Utilização do solidThinking Embed como interface gráfica para a construção de modelos de processos termofluidodinâmicos

1 Introdução

Este artigo tem por objetivo descrever as etapas de desenvolvimento de uma metodologia para representar visualmente modelos de processos termofluidodinâmicos com o solidThinking Embed. Inicialmente, o problema é contextualizado, de forma a identificar a necessidade de uma ferramenta gráfica para mitigar falhas humanas e conferir celeridade ao fluxograma de criação de modelos. A seguir, é justificado o uso do solidThinking Embed para a elaboração dessa ferramenta e são descritas as duas etapas de desenvolvimento: criação de uma paleta de blocos com os componentes dos modelos e criação de um tradutor responsável pela interpretação das informações contidas nos diagramas representativos dos modelos. Por fim, são apresentados os resultados do processo de tradução e uma amostra de como os arquivos gerados são utilizados em problemas de simulação.

2 Contexto

Processos termofluidodinâmicos são, em resumo, fenômenos físicos nos quais ocorre transferência de massa e/ou energia. O escoamento de um fluido numa tubulação, por exemplo, compreende transferência de massa, que se dá entre as extremidades inicial e final da tubulação no sentido axial, e de energia, na forma de trabalho nas fronteiras da tubulação e calor trocado entre o fluido e a parede da tubulação. A simulação de processos dessa natureza é de fundamental importância em diversas aplicações, tais como o desenvolvimento de simuladores para treinamento de operadores (OTS), que é um dos serviços prestados pela GT2 Tecnologia.

Uma das ferramentas utilizadas internamente pela GT2 Tecnologia para a modelagem e simulação de processos termofluidodinâmicos é a Plataforma Não Linear (PNL), que é um *solver* numérico desenvolvido em C++ sob o paradigma de orientação a objetos. Na PNL, um sistema qualquer é constituído por componentes, que podem representar equipamentos do sistema ou processos que ocorrem entre eles, e conexões, que estão associadas às linhas de fluxo de massa e energia no interior do sistema. A Figura 1, por exemplo, mostra um esquema representativo de uma seção de uma turbina a vapor de acordo com as convenções adotadas pela PNL.

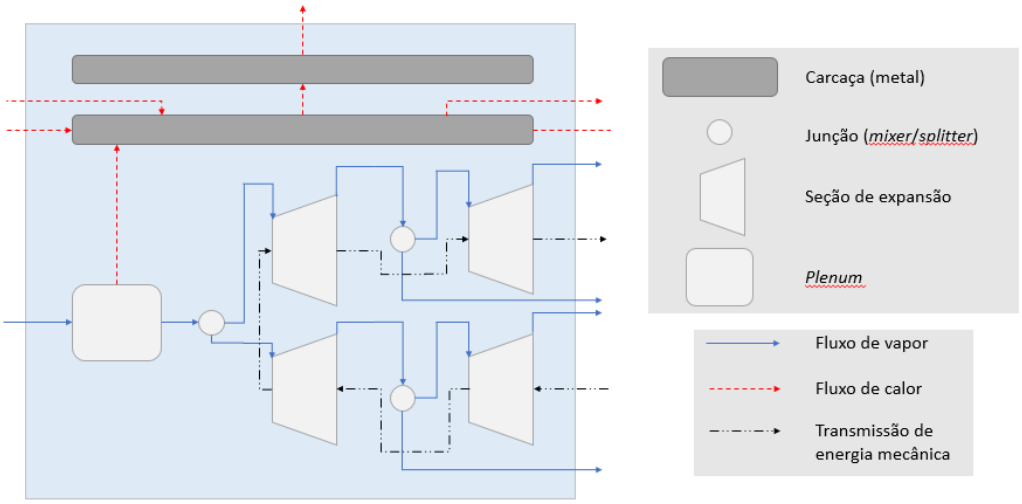

Figura 1 - Esquema de uma seção de uma turbina a vapor na Plataforma Não Linear (PNL).

