *Framework* TADEM

Para facilitar o uso do *framework*, foi desenvolvida uma interface gráfica para transformar modelos do Ptolemy II® para o Simulink®. O usuário pode fornecer apenas o arquivo de entrada. Se o modelo contiver componentes não mapeados, será necessário estender a tabela de mapeamento ou complementar o modelo manualmente antes de usar a interface.

O *framework*, ilustrado na **Figura 2**, possui na camada do utilizador dois componentes: a interface gráfica, que carrega o arquivo do modelo e verifica sua compatibilidade, e a atualização da solicitação de transformação, informando o caminho do arquivo no sistema.

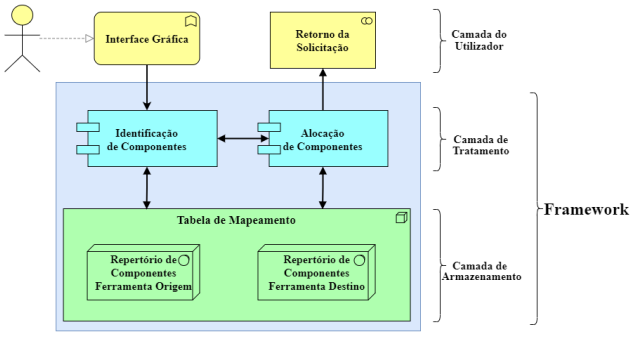


Figura 2: Arquitetura do *framework* TADEM. FONTE: Autoria Própria (2023).

À medida que os componentes são identificados no modelo, um novo conjunto é listado conforme a tabela de mapeamento, que armazena as relações entre os componentes da ferramenta de origem e os da ferramenta de destino.

O *framework* possui duas camadas para processar os dados. A primeira identifica os componentes do modelo e na segunda camada verifica sua presença no repositório da ferramenta de origem. Se um componente for encontrado e correspondido na tabela de mapeamento, ele é alocado em um novo arquivo no repositório da ferramenta de destino, seguindo a função associada entre as ferramentas. Uma tabela de mapeamento lista e aloca novos componentes conforme são identificados na coluna da ferramenta de destino.

DOI: 10.59254/sbpo-2025-212143
https://proceedings.science/p/212143?lang=pt-br

4.5. Utilização do *Framework*

4.5.1. Interface Gráfica do Usuário

Diversas linguagens de programação e ferramentas possibilitam a implementação do TADEM. A linguagem de programação Python foi escolhida para o desenvolvimento. A parte gráfica da aplicação do *framework* (**Figura 3**) foi desenvolvida utilizando o pacote tkinter[3] ("interface Tk"), que trata-se da biblioteca padrão do Python para interface gráfica de usuário. O código da aplicação construída encontra-se disponível online[4].



Figura 3: Tela Principal da Aplicação do *Framework* TADEM. FONTE: Autoria Própria (2023).

O "Botão 1" abre uma janela para seleção de arquivos. O "Botão 2" inicia a transformação para a ferramenta de destino. Arquivos auxiliares são gerados para controle e análise de dados. O "Botão 3" elimina esses arquivos após a transformação. O "Botão 4" encerra a aplicação, e o "Botão 5" abre o diretório onde o MIP está armazenado.

4.5.2. Cenário de Uso

Após analisar o código gerado pelo Ptolemy II® e identificar a organização dos componentes em XML, decidiu-se converter o conteúdo para JSON, um formato leve e de fácil leitura e escrita, para facilitar a seleção dos componentes úteis.

O módulo de processamento de XML em Python facilitou a conversão sem alterar o conteúdo. Foi necessária uma análise paralela do modelo no Ptolemy II® e do código JSON gerado para selecionar termos-chave que identificam os componentes do modelo. Esses termos, nomes de tipos de atores na ferramenta, resultaram em um *"array"* final.

Assim, é possível realizar uma listagem de nomes de componentes comparando cada elemento do arquivo JSON a ser transformado com os termos chave encontrados no Ptolemy II®. Como demonstrado no fluxograma da **Figura 4**, um elemento chave (variável **ele**) é buscado no arquivo por vez, onde todos os nomes e valores do arquivo JSON são analisados e comparados com os termos chave. Caso o componente seja identificado como um componente chave do modelo, seu nome e valor é armazenado em uma nova estrutura JSON. Essa função é finalizada após a análise do último elemento.

4.5.3. Listagem e Identificação de Componentes

Para identificar os componentes descartados e analisados, cada linha do arquivo foi inspecionada. Os componentes relevantes foram alocados em memória e adicionados a um novo arquivo JSON com o conteúdo selecionado. Cada valor refere-se a um componente, passando pelo nome da variável, seu tipo e os nós listados durante a varredura.

