INSTITUTO DE PESQUISAS ENERGÉTICAS E NUCLEARES
Autarquia Associada à Universidade de São Paulo

FastLAP - Desenvolvimento de um pré-processador gráfico visual para o código RELAP5

DANIEL FERNANDO MONACO

Dissertação apresentada como parte dos requisitos para obtenção do Grau de Mestre em Ciências na Área de Tecnologia Nuclear - Reatores

Orientadora:
Profa. Dra. Gaianê Sabundjian

São Paulo
2019

INSTITUTO DE PESQUISAS ENERGÉTICAS E NUCLEARES
Autarquia Associada à Universidade de São Paulo

FastLAP - Desenvolvimento de um pré-processador gráfico visual para o código RELAP5

Versão Corrigida

Versão Original disponível no IPEN

DANIEL FERNANDO MONACO

Dissertação apresentada como parte dos requisitos para obtenção do Grau de Mestre em Ciências na Área de Tecnologia Nuclear - Reatores

Orientadora:
Profa. Dra. Gaianê Sabundjian

São Paulo
2019

Como citar:

MONACO, D. F. ***FastLAP - Desenvolvimento de um pré-processador gráfico visual para o código RELAP5***. 2019. 222 p. Dissertação (Mestrado em Tecnologia Nuclear), Instituto de Pesquisas Energéticas e Nucleares, IPEN-CNEN/SP, São Paulo.
Disponível em: (data de consulta no formato: dd/mm/aaaa)

Ficha catalográfica elaborada pelo Sistema de geração automática da Biblioteca IPEN/USP,
com os dados fornecidos pelo(a) autor(a)

Monaco, Daniel Fernando
FastLAP - Desenvolvimento de um pré-processador gráfico visual para o código RELAP5 / Daniel Fernando Monaco; orientador Gaianê Sabundjian. -- São Paulo, 2019.
222 p.

Dissertação (Mestrado) - Programa de Pós-Graduação em Tecnologia Nuclear (Reatores) -- Instituto de Pesquisas Energéticas e Nucleares, São Paulo, 2019.

1. RELAP5. 2. visual. 3. pré-processador. 4. reator nuclear. 5. Excel. I. Sabundjian, Gaianê, orient. II. Título.

FOLHA DE APROVAÇÃO

Autor: Daniel Fernando Monaco

Título: FastLAP - Desenvolvimento de um pré-processador gráfico visual para o código RELAP5

Dissertação apresentada ao Programa de Pós-Graduação em Tecnologia Nuclear da Universidade de São Paulo para obtenção do título de Mestre em Ciências

Data: 18 / 06 / 19

Banca Examinadora

Prof. Dr.: JORGE LUIS BALIÑO

Instituição: PME-USP Julgamento: APROVADO

Prof. Dr.: Gaianê Sabundjian

Instituição: IPEN Julgamento: APROVADO

Prof. Dr.: Patricia Andrea Paladino

Instituição: IFSP - Campus São Paulo Julgamento: Aprovado

Prof. Dr.: _____

Instituição: _____ Julgamento: _____

Prof. Dr.: _____

Instituição: _____ Julgamento: _____

DEDICATÓRIA

A Deus, de onde provêm todo conhecimento e toda ciência.

À minha esposa, Luciana, amor da minha vida, mulher virtuosa que está ao meu lado em todas as vitórias e derrotas.

A minhas filhas, Melina e Olívia, anjos na terra, herança do Senhor à minha vida, para que sempre vejam os frutos do esforço e dedicação.

A meus pais, Renato (*In memoriam*) e Mônica, pois foram por seus exemplos e esforços que cheguei até aqui.

À Gaianê, minha orientadora e amiga, por sua perseverança e fé neste trabalho e em mim.

AGRADECIMENTOS

Agradeço a Deus, por me permitir chegar até aqui, por ter me fortalecido onde fraquejei, e me dado conhecimento e capacitação.

Agradeço à minha esposa, Luciana, pela paciência em assistir meus seminários, revisões de ortografia, avaliações de meu desempenho, e pela motivação diária. Sem você ao meu lado, este trabalho jamais teria se concluído.

Agradeço às minhas filhas Olívia e Melina, pela compreensão e pelas horas e horas de ausência: vocês serão recompensadas!

Agradeço à minha professora e orientadora, Dra. Gaianê, pelas horas e horas ao meu lado, pelo apoio, pelo trabalho e pelo carinho.

Agradeço também ao Instituto de Pesquisas Energéticas e Nucleares – IPEN-CNEN, em especial ao grupo do CEN, pela oportunidade de conduzir este trabalho.

Pois o Senhor é quem dá sabedoria; de sua boca
procedem o conhecimento e o discernimento.
Provérbios 2:6

RESUMO

MONACO, D. F. FastLAP - Desenvolvimento de um pré-processador gráfico visual para o código RELAP5. 2019, 222 p. Dissertação (Mestrado em Tecnologia Nuclear) – Instituto de Pesquisas Energéticas e Nucleares – IPEN-CNEN/SP. São Paulo.

As energias limpas têm contribuído para o aumento de investimento e pesquisas em energia nuclear na última década. No entanto, as ocorrências dos acidentes nucleares ao longo da história ainda geram insegurança para a população em geral. Os órgãos reguladores têm aumentado as exigências de segurança em plantas nucleares e, devido a isto, vêm fazendo esforços na realização de simulações numéricas com programas computacionais de análise de acidentes em instalações nucleares, com a finalidade de garantir a segurança da planta e da população do entorno, antes mesmo de sua construção. No Brasil, para atender as exigências do órgão regulador brasileiro, a administradora dos reatores nucleares nacionais deve apresentar um estudo termo-hidráulico na área de análise de acidentes e transientes operacionais para as instalações nucleares. Isto é feito com a finalidade de licenciar as plantas nucleares, utilizando ferramentas computacionais apropriadas, tais como o código RELAP5. Esse programa computacional é muito eficiente na simulação de acidentes em usinas nucleares, mas não é muito amigável quanto à inserção de seus dados de entrada. Essa dificuldade motivou o desenvolvimento de pré-processadores para auxiliar a preparação dos dados geométricos de plantas nucleares, que é uma parte dos dados de entrada para o código RELAP5. Além disso, antes de iniciar o uso dessas ferramentas computacionais, faz-se necessário que o usuário monte uma nodalização ou modelagem do problema, de forma a representar mais adequadamente a planta e a fenomenologia envolvida durante um acidente ou transiente, sendo que ambas sejam adequadamente atendidas pela ferramenta. O objetivo desse trabalho foi o de criar um pré-processador capaz de auxiliar o usuário na tarefa de preparar os dados de entrada para o código RELAP5 e, também, de auxiliá-lo na elaboração da nodalização necessária para representar de forma mais real possível a planta em estudo. O pré-processador desenvolvido nesse trabalho é gráfico, visual e amigável, de forma a permitir que o usuário inicie a nodalização com o uso desta ferramenta, integrando assim as etapas de modelagem e preparação dos dados de entrada para o código RELAP5 em uma única fase, reduzindo assim, os esforços necessários para a sua realização, otimizando o

tempo gasto. Para atingir esse objetivo, foi utilizado como plataforma de desenvolvimento o *MS Excel*®, uma ferramenta de planilha de cálculo eletrônica largamente utilizada, e foi construído para ele um complemento por meio da linguagem C# e da plataforma *.NET*. E através desta linguagem, seus recursos de orientação a objetos e total integração com a ferramenta *MS Excel*®, como *Interop* e *Visual Studio Tools for Office* (VSTO) integrados, foi possível um desenvolvimento mais rápido de uma ferramenta eficiente para essa finalidade, fazendo uso de recursos que não estariam disponíveis por meio do VBA (*Visual Basic for Applications*). O pré-processador desenvolvido nesse trabalho permite a criação da nodalização de um problema termo-hidráulico, onde os componentes hidrodinâmicos são desenhados por meio da automação de *AutoShapes* do *MS Excel*® e os dados de entrada desses componentes são alimentados por meio de caixas de diálogo amigáveis e funcionais. Uma vez que o pré-processador foi criado como um complemento para *MS Excel*®, as linhas de programação do pré-processador criado não ficam restritas a uma única planilha, facilitando sua atualização e redistribuição. O resultado obtido por meio desse trabalho foi o FastLAP, um pré-processador para RELAP5 visual, robusto e amigável. Por meio do FastLAP, criado nesse trabalho, reduziu-se o esforço do usuário do código RELAP5 tanto no preparo da nodalização como no preparo dos dados de entrada para o código, uma vez que a ferramenta é amigável e exibe tanto os nomes das propriedades conforme definidos pelo código RELAP5, bem como os nomes das grandezas físicas reais que estão sendo representadas. O pré-processador foi testado na elaboração da nodalização e dos dados de entrada do RELAP5 para um problema experimental encontrado na literatura e mostrou-se uma poderosa ferramenta gráfica, ajudando os usuários do RELAP5 a organizar visualmente os dados de entrada e oferecendo condições para analisar os resultados mais rapidamente. Esse trabalho criou não somente uma nova ferramenta de apoio para o usuário RELAP5, mas sim uma nova abordagem para a simulação de acidentes termo-hidráulicos com o código, fundindo as duas etapas: de nodalização e preparação dos dados de entrada.

Palavras-chave: RELAP5; visual; pré-processador; reator nuclear, *Excel*.

ABSTRACT

MONACO, D. F. FastLAP - ***Development of a graphic visual preprocessor for RELAP5 code*****.** 2019, 222 p. Dissertação (Mestrado em Tecnologia Nuclear) – Instituto de Pesquisas Energéticas e Nucleares – IPEN-CNEN/SP. São Paulo.