O esquema da Figura 1 é uma abstração lógica. A plataforma recebe explicitamente as informações sobre o sistema modelado através de um arquivo XML contendo as declarações de todos os componentes e conexões existentes, além das definições de todas as variáveis de interface, isto é, grandezas que representam as entradas, saídas e variáveis de estado do modelo matemático associado ao sistema. A Figura 2 mostra o código XML correspondente à turbina a vapor mostrada como exemplo.

```
<?xml version="1.0" encoding="utf-8" ?>
<SubSistema>
  <Assembly name="SteamTurbine" delta_t="1.0">
    <Components>
      <SimpleInletDuct name="MAC10Inlet" flag_pressure="true" flag_flow="true" fluido_design="water" />
      <Plenum name="MAC10Plenum" n_in="1" n_out="1" n_qIn="0" n_qOut="1" tau="10" V="50000" />
      <Node name="MAC10Division" n_in="1" n_out="2" flag="1" />
      <SteamTurbineSection name="MAC10Section11" section="1" />
      <Node name="MAC10Extraction1" n_in="2" n_out="1" flag="2" />
      <SteamTurbineSection name="MAC10Section12" section="1" />
      <SteamTurbineSection name="MAC10Section21" section="1" />
      <Node name="MAC10Extraction2" n_in="2" n_out="1" flag="2" />
      <SteamTurbineSection name="MAC10Section22" section="1" />
      <Lumped name="MAC10InternalShell" n_qIn="3" n_qOut="3" />
      <Lumped name="MAC10ExternalShell" n_qIn="1" n_qOut="1" />
      <NaturalConvection name="MAC10Ambient" alpha="2.0" />
      <Conduction name="MAC10ShellConduction" L="0.05" />
      <Conduction name="MAA10MAC10Conduction" k="0" />
      <Conduction name="MAC10MAC20Conduction" k="0" />
      <Conduction name="MAC10AdjConduction1" k="0" />
      <Conduction name="MAC10AdjConduction2" k="0" />
    ...
    </Components>
    <Connections>
      <Connector equip1="MAC10Plenum" port1="out1" equip2="MAC10Division" port2="in1" />
      <Connector equip1="MAC10Division" port1="out1" equip2="MAC10Section11" port2="in" />
      <Connector equip1="MAC10Section11" port1="out" equip2="MAC10Extraction1" port2="in1" />
      <Connector equip1="MAC10Extraction1" port1="out1" equip2="MAC10Section12" port2="in" />
      <Connector equip1="MAC10Division" port1="out2" equip2="MAC10Section21" port2="in" />
      <Connector equip1="MAC10Section21" port1="out" equip2="MAC10Extraction2" port2="in1" />
      <Connector equip1="MAC10Extraction2" port1="out1" equip2="MAC10Section22" port2="in" />
    ...
    </Connections>
  </Assembly>
  <InterfaceEntrada>...</InterfaceEntrada>
  <InterfaceSaida>...</InterfaceSaida>
  <CI>...</CI>
</SubSistema>
```

Figura 2 - Código XML com as informações da turbina a vapor mostrada como exemplo.

Em um primeiro momento, a etapa de criação do arquivo XML que representa um sistema a ser simulado na PNL é feita de forma manual, isto é, o próprio desenvolvedor escreve o arquivo integralmente, definindo os componentes, as conexões e as variáveis do modelo. Por seu caráter manual, tal etapa é passível de erros, além de demandar tempo, principalmente para sistemas com muitos componentes e conexões.

A automatização dessa tarefa, portanto, agrega valor ao fluxo de desenvolvimento de modelos, pois elimina a possibilidade de erros humanos e torna o processo mais rápido. Uma forma de automatizar o processo consiste na construção de uma ferramenta visual, na qual o desenvolvedor possa montar um esquema como o mostrado na Figura 1, e o código da Figura 2 seja gerado de forma automática, a partir de um algoritmo de tradução do esquema.

3 Metodologia

Para a criação da ferramenta gráfica, há uma variedade de possibilidades. É possível construir uma interface do zero, utilizando linguagens como C++, C# ou Java e *frameworks* como Windows Forms, WPF ou Java Swing, por exemplo. No entanto, a implementação de todos os recursos mínimos necessários para o funcionamento de um editor de diagramas é uma tarefa trabalhosa e desnecessária, pois já existem soluções similares disponíveis no mercado. No trabalho aqui apresentado, opta-se por aproveitar os recursos do solidThinking Embed, que é um ambiente visual para desenvolvimento de modelos de sistemas embarcados. No Embed, é possível criar diagramas arrastando blocos e conectando-os entre si, o que converge com os conceitos de componentes e conexões da PNL.