[3]Disponível em: docs.python.org/3/library/tk.html
[4]Disponível em: dropbox.com/scl/fi/5d9l0zjmrln1p89up8bwz/Json-m2m.zip

DOI: 10.59254/sbpo-2025-212143
https://proceedings.science/p/212143?lang=pt-br

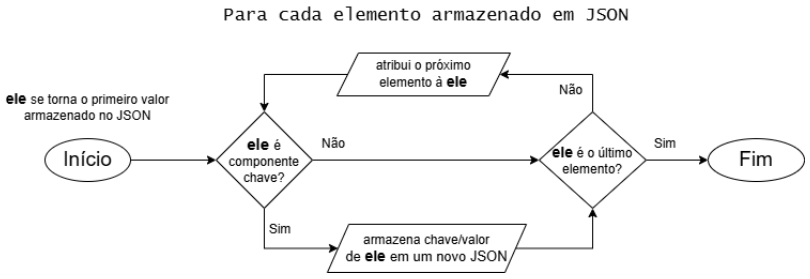


Figura 4: Busca por Componentes Chaves no Arquivo JSON. FONTE: Autoria Própria (2023).

4.5.4. Criação do MIP

Para alocar os componentes selecionados em uma estrutura independente de plataforma, foi necessário primeiro identificar essa estrutura. A maioria dos trabalhos na literatura utilizava UML para MIP, mas, conforme demonstrado por [Kobeissy et al., 2007], é possível utilizar o formato ontológico. A principal vantagem de usar uma ontologia formal, em vez de diagramas UML, é a verificação automática da consistência do modelo.

O MIP foi especificado em OWL, a linguagem recomendada pelo *World Wide Web Consortium* (W3C) para ontologias formais[5], usando a sintaxe *Terse RDF Triple Language* (Turtle). Os componentes seguem a nomenclatura do Ptolemy II® e são alocados por uma função recursiva em "classes" e "indivíduos". Em ontologias, as classes representam conjuntos de indivíduos, enquanto os indivíduos são instâncias dessas classes [Euzenat et al., 2007]. Após todos os componentes serem alocados em uma variável, uma segunda função armazena o conteúdo em um arquivo OWL.

4.5.5. Mapeamento de elementos do modelo inicial para elementos dos modelo final

Devido à grande quantidade de componentes, a tabela de mapeamento foi limitada aos componentes dos modelos de cenários clínicos de [Silva et al., 2015], estudados e usados para concepção e validação do *framework* TADEM.

Para desenvolver o mecanismo, foi estabelecido um conjunto de regras que define a lógica do mapeamento entre as ferramentas. Apenas os componentes identificados na tabela de mapeamento foram instanciados; os não mapeados não são instanciados, impossibilitando seu mapeamento.

Durante o desenvolvimento, foi identificado que algumas combinações de mapeamentos exigiam o agrupamento de componentes. Para replicar o comportamento de um componente da ferramenta de origem, era necessário integrar um subconjunto de componentes na ferramenta de destino e vice-versa. Essas combinações formaram modelos de mapeamento.

A tabela de mapeamento contém elementos do modelo de entrada e seus correspondentes na ferramenta de destino. Por exemplo, o elemento *"Scale"* do Ptolemy II® corresponde aos componentes *"Constant"* e *"Product"* no Simulink®. Outros componentes e agrupamentos podem ser visualizados em um documento disponível online[6].

4.6. Criação do MEP

Com os componentes identificados, começa-se o processo de alocação conforme mostrado na **Figura 5**. Consultando a tabela de mapeamento, é possível identificar o componente ou conjunto de componentes com estrutura e comportamento equivalentes no Simulink®, para serem adicionados ao modelo de destino. O nó na *string* do componente indica a ordem hierárquica para sua alocação no arquivo MDL do Simulink®. A colocação desses componentes mapeados no modelo de saída é

[5]Disponível em: w3.org/TR/owl2-overview/

[6]Disponível em: dropbox.com/scl/fi/5pnwyfssowmm2cq9ua6sn/TABELA-DE-COMPONENTES-MAPEADOS.pdf

DOI: 10.59254/sbpo-2025-212143
https://proceedings.science/p/212143?lang=pt-br

feita de forma incremental, sem levar em conta o posicionamento dos componentes no modelo de entrada.

A estrutura base do arquivo MDL, contendo as configurações padrão para execução no Simulink®, permanece fixa. Esse conteúdo é incrementado à medida que os componentes são identificados e comparados na tabela de mapeamento.