Clean energy has contributed to increased investment and research in nuclear power in the last decade. However, the occurrence of nuclear accidents throughout history still causes the population to feel unsafe. Regulatory agencies have increased the safety requirements in nuclear plants and due to this they have been making efforts to carry out numerical simulations with computing programs for the analysis of accidents in nuclear installations to assure the safety for the plant and the surrounding population, even before its construction. In Brazil, in order to meet the requirements of the Brazilian regulatory agency, the administrator of the national nuclear reactors must present a thermo-hydraulic study in the area of accident analysis and operational transients for nuclear installations. This is done in order to license nuclear plants, using appropriate computational tools, such as the RELAP5 code. This computing program is very efficient in simulation of accidents in nuclear power plants, but it is not very friendly on entering its input data. This issue has motivated the development of preprocessors to assist the preparation of geometric data from nuclear plants, which is part of the input data for the RELAP5 code. In addition, before starting to use these computing tools, the user needs to assemble a nodal or modeling of the problem to better represent the plant and the phenomenology involved during an accident or transient in order to allow both to be properly simulated by the tool. The aim of this work was to create a preprocessor capable of leveraging user on input data preparation for the RELAP5 code as well as assisting him in the creation of the nodalization diagram required to get the best representation – as real as possible – of the power plant being studied. The preprocessor developed in this work is graphical, visual and user-friendly in order to allow the user to begin the nodalization by using this tool, thus integrating the steps of modeling and preparing the input data for the RELAP5 code in a single phase, and also reducing the efforts needed to achieve it, reducing the time spent in this task. To achieve this goal, MS Excel® – a widely used electronic spreadsheet tool – was used as a development platform, and a MS-Excel® add-in was built with the C # language and the .NET platform. With the use of this

programming language, its object-oriented features and full integration with the MS Excel® tool, thru Interop and Visual Studio Tools for Office (VSTO), it was possible to achieve a faster development of an efficient tool for this purpose, making use of features that would not be available through the Visual Basic for Application (VBA). The preprocessor developed in this work allows the building of the nodalization of a thermo-hydraulic problem, where the hydrodynamic components are designed through the automation of MS Excel® AutoShapes, and the input data of these components are inputted thru friendly and functional interfaces. Once the preprocessor was created as a MS Excel® add-in, the programming lines created are not restricted to a single worksheet, which makes it easier to be updated and redistributed. The result of this work is FastLAP, a RELAP5 Preprocessor, which reduces the user's effort in both preparing the nodalization and preparing the input data for RELAP5 code, once the tool is user-friendly and displays both the names of the properties as defined by the RELAP5 code and the names of the actual physical amounts being represented. The preprocessor was tested in the elaboration of the input data for RELAP5 regarding an experimental problem found in the literature and has proven to be a powerful graphical tool, helping RELAP5 users to visually organize the input data and giving conditions for a faster analysis of results. This work created not only a new aid tool for the RELAP5 users, but also a brand-new approach for the simulation of thermo-hydraulic accidents with the code, merging two phases: nodalization and preparation of the input data.

Key words: RELAP5; visual; preprocessor; nuclear reactor; Excel.

LISTA DE TABELAS

Página

LISTA DE FIGURAS

Página

LISTA DE SIGLAS E ABREVIAÇÕES

API	*Application Programming Interfaces, do MS Windows®*
ATHLET	*Analysis of Thermal-hydraulics of Leaks and Transients*
AEC	*Atomic Energy Commission*
BWR	*Boiling Water Reactor*
CAFEAN	*Common Application Framework for Engineering Analysis*
CANDU	*CANadian Deuterium Uranium*
CATHARE	*Code for Analysis of Thermal-Hydraulics during an Accident of Reactor and Safety Evaluation*
CDTN	Centro de Desenvolvimento da Tecnologia Nuclear
CIL	*Common Intermediate Language*
CLR	*Common Language Runtime*, da Plataforma .NET
CNEN	Comissão Nacional de Energia Nuclear
COBRA	*COolant Boiling in Rod Arrays*
COM	*Component Object Model*
CONTAIN	*Containment Transient Analysis*
DLL	*Dynamic Link Library*
DOM	*Document Object Model*
ERA	*Energy Reorganization Act*
ERP	*Enterprise Resource Planning*
FORTRAN	*Formula Translator*
GRS	*Gesellschaft für Angalen-und Reaktorsicherheit*
GUI	*Graphical User Interface*
IAEA	*International Atomic Energy Agency*
IDE	*Integrated Development Environment*
IEN	Instituto de Energia Nuclear

INES	*International Nuclear and Radiological Event Scale*
INL	*Idaho National Laboratory*
IPEN	Instituto de Pesquisas Energéticas e Nucleares
JRM	*Java Runtime Machine*
JSON	*JavaScript Object Notation*
LOCA	*Loss of Coolant Accident*
MELCOR	*Methods for Estimation of Leakages and Consequences of Release*
MS	*Microsoft®*
NRC	*Nuclear Regulatory Commission*
NSC	*Nuclear Safety Convention*
OOP	*Object-Oriented Programming*
PREREL5	Pré-processador matemático para o RELAP5
PWR	*Pressurized Water Reactor*
RELAP	*Reactor Excursion and Leak Analysis Program*
SIE	Sistema de Injeção de Emergência
SIS	*Safety Injection System*
SNAP	*Symbolic Nuclear Analysis Package*
TRAC	*Transient Reactor Analysis Code*
TRACE	*The TRAC/RELAP Advanced Computational Engine*
VBA	*Visual Basic for Application*
VSTO	*Visual Studio Tools for Office*
VVER	*Water-Water Energetic Reactor*
WEB	*World Wide Web* – a *Internet*
WWW	*World Wide Web* – a *Internet*

SUMÁRIO

1 INTRODUÇÃO

A busca por opções de energias alternativas no mundo tem contribuído para o aumento em investimento e pesquisa nessa área. Uma das fontes de energia considerada limpa é a energia nuclear, que teve um crescimento na última década, no entanto, os acidentes nucleares, como os ocorridos em *Three Mile Island* em 1979 (COREY, 1979) *Chernobyl* em 1986 (INTERNATIONAL ATOMIC ENERGY AGENCY, 2005), e mais recentemente em Fukushima em 2011 (GAUNTT et al., 2012), ainda trazem insegurança para a população. Esses eventos aumentaram a pressão por parte dos órgãos reguladores quanto às exigências de segurança em plantas nucleares. Hoje, os órgãos reguladores, internacional e nacional, de controle do uso da energia nuclear, a *International Atomic Energy Agency* (IAEA) e a Comissão Nacional de Energia Nuclear (CNEN), respectivamente, têm aplicado esforços realizando simulações numéricas com programas computacionais de análise de acidentes em instalações nucleares, com a finalidade de garantir a segurança da população no entorno.

O estudo termo-hidráulico utilizado para análise de acidentes e transientes em reatores nucleares é feito com o uso de algumas ferramentas computacionais sofisticadas. A maioria destes programas possui uma filosofia realista (*best estimate*) aplicada a reatores à água leve do tipo *Pressurized Water Reactor* (PWR) e sistemas associados.

Com a finalidade de atender essas exigências, as empresas que gerenciam os reatores nucleares nacionais necessitam realizar alguns estudos termo-hidráulicos na área de análise de acidentes e transientes, para que a instalação seja certificada. Isto é feito com a utilização de algumas ferramentas computacionais sofisticadas, ou seja, códigos computacionais tais como: *Reactor Excursion and Leak Analysis Program* (RELAP5) (IDAHO NATIONAL LABORATORY, 1995), *Transient Reactor Analysis Code* (TRAC) (SPORE, 1993), *Analysis of Thermal Hydraulics of LEaks and Transients* (ATHLET) (GESELLSCHAFT FÜR ANGALEN-UND REAKTORSICHERHEIT, 2006), *Code Avancé de Thermo Hydraulique pour les Accidents sur les Réacteurs à Eau* (CATHARE) (MICAELLY; BESTUIB, 1988), entre outros. A maioria desses programas possui uma filosofia realista, ou seja, tenta retratar de forma mais real possível a fenomenologia dos acidentes e transientes analisados. Todos esses programas foram desenvolvidos segundo essa abordagem, para os reatores refrigerados a água leve do tipo PWR, *Boiling Water Reactor* (BWR) e *CANadian Deuterium Uranium* (CANDU).

No Brasil, a ferramenta escolhida para a realização dos estudos termo-hidráulicos e simulações de acidentes é o código RELAP5. Esse programa é muito eficiente na simulação de acidentes em usinas nucleares tais como: a perda de refrigerante primário por pequenas ou grandes rupturas de tubulações, perda de potência elétrica, perda de água na alimentação, entre outros. Entretanto, a grande quantidade de dados de entrada para o RELAP5 o torna extremamente trabalhoso, e pode originar uma série de erros por parte dos usuários. Essas dificuldades motivaram o desenvolvimento de pré-processadores, tais como Pré-Processador do RELAP5 (PREREL5) (PALADINO, 2006) e o Programa de Cálculo do RELAP5 (PCRELAP5) (SILVESTRE, 2016), para a preparação dos dados de entrada geométricos de plantas nucleares para o código em questão. Todavia, verificou-se também a necessidade de um processador gráfico visual para auxiliar na nodalização de plantas nucleares, o que não foi contemplado nessas versões de pré-processadores citados.

O PCRELAP5 é uma ferramenta que auxilia o usuário do RELAP5, na preparação de seus dados de entrada, ou seja, realiza uma série de cálculos algébricos por meio do *MS Excel*® (MICROSOFT EXCEL, 2019).

O *MS Excel*® possui embutido uma linguagem de programação conhecida como *MS-Visual Basic for Applications*® (VBA) (BATTISTI, 2013), que permite que virtualmente qualquer tarefa dentro, ou mesmo fora, do ambiente do *MS Excel*® seja automatizada. Isto facilita a criação de fórmulas de forma automática, bem como a possibilidade de criação de desenhos e formas geométricas para representar a modelagem ou a nodalização de uma planta nuclear para ser utilizada por códigos de análise termo-hidráulica.

Este trabalho tem a finalidade de desenvolver um processador gráfico visual para o código RELAP5, um aperfeiçoamento do PCRELAP5, disponibilizando uma versão gráfica, visual, e orientada à nodalização de plantas nucleares, tornando as correções, modificações e a utilização do RELAP5 mais amigáveis para com o usuário. Para esse fim será utilizada a linguagem C# para programação, tendo o *MS Excel*® como base e a Plataforma *.NET*® como plataforma de desenvolvimento, o que permitirá ao usuário do código criar graficamente a nodalização de plantas nucleares.

1.1 Objetivos

O objetivo deste trabalho é a criação de um pré-processador gráfico visual orientado à nodalização de plantas nucleares, para que essas sejam utilizadas pelo código RELAP5. Esse pré-processador gráfico inclui também os cálculos geométricos da planta e

pretende ser o mais amigável e mais confiável para o usuário, usando para tal fim o *MS Excel*® como plataforma de desenvolvimento. A ferramenta tem a finalidade de criar graficamente a nodalização para o RELAP5.

Assim, esse trabalho visa facilitar a manipulação de dados de entrada em códigos de análise de acidentes nucleares, por parte de seus usuários, trazendo mais segurança e confiança dos resultados obtidos para os órgãos reguladores e para as operadoras.

1.2 Motivação

A motivação desse trabalho surgiu devido as dificuldades encontradas pelos usuários e da comunidade em geral, na preparação dos dados de entrada e da modelagem de usinas nucleares com o código RELAP5.

As ferramentas atuais existentes para apoiar os usuários do RELAP5 nessa tarefa ou são pouco práticas, exigindo comandos e cálculos complexos para seu uso, ou então limitam-se ao apoio do usuário na organização visual do código RELAP5, como é o caso do programa *Symbolic Nuclear Analysis Package* (SNAP, 2015).