A criação da ferramenta no Embed compreende duas etapas básicas: primeiramente, deseja-se que seja possível desenhar o esquema do modelo; em seguida, deve-se haver uma forma de converter tal desenho no arquivo XML desejado. Para a primeira etapa, é utilizado o próprio ambiente de desenho de diagramas no Embed, tendo em destaque o seu recurso de criação de paletas de blocos. Para a última, é implementado um tradutor em Python, capaz de interpretar arquivos exportados pelo Embed em arquivos XML que obedecem às premissas da PNL.

3.1 Criação de paletas de blocos

Uma paleta é um conjunto de blocos pré-definidos que podem ser incluídos em qualquer diagrama no Embed. A criação de uma paleta no programa é feita de maneira intuitiva: dentro do diretório de instalação do programa há uma pasta chamada “Toolbox” contendo as paletas já existentes; para se criar uma nova, basta adicionar uma subpasta, e cada arquivo em seu interior corresponderá a um bloco. Além disso, é possível criar paletas aninhadas, pois o Embed lê as pastas de maneira recursiva.

Dessa forma, foi criada a paleta destacada na Figura 3, que também mostra o ambiente de edição de diagramas do programa. Para utilizar algum de seus blocos, o usuário precisa apenas clicar no nome do bloco desejado, arrastando o mouse até soltá-lo no diagrama.

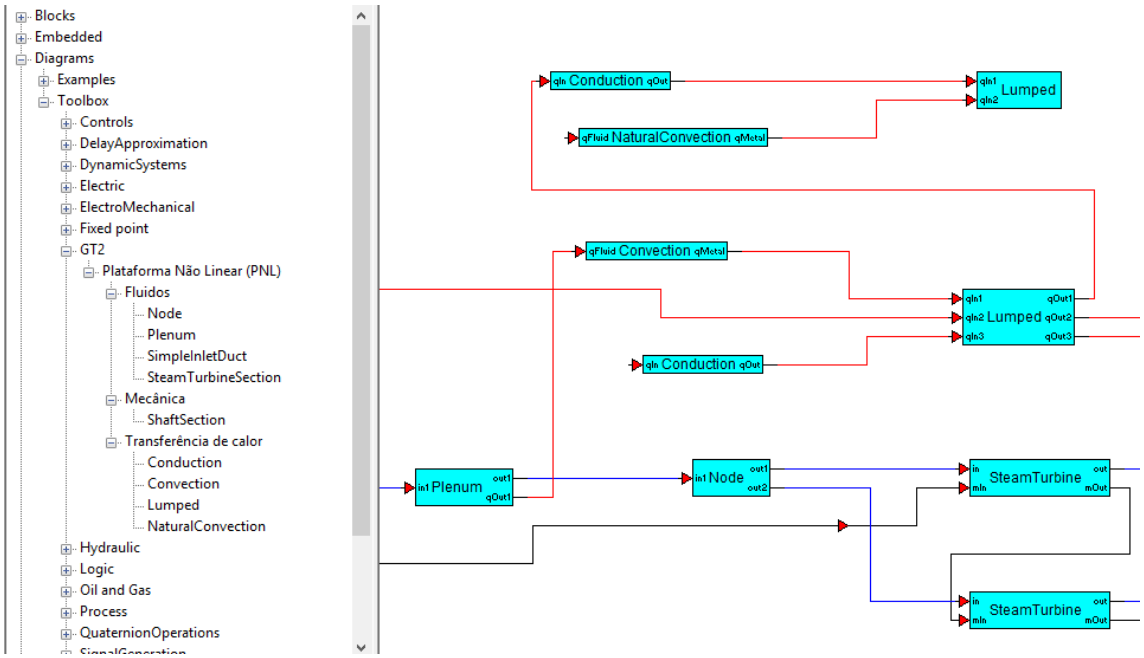

Figura 3 - Ambiente do solidThinking Embed para desenho de diagramas.