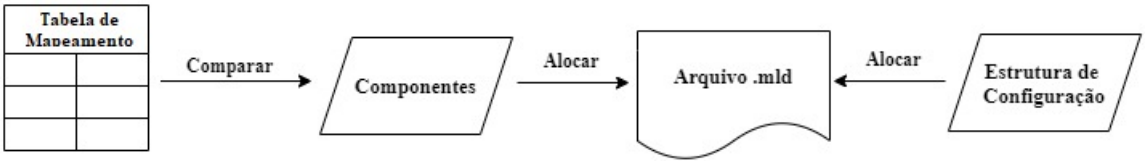


Figura 5: Sequência para a Alocação dos Componentes. FONTE: Autoria Própria (2023).

5. Validação do *Framework*

Para verificar a validade e a funcionalidade do *framework* proposto, foram adotadas etapas compatíveis com a metodologia *Design Science*. As etapas seguidas incluíram: a construção de um protótipo; a aplicação de um modelo representativo no contexto médico físico-cibernético como artefato de estudo; e a execução controlada do artefato em um cenário de teste, com o intuito de observar suas respostas.

Durante os testes, foi realizada a análise do modelo de saída, buscando compreender como ele foi gerado a partir dos mecanismos internos do *framework* TADEM. Essa observação permitiu avaliar o processo de mapeamento entre os modelos, identificar eventuais necessidades de complementação no modelo transformado e, por fim, examinar o comportamento funcional do modelo resultante.

5.1. Análise do Problema e Projeto de Validação

De maneira restrita, o objeto de estudo trata-se de um modelo de bomba de insulina, projetado no Ptolemy II®, interagindo com o *framework* TADEM. A população da pesquisa foi definida como o conjunto de todos os componentes do modelo. Este cenário foi escolhido por já estar totalmente desenvolvido e disponível a partir do trabalho de [SILVA et al., 2015], além de ter grande importância, pois simula o comportamento da bomba de insulina quando usado por pessoas com Diabetes. O Diabetes Mellitus é uma doença crônica que afeta aproximadamente 463 milhões de pessoas em todo o mundo, segundo a Organização Mundial da Saúde [WHO, 2020].

Um experimento de mecanismo de caso único foi realizado para avaliar a implementação e investigar o problema com o intuito de receber informações sobre a aplicação do *framework* TADEM. Para realizar o experimento, primeiro foi necessário analisar a estrutura organizacional dos modelos gerados pelas ferramentas Ptolemy II® e Simulink®.

5.2. Execução do Experimento

Após a execução do *framework*, selecionando como modelo de entrada o arquivo XML do modelo de cenário clínico, foi possível gerar um modelo MDL[7] contendo algumas restrições. Alguns componentes não puderam ser mapeados, como a faixa de glicose que é disparada por um gatilho de acordo com o *"DiscretClock"* e o *"CurrentTime"*, já que eles seriam mapeados para o *"SignalBuilder"* do Simulink®. Esse componente é composto por uma série de caracteres de comprimento fixo, não podendo ser traduzidos por outra ferramenta. Esse empecilho pode ser configurado manualmente pelo projetista no Simulink®, tornando possível a execução do modelo.

Outra limitação do TADEM é que os componentes do modelo destino gerado também não são apresentados de forma esteticamente agradável, mas podem ser configurados apenas arrastando-os para posições mais adequadas, o que não interfere na execução do modelo pelo Simulink®.

[7]Disponível em: dropbox.com/scl/fo/dl0uqheq12h4a04hqge1n/AEyExIdzrB6B5-povQUYdxI

DOI: 10.59254/sbpo-2025-212143
https://proceedings.science/p/212143?lang=pt-br

5.3. Análise e Resultados Quantitativos

O modelo transformado foi executado com poucas intervenções, corroborando a eficácia do TADEM: diferenças de notação/posição entre as ferramentas não impediram o mapeamento, embora elementos cuja semântica depende de *layout* (p.ex., máquinas de estados) e fontes temporizadas (*SignalBuilder*, derivadas de *DiscreteClock*/*CurrentTime*) tenham exigido complementos manuais; casos sem equivalente direto entre Ptolemy II® e Simulink® foram resolvidos por composição de blocos ou ajuste posterior (lista detalhada no material suplementar[8]). Quanto às métricas, a cobertura foi de 82,2% mapeados e 17,8% não mapeados; as intervenções *mínimas* pós-TADEM foram de $\geq 17{,}8$ por 100 componentes (limite inferior que assume ao menos uma intervenção por componente não mapeado; *não inclui conexões*); e o tempo não é reportado por depender da experiência do modelador em Ptolemy II®/Simulink®.