A criação de pré-processadores para o RELAP5 já foi objeto de algumas pesquisas no Brasil (PALADINO, 2006; SILVESTRE, 2016) e no mundo. Porém, este é um trabalho que tem o objetivo de criar também a modelagem visual de plantas nucleares para esse código.

1.3 Organização da dissertação

Essa dissertação foi organizada e escrita seguindo o Guia para elaboração de dissertações e teses, instrumento oficial do programa de Pós-graduação de Tecnologia Nuclear IPEN/USP (IGAMI; VIEIRA, 2017).

O capítulo 1 apresenta a introdução do trabalho, bem como os objetivos, a motivação e a organização geral dessa dissertação de mestrado.

O capítulo 2, revisão bibliográfica, apresenta um breve histórico dos acidentes nucleares, além de apresentar os principais programas computacionais de análise termo-hidráulica, com destaque para o código RELAP5, traz uma visão geral sobre outros pré e pós-processadores para esses códigos, e apresenta uma visão geral sobre a programação orientada a objetos, usada para desenvolvimento desse trabalho.

O capítulo 3 apresenta o desenvolvimento do trabalho em si. A primeira parte mostra uma visão geral das ferramentas utilizadas para o desenvolvimento do trabalho, e as principais razões que as levaram a ser escolhidas, além de apresentar uma visão geral da

“Experiência SUPER CANON”, usada como base para teste do pré-processador desenvolvido. Em seguida, são apresentadas as principais etapas para o desenvolvimento do pré-processador. Também é apresentado o pré-processador desenvolvido, além de todo os paradigmas conceituais que embasaram seu estado atual.

O capítulo 4, resultados e discussões, apresenta um compilado do aprendizado obtido durante o desenvolvimento desse trabalho, mostra os resultados obtidos com a “Experiência SUPER CANON” simulada através do código RELAP5 manualmente, apresenta o desenvolvimento passo a passo do mesmo experimento com a utilização do pré-processador, e a comparação, de ambos os resultados obtidos, com os resultados experimentais.

No capítulo 5 estão as conclusões e propostas para trabalhos futuros.

Por fim, são fornecidas as referências bibliográficas utilizadas neste trabalho.

O APÊNDICE A apresenta um trecho da programação do componente PIPE, onde são programadas suas propriedades e regras de negócio.

O APÊNDICE B apresenta a programação da tela do componente PIPE, onde as principais validações de tela referentes a esse componente são apresentadas.

O APÊNDICE C apresenta a programação da Estrutura de Troca de Calor, onde as principais validações de tela referentes a esse componente são apresentadas.

O APÊNDICE D apresenta a programação da tela de Estrutura de Troca de Calor, onde as principais validações de tela referentes a essa estrutura são apresentadas.

O APÊNDICE E apresenta um trecho da programação contendo a programação da superclasse “CComponent”, usada como classe base para a programação de todos os componentes programados neste pré-processador.

O APÊNDICE F apresenta a programação da superclasse de telas, a classe “FrmH”, que é usada como base para todas as telas de componentes e estruturas programados no pré-processador.

O APÊNDICE G apresenta o arquivo de dados de entrada feito manualmente para o código RELAP5, usado para simulação da “Experiência SUPER CANON” nesse código.

O APÊNDICE H apresenta o arquivo de dados de entrada para o código RELAP5, referente a “Experiência SUPER CANON”, criado por meio do uso do pré-processador criado neste trabalho.

O APÊNDICE I apresenta um resumo sobre as curvas homólogas de potência, usado como base para desenho da tela do componente PUMP.

Por fim, o APÊNDICE J apresenta um resumo sobre a Programação Orientada a Objetos, que foi o paradigma de programação usado para desenvolvimento deste trabalho.

2 REVISÃO BIBLIOGRÁFICA

A revisão bibliográfica apresenta um breve resumo de pré-processadores e ferramentas termo-hidráulicas utilizadas para o cálculo de acidentes e transientes em plantas nucleares existentes na literatura.

2.1 Breve histórico dos acidentes nucleares

Alguns dos principais acidentes nucleares da história são apresentados a seguir.

A ocorrência de dois grandes acidentes, como nas usinas de *Three Mile Island*, nos Estados Unidos em 1979 (COREY, 1979) e *Chernobyl*, na Ucrânia em 1986 (CHERNOBYL, 2003), foi o motivo para o questionamento sobre as tecnologias e procedimentos de segurança nas instalações nucleares.

A INTERNATIONAL NUCLEAR AND RADIOLOGICAL EVENT SCALE 2009 (INES), classifica a severidade das ocorrências de eventos nucleares em uma escala logarítmica que revela o grau destas ocorrências. Os eventos classificados na escala com níveis de 1 a 3 são considerados incidentes e com níveis de 4 a 7, considerados acidentes.

No caso do reator nuclear em *Three Mile Island*, o acidente ocorrido em 1979 foi a falta de resfriamento devido a uma série de erros humanos e falhas de sistemas, fazendo com que o núcleo desse reator iniciasse o processo de derretimento. Isso causou uma séria contaminação no interior da contenção. Apesar deste acidente ter sido classificado no nível 5 da escala INES, ele não afetou nem a população local, nem o meio ambiente. Entretanto, cerca de 140 mil pessoas foram temporariamente deslocadas.

Em *Chernobyl*, Ucrânia, o reator de número 4 explodiu durante um teste de segurança no dia 26 de abril de 1986. Esse foi considerado o maior desastre nuclear da história. De acordo com as estimativas, a intensidade de radionuclídeos jogados na atmosfera devido ao derretimento do núcleo do reator contaminou parte da Europa. O acidente foi classificado como sendo de nível 7 na escala INES.

Após *Chernobyl* ficou em evidência a insuficiência das diretrizes instituídas pela Agência Internacional de Energia Atômica (IAEA) para os países com programas nucleares, já que a central de *Chernobyl* cumpria parte da regulamentação baseada nessas diretrizes. Logo, foram consultados especialistas para a criação de documentos que assegurassem de forma efetiva a segurança em centrais nucleares. Consequentemente houve mudanças nas exigências das agências reguladoras.

O acidente de Fukushima que atingiu o Japão em 2011 (GAUNTT et al., 2012), ocasionando graves consequências à população do entorno, é um exemplo de acidente na condição estendida de projeto ou acidentes além da base de projeto ou acidentes severos. Esse acidente teve graves consequências ocasionando insegurança e medo por parte do público com relação às instalações nucleares em operação.

Hoje as organizações, internacional e nacional de controle do uso da energia nuclear, a IAEA e a CNEN, tem aplicado esforços na realização de simulações numéricas com programas computacionais de acidentes severos em instalações nucleares, com a finalidade de garantir a segurança da população do entorno. O código *Methods for Estimation of Leakages and Consequences of Release* (MELCOR) (GAUNTT et al., 2000) é o mais utilizado para esse fim.

A seguir são descritos alguns destes programas computacionais.

2.2 Programas computacionais de análise termo-hidráulica

Um dos programas mais utilizados no mundo e no Brasil para a análise de acidentes e transientes nucleares é o código RELAP5, desenvolvido pela *Idaho National Laboratory* (INL) para a *Nuclear Regulatory Commission* (NRC) e vem sendo aperfeiçoado desde 1970. Existe uma série de versões do código RELAP5 sendo que os mais recentes são, por exemplo, RELAP5/MOD3.3 (IDAHO NATIONAL LABORATORY, 2010) e RELAP5/3D (IDAHO NATIONAL LABORATORY, 2019), que se distinguem das versões anteriores pela capacidade de modelagem termo-hidráulica e neutrônica tridimensional, respectivamente. Essas características, por sua vez, removem qualquer tipo de restrição para a aplicação do código na simulação de acidentes em reatores, principalmente em transientes, onde há forte acoplamento entre efeitos neutrônicos e termo-hidráulicos.

Há alguns códigos termo-hidráulicos similares ao RELAP5 que são utilizados na simulação de plantas nucleares desde a década de 80 e que não estão abertos à comunidade científica. São eles:

- *Analysis of Thermal Hydraulics of LEaks and Transients* (ATHLET), desenvolvido na Alemanha pela empresa *Gesellschaft für Angalen-und Reaktorsicherheit* (GESELLSCHAFT FÜR ANGALEN-UND REAKTORSICHERHEIT, 2006);
- *Code for Analysis of Thermal Hydraulics during an Accident of Reactor and Safety Evaluation* (CATHARE), desenvolvido na França pela GRS, destinado

a reatores PWR e *Water-Water Energetic Reactor* (VVER) (MICAELLY; BESTUIB, 1988);

- *COolant Boiling in Rod Arrays* (COBRA) (ROWE, 1967);
- *The TRAC/RELAP Advanced Computational Engineer* (TRACE) (TRAC/RELAP5), desenvolvido pela *Nuclear Regulatory Commission* (TRACE, 2007).

O ATHLET (GESELLSCHAFT FÜR ANGALEN-UND REAKTORSICHERHEIT, 2006) é um *software* de simulação para análise termo-hidráulica de acidentes e transientes. O código funciona por meio dos seguintes módulos: a) dinâmica dos fluidos e termodinâmica; b) transferência e condução de calor; c) neutrônica e cinética; d) simulação do controle da planta e uso das diferenças finitas para a solução das equações de conservação. Esses módulos possibilitam a descrição da geometria, das condições iniciais, de contorno e de limites físicos de uma instalação nuclear, bem como todas as ações de controle e segurança que existem em uma usina real para posterior simulação e análise dos acidentes e transientes.

Uma série de novos programas fundamentados na estratégia computacional do código COBRA foi desenvolvida e vários deles têm sido largamente utilizados pela indústria nuclear e órgãos reguladores como ferramentas de projeto e análise termo-hidráulica. Os programas computacionais da família COBRA são, ainda hoje, as principais ferramentas para análise termo-hidráulica do núcleo de reatores do tipo PWR utilizadas pela CNEN, órgão regulador da área nuclear no Brasil.

O código termo-hidráulico TRACE (2007), desenvolvido pela NRC, envolve as capacidades de três códigos de segurança – TRAC-P, TRAC-B e RELAP5 – sendo capaz, portanto, de analisar pequenas e grandes rupturas (*Loss of Coolant Accidents* - LOCA), além de transientes operacionais tanto em reatores PWR quanto nos BWR.

Todos os programas termo-hidráulicos mencionados nesse item do trabalho necessitam de uma série de cálculos para gerar os seus respectivos dados de entrada. Por essa razão, observou-se a necessidade da criação de pré e pós-processadores como ferramentas facilitadoras para o usuário final desses códigos.

Como este trabalho está focado no código RELAP5, esse programa será descrito de forma detalhada a seguir.