Na Figura 3, nota-se que cada bloco possui uma quantidade diferente de portas de entrada e saída, e blocos e portas são identificados por nomes customizados. Também há diferentes tipos de conectores: as conexões em azul representam fluxo de massa, enquanto as conexões em vermelho correspondem ao fluxo de calor e as em preto dizem respeito à transmissão de energia mecânica na forma de trabalho de eixo.

Vale destacar ainda que cada bloco é capaz de armazenar propriedades. Ao se clicar com o botão direito em um dos blocos do tipo "SteamTurbine", por exemplo, é aberta a janela mostrada na Figura 4, na qual o usuário pode editar o nome da seção de expansão da turbina a vapor em questão ("name"), além de modificar o valor de sua eficiência isentrópica ("eta") e de um *flag* de seu modelo matemático ("section").

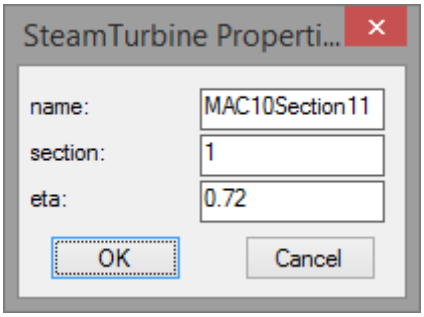

Figura 4 - Exemplo de janela para edição das propriedades de um bloco.

Assim, é possível concluir que o Embed atende a todos os requisitos levantados para a ferramenta: no ambiente do programa, podem ser criados diagramas contendo blocos customizados que armazenam propriedades e se comunicam entre si através de portas e conectores de diferentes tipos.

3.2 Interpretação de diagramas

O diagrama de blocos é exportado pelo Embed em um arquivo com a extensão *.vsm. Porém, a PNL espera receber um arquivo XML com as informações necessárias. Para fazer a conversão entre os dois protocolos,

cria-se um *script* em Python, linguagem escolhida pelas seguintes razões: fácil manipulação de *strings* e arquivos de texto; possui um pacote padrão ("ElementTree") para a manipulação de dados em XML; suporte a orientação a objetos e possibilidade de integração com diversas ferramentas, incluindo o solidThinking Compose, programa que compõe a mesma suíte do Embed.

O primeiro passo para a construção do tradutor é o entendimento da estrutura do arquivo *.vsm gerado pelo Embed. Caso o arquivo fosse binário ou utilizasse algum tipo desconhecido de criptografia, a interpretação do arquivo seria inviável. Porém, o arquivo *.vsm é, na verdade, um arquivo de texto, no qual cada linha fornece uma informação sobre o diagrama ou algum de seus elementos. Tais informações incluem nomes de blocos e portas, conexões, coordenadas geométricas e propriedades dos blocos, por exemplo. Além disso, essas informações estão presentes no arquivo de forma consistente, isto é, cada tipo de informação é sempre descrito da mesma forma, o que permite a identificação de padrões para a construção do tradutor.

O tradutor pode, portanto, ser implementado de forma procedural: as linhas do arquivo *.vsm são lidas uma a uma, e listas contendo os objetos básicos (blocos, conectores e propriedades) vão sendo atualizadas incrementalmente. Para cada tipo de bloco, uma rotina de validação específica é chamada e, caso as informações sejam todas válidas, o arquivo XML é gerado no diretório corrente ao final do processo. O *script* pode ser chamado pelo próprio *prompt* de comando do Python ou através de um programa com suporte à linguagem, como o solidThinking Compose.

4 Resultados

Com a implementação do tradutor, o processo de desenho e testes de modelos na PNL se completa. O fluxograma da Figura 5 mostra as etapas que o usuário deve realizar, na nova solução, para construir um modelo do zero e testá-lo.