5.4. Limitações

Na forma atual, o *framework* TADEM está restrito a duas ferramentas de modelagem maduras e em evolução contínua. Essa condição pode acarretar problemas de compatibilidade entre versões distintas das empregadas neste estudo, exigindo atualização recorrente do próprio *framework*. Em particular: (i) **dependência de versões** — alterações nas bibliotecas e na semântica de blocos podem invalidar mapeamentos existentes; (ii) **elementos com semântica dependente de *layout* e séries temporais** — não são plenamente automatizáveis; (iii) **cobertura** — a tabela de mapeamento reflete o conjunto de componentes do estudo clínico e, fora desse escopo, a transformação tende a ser parcial, requerendo complementação manual.

Para os componentes não mapeados e causas, além do caso *SignalBuilder* citado (derivado de *DiscreteClock* e *CurrentTime*, cuja especificação depende de séries temporais codificadas), identificamos quatro categorias recorrentes:

1. **Fontes/traçados temporizados**: sinais definidos por amostras/curvas (*SignalBuilder*) exigem pontos e perfis não inferíveis do arquivo de origem; solução: parametrização manual (*.mldatx*) ou script gerador.

2. **Máquinas de estados/posicionamento sensível**: elementos cujo significado depende de topologia/layout (p.ex., *charts* de estado) requerem reespecificação no destino.

3. **Blocos proprietários ou parametrização opaca**: componentes cujo comportamento depende de máscaras/códigos proprietários ou parâmetros não textuais.

4. **Atores específicos de domínio sem equivalente direto**: demandam composições de blocos ou extensões na tabela de mapeamento.

Esta avaliação baseia-se em um único estudo de caso (bomba de insulina), o que limita a validade externa. Diferenças de estilo de modelagem, presença de máquinas de estados e perfis dinâmicos (contínuo/discreto) podem afetar a aplicabilidade das regras de mapeamento. Como mitigação, reportamos métricas operacionais e detalhamos os componentes não mapeados.

6. Conclusão

Neste trabalho, foi discutida a proposta inicial de uma solução para a transformação semiautomática de componentes presentes em um modelo de simulação de cenário clínico, gerado a partir da ferramenta Ptolemy II®, para os componentes do Simulink®.

Pôde-se mostrar que a aplicação do *framework* proposto realizou a transformação do modelo selecionado de forma semiautomática, pois houve necessidade de algum ajuste no modelo

[8]Disponível em: dropbox.com/scl/fi/TABELA-DE-COMPONENTES-N-O-MAPEADOS.pdf

DOI: 10.59254/sbpo-2025-212143
https://proceedings.science/p/212143?lang=pt-br

transformado para possibilitar a execução na ferramenta destino. Além disso, o *framework* fornece uma base estrutural que pode servir de solução como transformação de modelos para diferentes cenários que apresentam os mesmos componentes.

Os seguintes tópicos foram considerados para trabalhos futuros: (1) **Ontologia como MIP executável**: migrar a tabela de mapeamento para OWL, com classes `Actor`, `Block`, `Port`, `Connection` e propriedades `mapsTo`/`composedOf`; usar raciocínio DL (*Description Logics*) para checar consistência (p.ex., tipos de porta, cardinalidade) e SPARQL para gerar o *template* MDL. Isso facilita incluir novos componentes por axiomas, sem alterar código. (2) **Versionamento e compatibilidade**: anotar mapeamentos com metadados de versão (`since`, `deprecated`) para suportar evoluções de Ptolemy II/Simulink. (3) **Auto-layout programático**: integrar o pós-processador de *layout* citado. (4) **Avaliação externa**: replicar o experimento em ao menos dois modelos adicionais (um com dinâmica contínua predominante; outro com máquina de estados), reportando métricas da **Seção 5.3**. (5) **Posicionamento automático**: adotar um pós-processador de *layout* no modelo de saída: (i) derivar um grafo acíclico a partir do MIP; (ii) aplicar ordenação topológica e empacotamento ortogonal; (iii) escrever coordenadas (`Position`) dos blocos e rotas de `add_line` no MDL. Essa etapa reduz o custo de organização visual e facilita inspeção do modelo..

Referências

Bakirtzis, G., Sherburne, T., Adams, S., Horowitz, B. M., Beling, P. A., e Fleming, C. H. (2022). An ontological metamodel for cyber-physical system safety, security, and resilience coengineering. *Software and Systems Modeling*, 21(1):113–137.