2.3 Código RELAP5

O RELAP5 é um código utilizado mundialmente para a análise de acidentes e transientes em plantas nucleares do tipo PWR, BWR, reatores CANDU e reatores de pesquisa. Esse programa computacional possui diversas versões que, inicialmente, possuíam modelos conservativos e, ultimamente, utilizam a filosofia *best estimate* ou realista.

A versão RELAP5/MOD3.3 tem sido a mais utilizada na área nuclear no Brasil e no mundo nas últimas décadas para a certificação de reatores nucleares no processo de licenciamento junto aos órgãos licenciadores, por possuir uma filosofia realista. Esse tipo de código tem a capacidade de reduzir os custos de projeto, tornando-os mais competitivos comercialmente, justamente por tratar a fenomenologia de forma mais realista.

O RELAP5 tem a capacidade de simular Acidentes de Perda de Refrigerante Primário por Pequena Ruptura (APRPPR – SBLOCA: *Small Break Loss of Coolant Accident*) ou Acidente de Perda de Refrigerante Primário por Grande Ruptura (APRPGR – LBLOBA: *Large Break Loss of Coolant Accident*) em plantas nucleares de pesquisa e de potência. Além disso, pode simular transientes de perda de potência elétrica, perda de água de alimentação, perda de vazão, entre outros. A análise do comportamento termo-hidráulico durante um desses acidentes ou transientes se aplica tanto para o circuito primário como para o secundário de uma instalação nuclear.

O processo de licenciamento de uma planta nuclear envolve, não apenas a simulação numérica de acidentes e transientes termo-hidráulicos, mas também a qualificação de sua modelagem, do código utilizado e de seus usuários.

Há a necessidade da aplicação dos cálculos de sensibilidade e de incertezas a fim de validar os resultados da simulação de acidentes e transientes termo-hidráulicos de uma instalação nuclear obtidos com o RELAP5. Na literatura, existem algumas metodologias de cálculo de sensibilidade/incerteza, como: *Best Estimate Plus Uncertainty* (BEPU) desenvolvido pela Universidade de Pisa (D'AURIA et al., 2012), e o *Software for Uncertainty and Sensitivity* (SUSA) desenvolvido pela *Global Research for Safety* (GRS) (KLOOS, 2015).

A nodalização de uma instalação nuclear para o código RELAP5 é feita separadamente para cada um dos componentes do reator, tais como: bombas, tubulações, trocadores de calor, acumuladores, válvulas, entre outros. A maioria dos componentes da planta é considerada como cilindros e, mesmo que alguns deles apresentem geometrias diferentes, o usuário deverá trabalhar com um diâmetro equivalente ao do cilindro, mantendo o mesmo volume do componente original.

O modelo hidrodinâmico é baseado em volumes de controle, que são considerados como cilindros com junções de entrada e saída. Propriedades escalares como pressão, energia, densidade e fração de vazio são representadas pela média dentro do volume de controle e estão localizadas no ponto central do mesmo. Por outro lado, propriedades vetoriais, tais como, as velocidades e vazões mássicas, são localizadas nas junções.

O modelo de escoamento do RELAP5 é bifásico, não homogêneo e unidimensional. O modelo de transferência de calor também é baseado numa aproximação unidimensional para o cálculo das temperaturas e fluxos de calor.

O RELAP5 possui seis equações de conservação: duas de massa, duas de energia e duas de quantidade de movimento, ou seja, uma para a fase líquida e outra para o vapor. Possui, também, duas equações adicionais, sendo uma para gases não condensáveis e outra para o boro solúvel. O programa conta com uma aproximação da equação de conservação de movimento para fluxo transversal e um modelo de fluxo reverso, onde esse último utiliza uma solução de condução bidimensional. O método numérico utilizado nesse código é o de diferenças finitas. Além disso, o programa conta com uma aproximação da equação de conservação de movimento para o fluxo transversal, múltiplas junções no núcleo do reator e um modelo de fluxo reverso. As múltiplas junções no núcleo dão uma conotação bidimensional nessa região da planta, sendo que no restante da instalação o escoamento é unidimensional.

O primeiro passo para a simulação de um acidente ou transiente com o RELAP5 é a elaboração da nodalização dos componentes hidrodinâmicos da instalação em estudo, que deverão ser modelados por meio da representação geométrica mais próxima da realidade. Alguns componentes existentes no RELAP5 são: PIPE para tubulações, BRANCH para bifurcações ou T´s, VALVE para válvulas, ACCUM para acumuladores, PUMP para bombas, SEPARATOR para separadores de vapor, entre outros. Além dos dados geométricos da planta, que fazem parte dos dados de entrada do código RELAP5, também são modelados os sistemas de controle da planta e as estruturas de troca de calor.

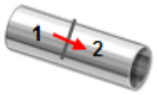
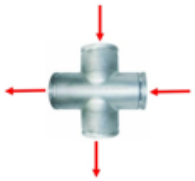
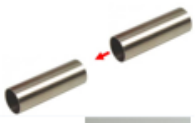
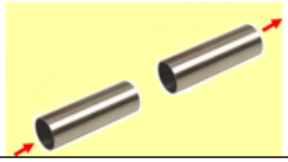
Na Tabela 1 são apresentados alguns dos componentes mais utilizados em simulações de sistemas com o código RELAP5.

De forma geral, o código RELAP5 necessita das seguintes informações:

- os dados geométricos da instalação;
- os dados dos sistemas de controle, por meio de blocos de controle lógico criados pelo usuário;

- as áreas de transferência de calor, representadas por estruturas de troca de calor entre dois ou mais componentes e também com o meio ambiente;
- a cinética do reator ou a potência da instalação, por meio de tabelas fornecidas pelo usuário;
- os dados dos materiais que constituem a planta, tabelas fornecidas pelo usuário ou pela biblioteca interna ao código.

Tabela 1 - Elementos mais utilizados em simulações com o código RELAP5

Tipo	Componente	Nodalização	Descrição
Volume simples (*Single-volume*)	**SNGLVOL**		Representação de algum trecho de tubo sem subdivisão.
Tubulação (*pipe*)	**PIPE**		Representação de algum trecho de tubo com divisão.
Conector com várias saídas (*branch*)	**BRANCH**		Representa uma tubulação com várias junções conectadas a ela.
Junção Simples (*single-junction*)	**SNGLJUN**		Conecta um componente a outro.
Válvula (valve)	**VALVE**		Pode simular a presença de até seis tipos de válvulas.
Bomba (*pump*)	**PUMP**		Simula a presença de uma bomba de refrigeração.
Acumulador *(accumulator)*	**ACCM**		Simula os acumuladores de um reator PWR.
Volume dependente do tempo (*time dependent volume*)	**TMDPVOL**		Representa um volume com condições de contorno.
Junção dependente do tempo (*time dependent junction*)	**TMDPJUN**		Representa junções como condições de contorno.

Fonte: SILVESTRE, L. J. B., 2016.

As estruturas de troca de calor são termicamente conectadas ao volume hidrodinâmico pelo fluxo de calor, que é calculado por meio de uma correlação de transferência de calor por convecção. São usadas para simular paredes de tubos, elementos aquecidos, combustíveis nucleares e qualquer superfície de troca de calor.

O RELAP5 gera uma grande quantidade de informações da planta, quando da simulação de um transiente ou acidente, e a sua visualização pode ser feita por meio de pós-processadores gráficos como ARAUJO FILHO et al. (1994) ou APTPLOT (2019).

Na literatura podem ser encontrados, também, alguns pré-processadores de programas termo-hidráulicos que são descritos a seguir.

2.4 Pré e pós-processadores para códigos termo-hidráulicos

Foram encontrados na literatura alguns pré e pós-processadores matemáticos para os códigos termo-hidráulicos, que estão apresentados abaixo.

Por meio da WESTINGHOUSE (2009), foram encontrados alguns programas de análise termo-hidráulica: o RETRAN-02, o FACTRAN, o OPTOAX, o TWINKLE, o XTRACTR e o NSAPLOT.

A maioria desses programas executa cálculos termo-hidráulicos de plantas nucleares em caso de transientes operacionais. Em particular, o programa RETRAN-02 (ELECTRIC POWER RESEARCH INSTITUTE, 2007), cuja estrutura assemelha-se à do RELAP5, no entanto, possui autoinicialização para o balanço térmico da planta em estudo e para alguns dados geométricos (perdas de carga). Esse código pode ser utilizado como um pré-processador para o RELAP5. No entanto, como fator limitante, esse programa é de uso exclusivo das operadoras de reatores nucleares, como a ELETRONUCLEAR, no caso do Brasil. Por esse motivo, faz-se necessário o desenvolvimento de outra ferramenta para essa finalidade.

Outro programa desenvolvido pela Westinghouse como um pós-processador é o NSAPLOT, que pode ser utilizado facilmente pelo RELAP5 e não possui limitações quanto à utilização.

O TROPIC (2004) é um pré-processador termo-hidráulico do código RELAP5 para a elaboração dos dados de entrada que descrevem os dados geométricos e operacionais de uma planta. Sua criação visou atingir os seguintes objetivos:

- redução no conhecimento do código de sintaxe;
- documentação do conjunto de dados de entrada;
- redução de custos em termos de recursos humanos e custos computacionais.

Outra interface homem-máquina amigável encontrada na literatura para o TRAC-PF1 (SPORE, 1991) foi o SIMulador de ENGenharia (SIMENG) que facilita a preparação dos dados de entrada para esse código (ARAUJO FILHO et al., 1994).

O SIMENG realiza também a análise dos resultados durante e após a simulação. No entanto, esse simulador foi desenvolvido apenas como protótipo e atende somente alguns componentes do circuito primário da planta. Esse protótipo não atende às necessidades dos usuários de outros códigos termo-hidráulicos.

Além desses programas, foram realizados alguns trabalhos no sentido de desenvolver cálculos relacionados com o RELAP5, como a aplicação do programa *MS Excel*® com o código RELAP5 na elaboração da memória de cálculo do reator IRIS (SABUNDJIAN et al., 2004).

O pré-processador matemático mais utilizado é *Symbolic Nuclear Analysis Package* (SNAP) (2017), que apresenta uma interface gráfica simplificando a tarefa do usuário no momento da criação dos dados de entrada para os códigos: TRACE, PARCS (DOWNAR; SEKER, 2010.), *Containment Transient Analysis* (CONTAIN) (MURATA et al., 1997), MELCOR, RELAP5, dentre outros, bem como auxiliar na visualização dos resultados destes códigos. O SNAP, baseado na linguagem Java, possui a facilidade de rodar em plataformas: *Windows XP* e *Vista*, e sistemas baseados em *Linux* e *Mac OS X* (SNAP, 2015).

O SNAP é construído sobre o *Common Application Framework for Engineering Analysis* (CAFEAN), que fornece uma estrutura flexível para a criação e edição dos dados de entrada para os códigos de análise de engenharia. Além disso, possui a funcionalidade de apresentação gráfica dos resultados desses códigos. No entanto, um dos fatores limitantes do programa é a possibilidade de induzir o usuário inexperiente a resultados totalmente equivocados.