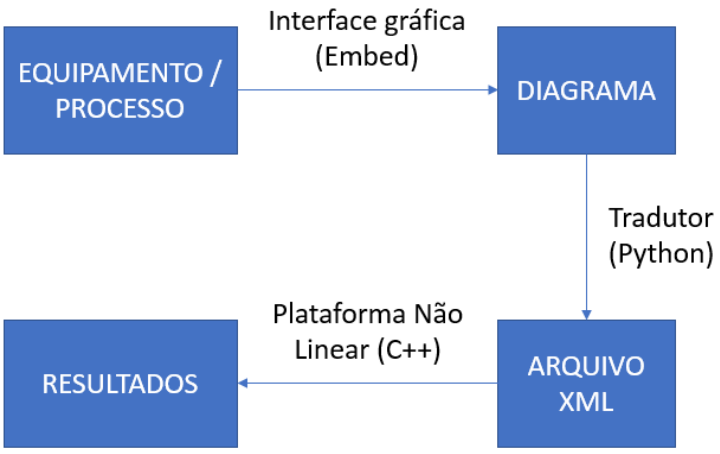

Figura 5 - Etapas do processo de desenho e testes de modelos de processos termofluidodinâmicos com a nova ferramenta.

Para validar o processo, toma-se o modelo de turbina a vapor usado como exemplo ao longo deste artigo. Os gráficos presentes na Figura 6 mostram alguns resultados de um teste qualitativo do modelo, no qual as pressões a montante e a jusante da turbina foram variadas, para se observar o efeito resultante na potência da turbina, que é uma das variáveis de saída do modelo.

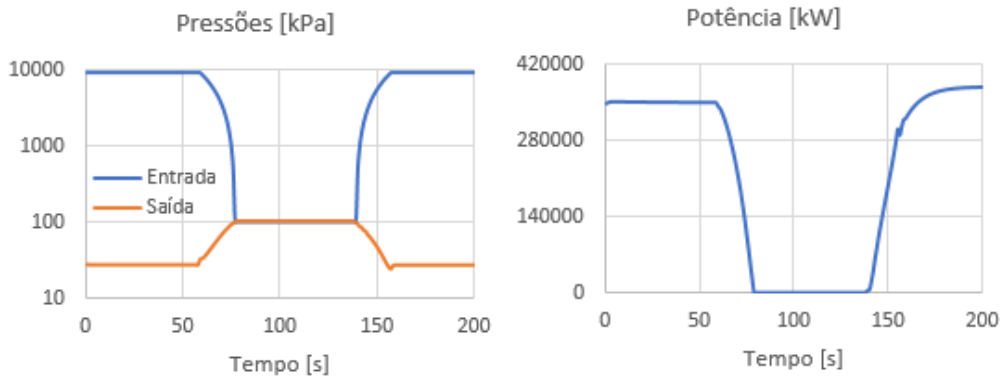

Figura 6 - Resultados do teste do modelo de turbina a vapor.

Nota-se, nos gráficos, a coerência física do modelo: quando as pressões na entrada e saída da turbina se igualam, a potência se anula; ao final do teste, quando a diferença entre as pressões retorna ao valor original, o valor da potência volta ao patamar previamente observado. Os resultados obtidos no teste são os mesmos dos testes realizados anteriormente, antes da implementação da ferramenta, uma vez que o conteúdo do arquivo XML gerado pelo tradutor não difere do arquivo escrito manualmente. Conclui-se, assim, a validação do conjunto interface gráfica + tradutor para o caso estudado.

5 Conclusão

Neste artigo, foram descritas as etapas de desenvolvimento de uma ferramenta visual para a modelagem de processos termofluidodinâmicos. Primeiramente, o problema foi contextualizado, salientando-se o potencial efeito da ferramenta em questão no tempo de desenvolvimento dos modelos. A seguir, foram descritas as etapas da criação da ferramenta, que incluem a customização do solidThinking Embed para o desenho dos modelos e a implementação de um *script* que serializa as informações do desenho no formato esperado pela plataforma de simulação já utilizada pela GT2 Tecnologia. Ao final, foram mostrados os resultados da metodologia aplicada a um caso já modelado anteriormente. Não houve diferença entre os resultados decorrentes com a ferramenta criada e os obtidos *a priori*, o que confirma a validade da estratégia adotada. O trabalho desenvolvido permite comprovar a praticidade do solidThinking Embed para o desenho de diagramas de blocos, pois foi possível construir diagramas que obedecem a uma série de premissas não elementares e integrá-los com ferramentas desenvolvidas em outras linguagens de programação e diferentes contextos. Além disso, o escopo deste artigo contemplou somente modelos pertinentes a áreas da engenharia mecânica, mas a estratégia adotada poderia ser estendida a outros tipos de modelagem por blocos, tais como circuitos elétricos e lógicos.