Bévan, R., Berruet, P., de Lamotte, F., Adam, M., Cardin, O., e Castagna, P. (2012). Generation of multiplatform control for transitic systems using a component-based approach. In *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies Factory Automation (ETFA 2012)*, p. 1–8, Krakow, Poland. IEEE.

Carreira, P., Amaral, V., e Vangheluwe, H. (2020). Multi-paradigm modelling for cyber-physical systems: foundations. In *Foundations of multi-paradigm modelling for cyber-physical systems*, p. 1–14. Springer.

Dabney, J. B. e Harman, T. L. (2004). *Mastering simulink*. Pearson, Upper Saddle River, New Jersey.

Drave, I., Michael, J., Müller, E., Rumpe, B., e Varga, S. (2022). Model-driven engineering of process-aware information systems. *SN Computer Science*, 3(6):479.

Euzenat, J., Shvaiko, P., et al. (2007). *Ontology matching*, volume 18. Springer.

Kherraf, S., Lefebvre, É., e Suryn, W. (2008). Transformation from cim to pim using patterns and archetypes. In *19th Australian Conference on Software Engineering*, p. 338–346, Adelaide, Australia. IEEE.

Klikovits, S. e Buchs, D. (2021). Pragmatic reuse for dsml development: Composing a dsl for hybrid cps modeling. *Software and Systems Modeling*, 20(3):837–866.

Kobeissy, N., Genet, M. G., e Zeghlache, D. (2007). Mapping xml to owl for seamless information retrieval in context-aware environments. In *IEEE International Conference on Pervasive Services*, p. 361–366, Istanbul. IEEE.

DOI: 10.59254/sbpo-2025-212143
https://proceedings.science/p/212143?lang=pt-br

Lee, E. A. (2015). The past, present and future of cyber-physical systems: A focus on models. *Sensors*, 15(3):4837–4869. ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/15/3/4837.

Lehner, D., Zhang, J., Pfeiffer, J., Sint, S., Splettstößer, A.-K., Wimmer, M., e Wortmann, A. (2025). Model-driven engineering for digital twins: a systematic mapping study. *Software and Systems Modeling*, p. 1–39.

Liu, B., Zhang, Y.-r., Cao, X.-l., Liu, Y., Gu, B., e Wang, T.-x. (2020). A survey of model-driven techniques and tools for cyber-physical systems. *Frontiers of Information Technology & Electronic Engineering*, 21(11):1567–1590.

Melo, T., Silva, L., Perkusich, A., Silva, J., e da Rocha Neto, J. (2015). Software environments as learning tools for modeling engineering systems. In *Proceedings of the 7th International Conference on Computer Supported Education-Volume 2*, p. 224–231, Lisbon,Portugal. Association for Computing Machinery.

Silva, L. C., Almeida, H. O., Perkusich, A., e Perkusich, M. (2015). A model-based approach to support validation of medical cyber-physical systems. *Sensors*, 15(11):27625–27670.

SILVA, L. C. et al. (2015). Uma abordagem baseada em modelos para suporte à validação de sistemas médicos físico-cibernéticos.

Sirjani, M., Provenzano, L., Asadollah, S. A., Moghadam, M. H., e Saadatmand, M. (2021). Towards a verification-driven iterative development of software for safety-critical cyber-physical systems. *Journal of Internet Services and Applications*, 12(1):2.

WHO (2020). Vision impairment and blindness, Fact Sheet Nº282. http://www.who.int, Last accessed on 2021-01-20.

Wieringa, R. J. (2014). *Design Science Methodology for Information Systems and Software Engineering*. Springer, Philadelphia, Pennsylvania.

Xiao, T. e Fan, W. (2012). Modeling and simulation framework for cyber physical systems. In *Advanced Methods, Techniques, and Applications in Modeling and Simulation*, p. 105–115. Springer, Korea.

Xinquan, W., Xuefeng, Y., Xingchan, L., e Yongzhen, W. (2023). Simulating hybrid sysml models: a model transformation approach under the devs framework. *The Journal of Supercomputing*, 79 (2):2010–2030.

Xiong, Y., Leet, E., Liu, X., Zhao, Y., e Zhong, L. C. (2005). The design and application of structured types in ptolemy ii. In *2005 IEEE International Conference on Granular Computing*, volume 2, p. 683–688, China. IEEE.

Zhan, B., Xu, X., Gao, Q., Ji, Z., Jin, X., Wang, S., e Zhan, N. (2024). Mars 2.0: A toolchain for modeling, analysis, verification and code generation of cyber-physical systems. *arXiv preprint arXiv:2403.03035*.

DOI: 10.59254/sbpo-2025-212143
https://proceedings.science/p/212143?lang=pt-br