Além desses programas computacionais, existem os seguintes pré-processadores desenvolvidos no Brasil:

- PALADINO (2006) propôs a criação de um pré-processador matemático para o RELAP5 (PREREL5) em sua dissertação de Mestrado. Esse processador facilita a utilização do código pelo usuário, diminuindo o tempo de preparação dos seus dados de entrada;
- SILVESTRE (2016) aperfeiçoou o pré-processador de Paladino, acrescentando os componentes hidrodinâmicos pendentes em conformidade

ao Manual do RELAP5, com uma interface mais interativa e exequível para o usuário final. Além disso, permitiu o cálculo das perdas de cargas, utilizando o programa *SF Pressure Drop 6.2* (SOFTWARE-FACTORY NORBERT SCHMITZ, 2010) por meio de um link no pré-processador (PCRELAP5), pois essas informações são parte dos dados de entrada dos componentes do RELAP5.

3 MATERIAIS E MÉTODOS

Esse trabalho utilizou uma série de ferramentas computacionais para o desenvolvimento do pré-processador ora apresentado, as quais estão detalhadas nesse item do trabalho.

3.1 Ferramentas utilizadas

São apresentadas a seguir as ferramentas utilizadas para o desenvolvimento do pré-processador proposto.

3.1.1 *MS Excel*®

O *MS Excel*® (MICROSOFT EXCEL, 2019) é um *software* de criação de planilhas numéricas eletrônicas, criado em 1985, para os computadores *Macintosh*, e posteriormente adaptado ao ambiente *Windows* em 1987 (ENCICLOPÉDIA BRITANICA, 2019). O *MS Excel*® foi criado com objetivo de ser uma alternativa melhorada ao *software Lotus 1-2-3*. O *Lotus 1-2-3*, criado pela *Lotus Software* (posteriormente adquirida pela IBM) foi um dos primeiros programas de criação de planilhas eletrônicas para o ambiente *MS-DOS*, que dominou esse mercado em meados de 1980 (ENCICLOPÉDIA BRITANICA, 2019).

O *MS Excel*® é atualmente uma das melhores plataformas para criação de planilhas eletrônicas. Seus recursos incluem uma interface intuitiva, ferramentas de cálculos poderosas e construção de gráficos, uma vasta gama de funções numéricas para diversas áreas de aplicação, como finanças, estatística e engenharia, ferramentas para importação de dados a partir de outros arquivos, uma poderosa linguagem para automação de tarefas, que é o *Visual Basic for Application* (VBA), recursos multiusuário e de compartilhamento de planilha, e muito mais. Todos esses recursos ajudaram a tornar o *MS Excel*® um dos mais populares *softwares* de computador até hoje.

O *MS Excel*®, assim como todo o pacote *MS Office*®, possui forte apelo à parte estética de seus documentos, tais como tamanho das letras, cores de fonte e de fundo, bordas e outras aparências das células. Também introduziu o conceito de recálculo inteligente, onde somente as células dependentes das células modificadas são recalculadas (MICROSOFT, 2017), o que o tornou na ocasião um *software* mais rápido e poderoso que seus concorrentes, uma vez que as planilhas de cálculo na época, ou recalculavam sempre todos os dados, ou então ficavam aguardando um comando do usuário para efetuar o recálculo das fórmulas.

Em sua versão atual, a capacidade que o *MS Excel*® oferece em termos de processamento de dados é tão grande que existem sistemas inteiros construídos usando planilhas *MS Excel*®, desde as telas de entrada de dados, bem como todo o repositório de dados em planilhas; tudo isso devido à quantidade de recursos que a ferramenta oferece, bem como à possibilidade de se automatizar praticamente qualquer tarefa por meio da linguagem VBA, embutida na ferramenta.

Além disso, a importância do *MS Excel*® atualmente no mundo corporativo é grande, tanto que praticamente todos os sistemas *Enterprise Resource Planning* (ERP) existentes possuem algum nível de interação com o *MS Excel*®.

Atualmente, o *MS Excel*® está na versão 2019, e faz parte do pacote de ferramentas da *Microsoft* chamado *Office* 365®, vendido como um serviço atualizável por meio da nuvem, com assinatura anual para empresas e para uso doméstico, bem como para escolas e entidades sem fins lucrativos, garantindo que o usuário tenha sempre a versão mais atual disponível para uso. Por meio desse serviço, as atualizações do programa são constantes, e grandes mudanças de versão têm ocorrido a cada dois ou três anos. O pacote *Office* inclui vários tipos de programas para escritórios, como criação de documentos, folhas de cálculo, agenda, correio eletrônico e apresentações.

3.1.2 VBA

O *MS Excel*®, assim como toda a linha *MS Office*®, oferece um ambiente de programação conhecido como *Microsoft Visual Basic for Applications*, ou VBA (BATTISTI, 2013). Essa é uma linguagem de programação com sintaxe muito similar à do *MS Visual Basic*®, porém com algumas peculiaridades.

O VBA, assim como o *Visual Basic*®, não é uma linguagem orientada a objeto. Como pode ser visto no APÊNDICE J, apesar do VBA permitir a definição de classes e a criação de instâncias dessas como objetos, ele não possui formas de tratar herança de objetos, sobreposição ou mesmo sobrecarga de métodos, não implementando, portanto, o polimorfismo de maneira satisfatória. Além disso, os tipos de dados primários, tais como inteiros, *strings*, datas, números reais ponto-flutuantes, entre outros, não são implementados como objetos, como ocorre na linguagem Java®, por exemplo. E, para compensar a carência de alguns atributos da orientação a objetos, tanto o VBA como o *MS Visual Basic*® fazem uso de alguns artifícios, como o tipo de dados "*Variant*", que nada mais é que um tipo de variável que aceita qualquer tipo de dado, e pode ser tratado dentro da programação conforme o tipo de dado passado pela função que a chamou. Para declarar uma variável

desse tipo, basta que o programador a declare com o tipo "*Variant*", ou a declare sem informar nenhum tipo, ou até mesmo não a declare: nesse último caso o VBA irá identificar que a variável em questão nunca foi usada, e irá automaticamente declará-la como sendo do tipo "*Variant*". Se por um lado esse tipo de variável diminui parte dos problemas com sobreposição ou sobrecarga de métodos, por outro lado, o mesmo pode provocar problemas no programa desenvolvido que são difíceis de serem identificados, por não gerarem erros em tempo de compilação caso a variável seja tratada pelo programa de forma incompatível à sua natureza.

Assim como o *Visual Basic*, o VBA é uma linguagem orientada a eventos, e essa característica pode ser facilmente observada em seu ambiente de programação. A implementação de programação para o tratamento de eventos dos objetos é feita de forma muito simples.

O VBA também não possui uma forma eficiente de tratamento de erros; para tal, ele faz uso de estruturas de tratamento de erros muito similares a de outras linguagens de programação modulares, fazendo uso de desvios incondicionais (GO TO), artificio esse criticado até mesmo pelos puristas da programação estruturada.

Um programa no VBA pode ser escrito dentro de um módulo funcional, um módulo de classe customizado ou de um módulo de classe com objetos nativos (tais como telas, planilhas de trabalho, pastas de trabalho, entre outros).

Todavia, o VBA permite de forma relativamente simples o acesso direto às *Application Programming Interfaces* (API) do *Windows*®, utilizando-se para isso do protótipo das mesmas em módulos funcionais, e acesso a bibliotecas de programas no padrão COM/DCOM, largamente utilizado pela *Microsoft*® nos últimos anos, e ainda com muitas bibliotecas publicadas, estendendo em muito o poder da linguagem VBA. Entretanto, o modelo COM/DCOM atualmente está em desuso, pela evolução da plataforma *Microsoft*® *.NET*®.

E, por fim, diferentemente do *MS Visual Basic*®, os programas escritos em VBA não são compilados, ou seja, não geram um programa separado; essas linhas de programação serão interpretadas todas as vezes em que forem chamadas; por um lado, isso facilita a depuração, mas, por outro, pode afetar o desempenho do programa elaborado.

Além disso, o *MS Excel*® vem com uma funcionalidade chamada "Gravação de Macros", que nada mais é que a criação automática de programação VBA que reproduzem as ações que os usuários vão fazendo em tela. Dessa forma, é possível criar programas para automatizar praticamente qualquer tarefa realizada dentro do *MS Excel*®. Vale ressaltar,

porém, que as linhas de programação geradas automaticamente pelo gravador de macros precisam ser revisadas para que possam ser utilizadas de forma efetiva. Ainda assim, esse é um recurso que ajuda bastante, pois reduz em muito a curva de aprendizado do modelo de objetos utilizado na programação em VBA do *MS Excel*®.

3.1.3 A plataforma de desenvolvimento *Microsoft*® *.NET*® *Framework*

Devido a evolução tecnológica iniciada com o advento da internet comercial, uma série de novas aplicações e dispositivos foram lançados a partir do início do século XXI. Com isso, a família de sistemas *Windows*® tornou-se extremamente vasta e heterogênea, envolvendo sistemas para servidores computacionais de missão crítica, servidores de aplicações, servidores de banco de dados, servidores corporativos, estações de trabalho *desktop* e *laptops*, dispositivos leves (*thin client*), bem como telefones, *smartphones*, dispositivos móveis integrados à internet, televisores, entre outros. Cada um destes dispositivos possui processadores peculiares, dispositivos anexados e formas peculiares de uso da memória (MICROSOFT .NET FRAMEWORK, 2019).

Essa heterogeneidade no ambiente de desenvolvimento *Windows*® preocupava a *Microsoft*®, que estudava uma forma de permitir que o desenvolvimento de aplicações para *Windows*® pudesse ser isolado em uma camada onde o desenvolvedor de programas pudesse ter uma curva de aprendizado reduzida para criar novas aplicações para diferentes plataformas *Windows*®. E foi dessa necessidade que surgiu o *.NET*® *Framework* (lê-se dót-net).

Lançado pela *Microsoft*® em 2002, e sendo atualizado constantemente desde então, o *.NET*® *Framework* foi uma iniciativa da *Microsoft*® com objetivo de prover uma plataforma única para desenvolvimento e execução de aplicações para toda a família *Windows*®, não importando se o programa rodaria num servidor de missão crítica, num *smartphone*, num *laptop* ou num televisor. A partir daquele momento, todo e qualquer programa gerado para *.NET*® poderia ser executado em qualquer dispositivo que possuísse um *.NET*® *Framework* para si.

Assim, o programador não precisaria se preocupar em escrever o programa para um determinado *hardware* ou dispositivo, mas escreveria programas para a plataforma *.NET*®. Essas aplicações, por sua vez, passariam a rodar num ambiente de execução gerenciado, que fornece os serviços do sistema, gerenciamento de memória, gerenciamento de pilha, gerenciamento de *threads* (multitarefa), uma extensa biblioteca de classes,

permitindo que os programadores usem sub-rotinas robustas e confiáveis para todas as áreas principais do desenvolvimento de *softwares*.

O *.NET® Framework* está disponível somente para sistemas *Windows®*. Entretanto, existe uma implementação de parte do *.NET*, chamada de *.NET Core* (MICROSOFT .NET CORE, 2019), para execução de programas *.NET®* no *Windows*, *MacOS* e *Linux*. O *.NET Core* é uma implementação de *software* livre, modular, de plataforma cruzada e para fins gerais do *.NET®*. Ele contém as mesmas *Application Programming Interfaces* (APIs) do *.NET® Framework* (porém, um conjunto reduzido das mesmas) e inclui componentes de execução, estrutura, compilador e ferramentas que oferecem suporte a vários sistemas operacionais e processadores.

O *.NET® Framework* consiste em dois componentes principais: o *Common Language Runtime* (CLR), que é o mecanismo de execução que manipula programas em execução, e a biblioteca de classes *.NET® Framework*, que oferece uma biblioteca de rotinas robustas, confiáveis, testadas e reutilizáveis, que os desenvolvedores podem chamar dentro de seus próprios *softwares*. Entre os serviços que o *.NET® Framework* oferece, tem-se os seguintes:

- gerenciamento de memória: em muitas linguagens de programação, os programadores são responsáveis por alocar e liberar memória, e por identificar o tempo de vida dos objetos. Em programas *.NET®*, o CLR fornece esses serviços em nome do *software*;
- sistema de tipos básicos: em linguagens de programação tradicionais, os tipos básicos são definidos pelo compilador, fato esse que dificulta a interoperabilidade entre diferentes linguagens de programação. No *.NET® Framework*, os tipos básicos são definidos pelo sistema de tipos do *.NET® Framework* e são comuns a todas as linguagens que segmentam o *.NET® Framework*;
- biblioteca de classes abrangente: ao invés de precisar gerar um vasto volume de linhas de programação para manipular operações de programação comuns de baixo nível, os desenvolvedores usam uma biblioteca facilmente acessível a partir do *.NET® Framework*;
- estruturas e tecnologias de desenvolvimento: o *.NET® Framework* inclui bibliotecas para áreas específicas do desenvolvimento de *softwares*, como o *ASP.NET®* para aplicações *web*, o *ADO.NET* para acesso a banco de dados, o

Windows Communication Foundation para programas orientados a serviços e o *Windows Presentation Foundation* para programas de área de trabalho *Windows*®;

- interoperabilidade de linguagens: os compiladores de linguagem integrados ao *.NET® Framework* geram o *software* em uma linguagem intermediária, chamada de *Common Intermediate Language* (CIL). Esse, por sua vez, é compilado em tempo de execução pela *Common Language Runtime* (CLR). Com esse recurso, as rotinas compiladas em uma determinada linguagem são plenamente acessíveis por softwares criados em outras linguagens, e os programadores podem focar na criação de software em sua linguagem preferida. Esse conceito certamente não era novo: o Java®, criado pela *Sun® Microsystems* já possuía conceito similar de gerar um *software* em linguagem intermediária que seria compilado em tempo de execução e executado pela *Java® Virtual Machine* (JVM) sob demanda. A grande revolução da *Microsoft*® nesse aspecto foi fazer com que a CLR não apenas compilasse o *software* intermediário CIL sob demanda para executá-lo na máquina destino, mas compilasse-o em tempo de execução, fazendo com que o desempenho dos programas criados para *.NET*® fossem superiores a programas criados em Java®, uma vez que o tempo de aquecimento (*warm up*) do JVM é maior que o do CLR (SINGER, 2003). Contudo, hoje em dia, essa diferença de desempenho foi reduzida drasticamente;
- compatibilidade de versões: os *softwares* desenvolvidos em uma versão específica do *.NET® Framework* são executados sem modificações em uma versão posterior, sendo sequer necessário recompilá-los (com raríssimas exceções);
- execuções lado a lado: nos casos em que possam ocorrer exceções, como as mencionadas no item anterior, o *.NET® Framework* ajuda a resolver esse tipo de conflito de versão permitindo que várias versões do CLR existam mutuamente num mesmo computador. Isso significa também que várias versões dos *softwares* podem coexistir, e que um *software* pode ser executado na versão para o qual foi compilado;
- independência de linguagem de programação: como todo compilador *.NET*® gera, na verdade, um *software* escrito em CIL, que será compilado para

> linguagem de máquina em tempo de execução pela CLR do dispositivo hospedeiro. O *.NET® Framework* oferece ao programador uma série de linguagens que podem ser escolhidas para o desenvolvimento de suas aplicações, permitindo inclusive que várias linguagens possam ser misturadas no desenvolvimento de *softwares* complexos, reduzindo a curva de aprendizado por profissionais do mercado. O *.NET® Framework* oferece atualmente suporte a diversas linguagens de programação que irão gerar programas *.NET®*, tais como *Visual Basic® .NET*, C++, F#, *Delphi Prism* (que possui sintaxe muito similar ao Delphi®, da Borland®), JavaScript e mesmo *Python*.

As versões mais recentes de sistema operacional *Windows®*, como o *Windows®* 8 e posteriores, tanto para estações de trabalho como para servidores, já vem com uma versão de *.NET® Framework* instalada, permitindo ao usuário, que não desenvolve programas e que somente os consome, não precise ter nenhum conhecimento sobre o *.NET® Framework* ou seu funcionamento. Caso o *.NET® Framework* não esteja instalado, o usuário que tentar executar um programa escrito para *.NET® Framework* se deparará com uma caixa de diálogo solicitando a instalação do *.NET® Framework*, que pode ser facilmente baixado e instalado de forma gratuita a partir da internet.

3.1.4 A linguagem de programação C#

A partir da criação do *.NET® Framework* em 2002, a *Microsoft®* introduziu com ele uma linguagem elegante, totalmente orientada a objetos e fortemente tipada, que permite aos desenvolvedores criar uma variedade de programas robustos e seguros para o *.NET® Framework*, chamada de C# (lê-se cê-xarp, do inglês *C-sharp*, ou, em tradução literal, C-sustenido). O sustenido foi escolhido para nomear essa linguagem por ser um sinal musical que aumenta o tom da nota musical em meio tom. A ideia foi dar a impressão de que o C# era a linguagem C/C++ subindo de tom (KOVACS, J., 2007).

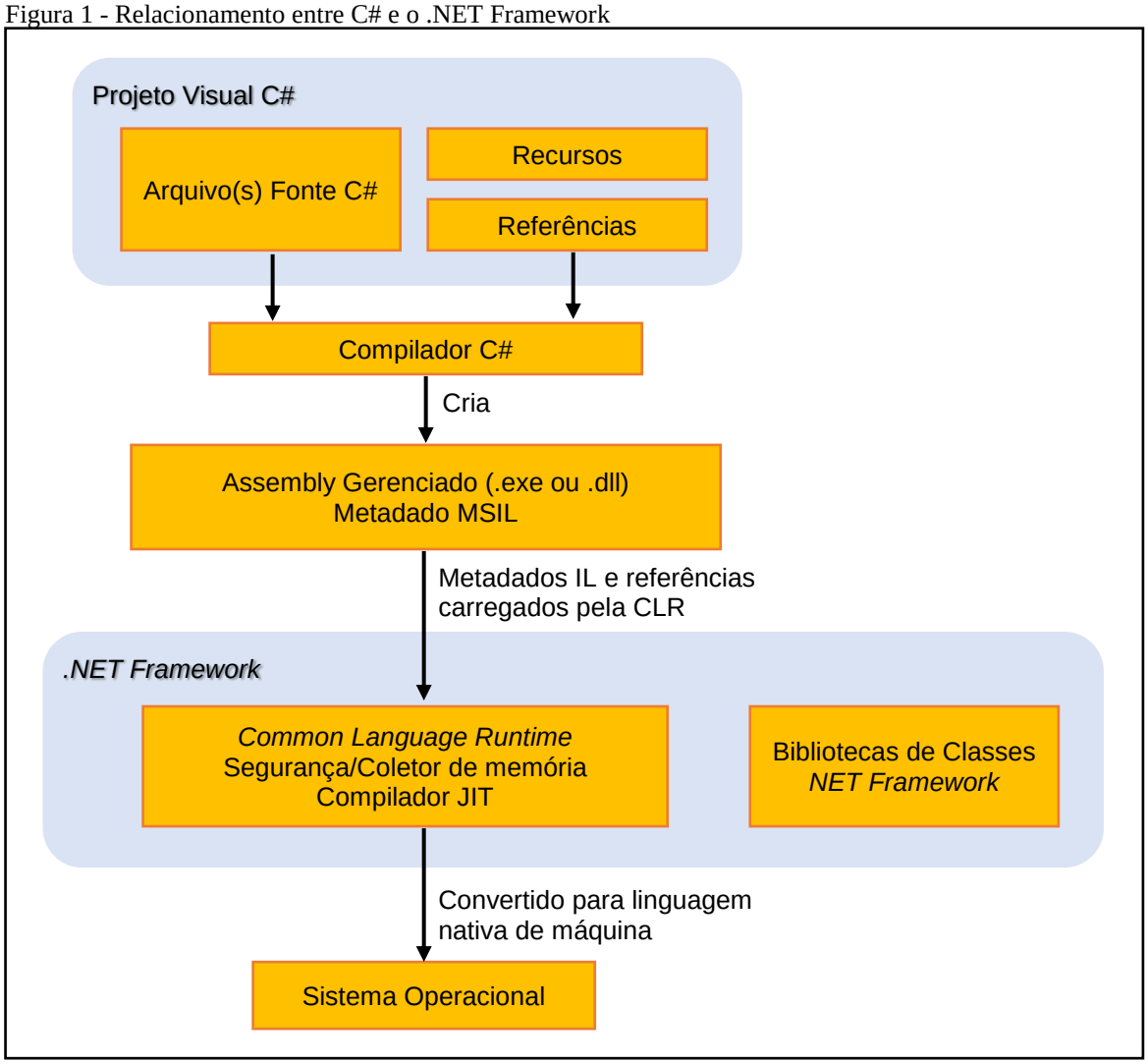
A sintaxe de C# é fortemente inspirada nas linguagens C, C++, mas também possui muitas influências de outras linguagens, como o Java®.

Como características principais, o C# ofereceu à comunidade desenvolvedora uma linguagem plenamente orientada a objetos, que alia o poder do C++ com a simplicidade do *Visual Basic®*. No C# todos os objetos e tipos de dados são objetos derivados da classe *Object*. O C# também oferece estruturas, enumerações, classes, interfaces, delegados,

tipagem estática, sistemas uniformes de exceção, eventos, sistema de versões e documentação de programação.

Por fim, mas não menos importante, o C# oferece interoperabilidade de programa nativo e de componentes COM®, ou mesmo a *Dynamic Link Library* (DLLs) nativas do *WIN32*, por meio de um processo chamado *Interop* (MICROSOFT INTEROP, 2015). Essa característica foi fundamental para o desenvolvimento de criação desse pré-processador, devido ao fato de todo o pacote *Microsoft® Office®* oferecer automação de suas tarefas a partir de bibliotecas COM®, ficando todas disponíveis para soluções em C# por meio do *Interop*, assim como ocorre com o VBA (MICROSOFT VBA, 2018). A Figura 1 mostra o relacionamento entre o C# e a plataforma *.NET*.

Figura 1 - Relacionamento entre C# e o .NET Framework



Fonte: MICROSOFT, 2015.

Por ser uma linguagem totalmente orientada a objeto, os programas escritos com C# se beneficiam fortemente das principais características desse tipo de programação. E um

dos principais benefícios da orientação a objetos é a “herança”. A herança é uma forma de reutilização da programação em que as classes são criadas pela absorção e comportamentos de uma classe existente, podendo aprimorá-las com novas funcionalidades. Ao criar uma nova classe, ao invés de reescrever propriedades e métodos comuns, o programador indica que essa classe herdará as variáveis, propriedades, métodos e comportamentos de outra classe. Essa classe fica conhecida como herdeira ou derivada da classe base, representando assim um grupo mais especializado de objetos da classe base. Diferentemente do C++, o C# não suporta herança múltipla; assim, uma classe derivada pode ter somente uma classe base direta.

Outra característica da orientação a objeto muito presente no C# é o polimorfismo. O C# permite que se acesse um objeto como se ele fosse de outro tipo, de acordo com o contrato estabelecido, de maneira polimórfica. As classes podem ser abstratas e exigirem que sejam derivadas, para que suas classes e métodos não sejam obrigatoriamente implementados na classe base, deixando que as classes derivadas implementem esses métodos.

3.1.5 O ambiente de desenvolvimento *Microsoft® Visual Studio® Community*

O *Microsoft® Visual Studio®* (MICROSOFT .NET FRAMEWORK, 2019) é um *Integrated Development Environment* (IDE) feito pela *Microsoft®*, que permite aos desenvolvedores criar programas de computador, além de programas para internet, *websites*, interfaces programáticas para internet (*Web Services*), e também programas para plataformas móveis. O *MS Visual Studio®* inclui um editor de programas com suporte à tecnologia *IntelliSense* (um componente que completa trechos da programação escrita, além de vários outros recursos, como listar membros, informações de parâmetros e variáveis, informação rápida, entre outros), uma ferramenta integrada de depuração de *software* com capacidade de depuração em nível de código e em nível de máquina, ferramenta para desenho de telas, desenho de classes, e modelagem de banco de dados. O *MS Visual Studio®* também suporta complementos que aumentam sua funcionalidade para qualquer nível, tais como suporte a ferramentas de versionamento de programa, bem como ferramentas para controle de todo o ciclo de vida do desenvolvimento.

Por ser o ambiente de desenvolvimento criado pela empresa criadora do *.NET® Framework*, o *MS Visual Studio®* suporta diferentes linguagens de programação, e seu editor de programas e depurador suportam, em diferentes graus, todas as diferentes linguagens oferecidas.

Algumas das linguagens padrão oferecidas pelo *MS Visual Studio*® incluem C, C++, VB.NET, C#, F#, JavaScript, XML, HTML e CSS. Outras linguagens, tais como *Python* e *Ruby* são suportadas por meio de complementos.

A edição mais básica do *MS Visual Studio*®, a edição Community, é oferecida pela *Microsoft* sem custo para desenvolvimento de complementos para o *MS Visual Studio*®, para desenvolvimento de *drivers* de dispositivos para *Windows*, para desenvolvimento e depuração de programas de código livre, e também para iniciativas de treinamento, educação, bem como de pesquisa e desenvolvimento.

Uma das principais características da nova versão do *MS Visual Studio*® foi a integração das ferramentas de programação de soluções para *MS Office*®, conhecida antigamente como *Visual Studio Tools for Office* (VSTO). Apesar de até hoje o pacote *MS Office*® oferecer suporte para automação de tarefas embutido por meio da linguagem VBA, foi a partir da versão do *MS Office*® 2003 que a *Microsoft* começou a oferecer à comunidade de desenvolvedores de soluções para *MS Office*® a possibilidade de se desenvolver programas que pudessem ser compilados usando toda a potencialidade fornecida pela plataforma *.NET*®, e que também pudessem se beneficiar da capacidade de automação de tarefas dentro do pacote *MS Office*®. Nasceu assim o VSTO, agora embutido na mais nova versão do *MS Visual Studio*®.

A vantagem disso é que, agora, o *MS Visual Studio*® permite que os programas para automação do *MS Office*® possam:

- ser compilados, ao invés de serem sempre interpretados, permitindo assim um ganho de desempenho nas soluções para *MS Office*®;
- ser armazenados fora dos documentos, diferentemente do que ocorre com o VBA, onde a programação fica sempre armazenada dentro do documento;
- ter mais um grau de liberdade na escolha da linguagem de programação: enquanto o VBA oferece apenas uma única linguagem, que é chamada de VBA, o VSTO oferece suporte para VB.NET e C#, ambas linguagens totalmente orientadas a objeto;
- ser beneficiadas de todas as caraterísticas dos programas escritos para a plataforma *.NET;*
- utilizar as macros gravadas para VBA, facilmente reescritas para VB.NET ou C#;

- ser submetidos ao controle de segurança de acesso da CLR, além do modelo de permissão de assinatura digital que governa as macros VBA.

Desta forma, as principais desvantagens do uso de soluções baseadas em VBA simplesmente desaparecem com sua substituição por soluções para *MS Office*® escritas usando-se o VSTO integrado ao *MS Visual Studio*® *Community* 2017.

E, assim como o VBA, o VSTO oferece aos programas desenvolvidos, usando-se C# ou VB.NET, acesso completo a todo modelo de objetos do *MS Office*®, permitindo que qualquer tipo de operação feita por um usuário na interface gráfica de um programas do pacote *MS Office*® possa ser também feita programaticamente via VSTO, assim como ocorre no VBA.

3.2 Desenvolvimento do trabalho

O desenvolvimento desse trabalho deu-se em diversas etapas, que serão detalhadas a seguir. As etapas apresentam uma divisão lógica de unidades de trabalho. Devido à complexidade inerente ao código RELAP5 e à modelagem de acidentes termo-hidráulicos, fez-se necessário um grupamento lógico das atividades realizadas para conclusão desse trabalho, que serão mais bem descritas nos itens a seguir. Apesar das atividades terem forte dependência dos resultados das atividades que as precederam, as realizações destas não necessariamente respeitaram essa dependência de forma temporal: muitas das descritas abaixo ocorreram em paralelo com o andamento de outras; como exemplo disso, são mencionadas as etapas de estudo da ferramenta RELAP5 e de programação do pré-processador.

3.2.1 Revisão do Código RELAP5

Nessa etapa foi realizada a familiarização com o código RELAP5, de forma mais geral, estudando os principais conceitos envolvendo modelagem de problemas termo-hidráulicos. Também foram estudadas outras ferramentas auxiliares para a análise do código RELAP5 e dos seus resultados, tais como APTPLOT (2015), SNAP, PREREL5 e PCRELAP5.

Essa etapa ocorreu de forma preliminar antecedendo o desenvolvimento do protótipo feito nesse trabalho, sendo que o manual do RELAP5 foi o principal documento estudado.

3.2.2 Desenvolvimento do protótipo do pré-processador

A fim de verificar a viabilidade do desenvolvimento de um pré-processador visual para o RELAP5 usando como plataforma o *MS Excel*®, foram feitos alguns testes. Tal protótipo não foi idealizado para contemplar todas as regras de validação necessárias para criação precisa de dados de entrada para o código RELAP5, mas tão somente para verificar a viabilidade de manipulação de gráficos, objeto desse trabalho.

Nessa etapa foram desenvolvidos dois protótipos:

- o protótipo desenvolvido em VBA embutido no documento *MS Excel*® e com moderna interface de Faixa de Opções *Ribbon*®;
- o protótipo de complemento (*add-in*) para *MS Excel*®, desenvolvido em C#, com moderna interface *Ribbon*®.

Os resultados obtidos a partir da análise de ambos os protótipos são discutidos no item 4.1, desse trabalho.

3.2.3 Simulação da "Experiência SUPER CANON" com o RELAP5

A fim de se ter uma base de comparação entre a simulação com o código RELAP5 com os dados de entrada gerados pelo usuário e com os obtidos utilizando o pré-processador, foi realizada a simulação da "Experiência SUPER CANON" (SABUNDJIAN et. al., 1986) com a versão do código RELAP5/MOD3.3. A "Experiência SUPER CANON" foi escolhida pelos seguintes motivos:

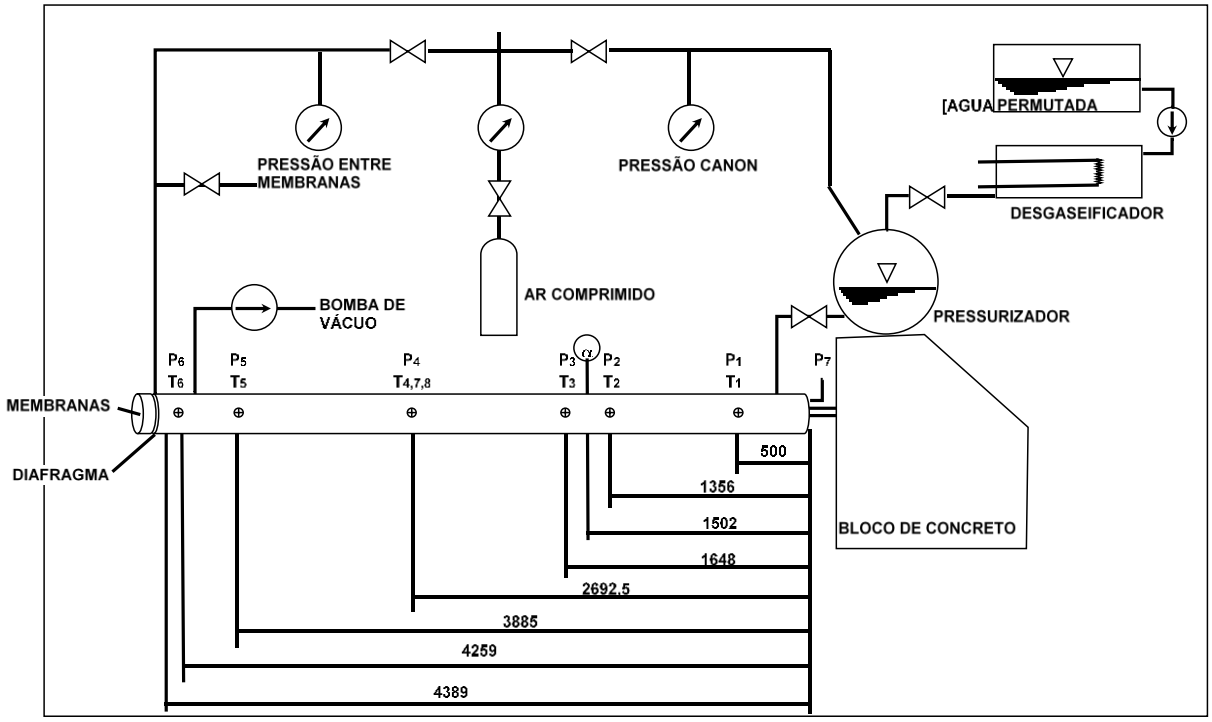
- trata-se de um experimento que pode ser simulado no RELAP5 considerando os componentes hidráulicos e as estruturas de transferência de calor;
- trata-se de um experimento de configuração menos complexa, uma vez que poucos elementos hidráulicos se fazem necessários para sua simulação; pois o RELAP5 possui uma grande quantidade de componentes, por esse motivo a familiarização com esse código precisa ocorrer com problemas de grau de complexidade menor e a "Experiência SUPER CANON" mostrou-se adequada para essa etapa;
- a "Experiência SUPER CANON" possui resultados experimentais publicados (SABUNDJIAN, 1986), o que permitiu a sua comparação com os resultados obtidos na simulação no RELAP5, tanto com os dados de entrada gerados pelo pré-processador quanto com os construídos manualmente;

além disso, essa etapa foi fundamental para o estudo de todos os passos necessários por parte do usuário do RELAP5 tais como: a nodalização, a preparação dos dados de entrada para o RELAP5, permitindo que a versão final do pré-processador disponibilizasse ao usuário uma maior rapidez para a realização dessas tarefas.

Além da criação dos dados de entrada do código RELAP5 para a "Experiência SUPER CANON", foi gerada a nodalização do problema num diagrama feito em planilha *MS Excel*®. Assim, pôde-se identificar as principais notações e desafios vivenciados pelos usuários de RELAP5 no uso do mesmo.

A Figura 2 apresenta o aparato experimental da "Experiência SUPER CANON".

Figura 2 - Esquema da experiência SUPER CANON, incluindo pontos de medida



Fonte: SABUNDJIAN et al., 1986.

3.2.4 Programação do pré-processador

A programação do pré-processador foi certamente a etapa mais longa. Essa etapa ocorreu em paralelo com uma revisão aprofundada do RELAP5, de forma a gerar subsídios conceituais para o desenvolvimento de um pré-processador robusto, capaz de reduzir a complexidade e esforço no desenvolvimento de problemas para o código RELAP5.

O arquivo de dados de entrada do RELAP5 é extremamente objetivo e orientado a dados, deixando pouco ou nenhum espaço para metadados sobre o problema, com exceção

dos cartões de comentários, o que dificulta muito a revisão de problemas existentes mesmo para usuários experientes.

Com base nisso, optou-se por desenvolver um pré-processador que pudesse gerar tanto a nodalização quanto os dados de entrada da maneira mais amigável possível. Apesar do conhecimento do código RELAP5 ser fundamental para o uso adequado do pré-processador, a forma com que as interfaces do pré-processador foram modeladas fazem com que seja mais fácil identificar quais os parâmetros necessários para seu uso correto.

A ferramenta também direciona os usuários do RELAP5 a manterem metadados para facilitar a análise de problemas no futuro. Apesar de todos os metadados gerados nos dados de entrada do RELAP5 serem guardados por meio de comentários, esses são armazenados de forma a serem reaproveitados pelo *add-in*. Informações como observações, versão, data de criação, nome do autor, descrição dos componentes, entre outros, que originalmente não fazem parte dos dados de entrada do RELAP5, formam um conjunto que facilita o estudo dos dados de entrada gerados por meio do pré-processador.

Os requisitos de interface escolhidos foram decididos utilizando-se das melhores práticas de modelagem de experiência de usuários. Assim, o resultado final é um pré-processador com interface fluida, não-intrusiva e não bloqueante, permitindo ao usuário decidir o que fazer a cada momento, e dando condições para que quaisquer usuários com experiência no RELAP5 possam criar novos problemas para o código por meio do pré-processador com curva de aprendizado reduzida, quando comparada com ferramentas similares. O desafio de aprender a criar novos problemas, fazendo uso deste pré-processador, não poderia ser superior ao desafio de se familiarizar com o código RELAP5.

Por fim, a criação dos diagramas visuais foi feita de forma a permitir que os usuários possam criar intuitivamente os diagramas dentro do problema.

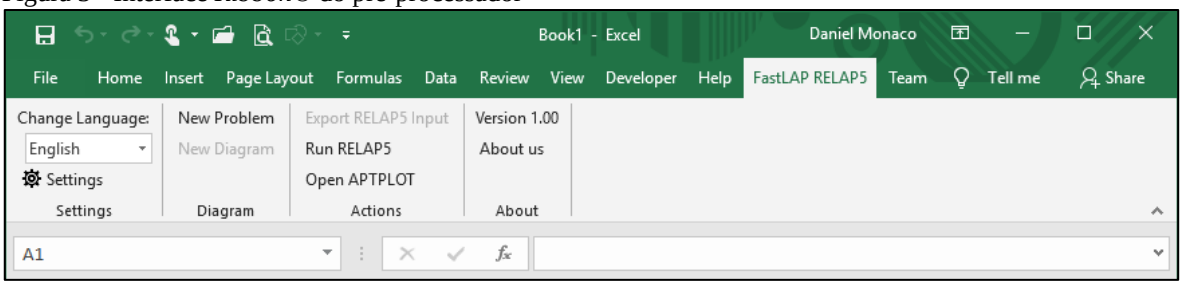
Nos itens a seguir serão apresentadas as principais janelas de entrada de dados do pré-processador desenvolvido. Para mais detalhes sobre a programação dos mesmos, consulte do APÊNDICE A ao APÊNDICE F.

3.2.4.1 A interface *Ribbon*® do pré-processador

Assim que o pré-processador for instalado no computador do usuário e o *MS Excel*® for iniciado novamente, o usuário perceberá que um novo menu será listado junto aos demais menus aos quais ele vê usualmente. Ao clicar sobre ele, o usuário poderá visualizar uma nova interface *Ribbon*®, nos mesmos padrões que as demais funcionalidades do *MS Excel*®, porém com os recursos criados para uso do pré-processador. É a partir desse

menu que o usuário poderá iniciar o uso do pré-processador, conforme ilustrado na Figura 3.

Figura 3 - Interface *Ribbon*® do pré-processador

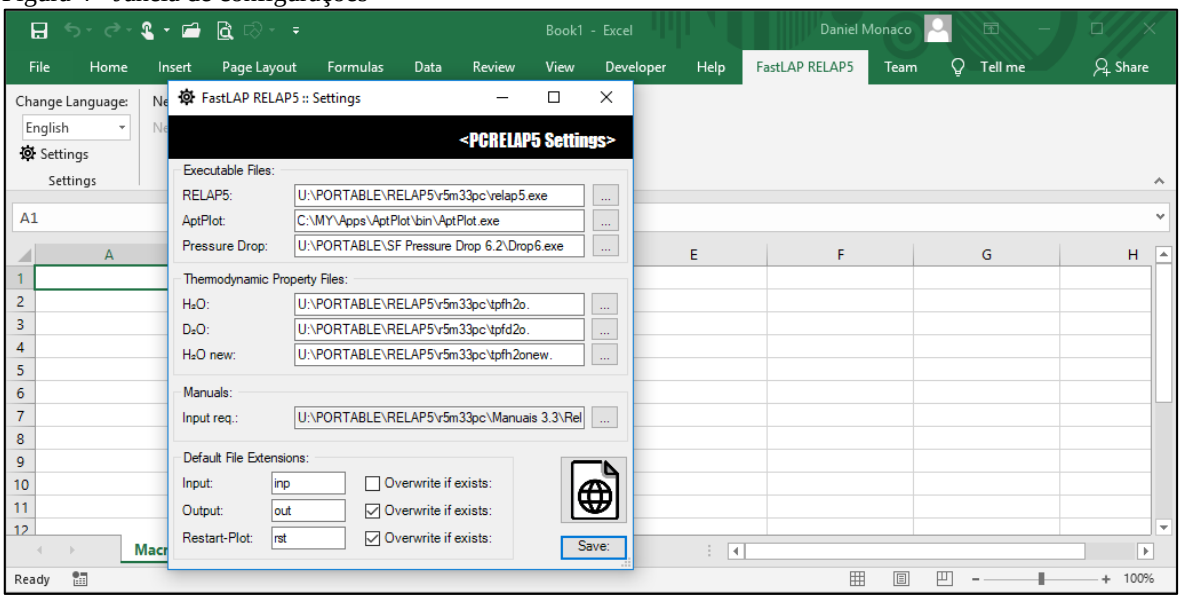


Fonte: autor da dissertação.

Na Figura 3 pode-se ver que na parte mais à esquerda da *Ribbon*® o usuário poderá escolher um dos idiomas disponíveis para trabalhar. Inicialmente, optou-se por distribuir o pré-processador somente em inglês, por ser o mesmo idioma do código RELAP5, e todas as imagens apresentadas neste trabalho foram coletadas usando esse idioma. Para mais informações sobre como obter novos idiomas para o pré-processador, consulte o item 3.2.4.8.

Logo abaixo da lista de idiomas, um botão de configurações ("*Settings*") está disponível, permitindo que o usuário configure o caminho das ferramentas que podem ser integradas ao pré-processador, bem como efetue outras configurações de preferência de extensão de arquivos de saída, conforme exibido na Figura 4.

Figura 4 - Janela de configurações



Fonte: autor da dissertação.

As ferramentas que podem ser integradas ao pré-processador são: o código RELAP5, o *AptPlot*, e o *SF Pressure Drop*®. Dessa forma, o acesso a essas ferramentas fica facilitado durante a criação dos diagramas através do pré-processador.

Imediatamente ao lado desse primeiro grupo de opções é disponibilizado um segundo grupo, onde pode ser criado um novo arquivo de problema RELAP5. O arquivo de problema RELAP5 nada mais é que uma planilha comum do *MS Excel*®; porém, sua formatação especial de abas e células permitem ao pré-processador identificar se tratar de uma planilha apta para receber os diagramas de nodalização.

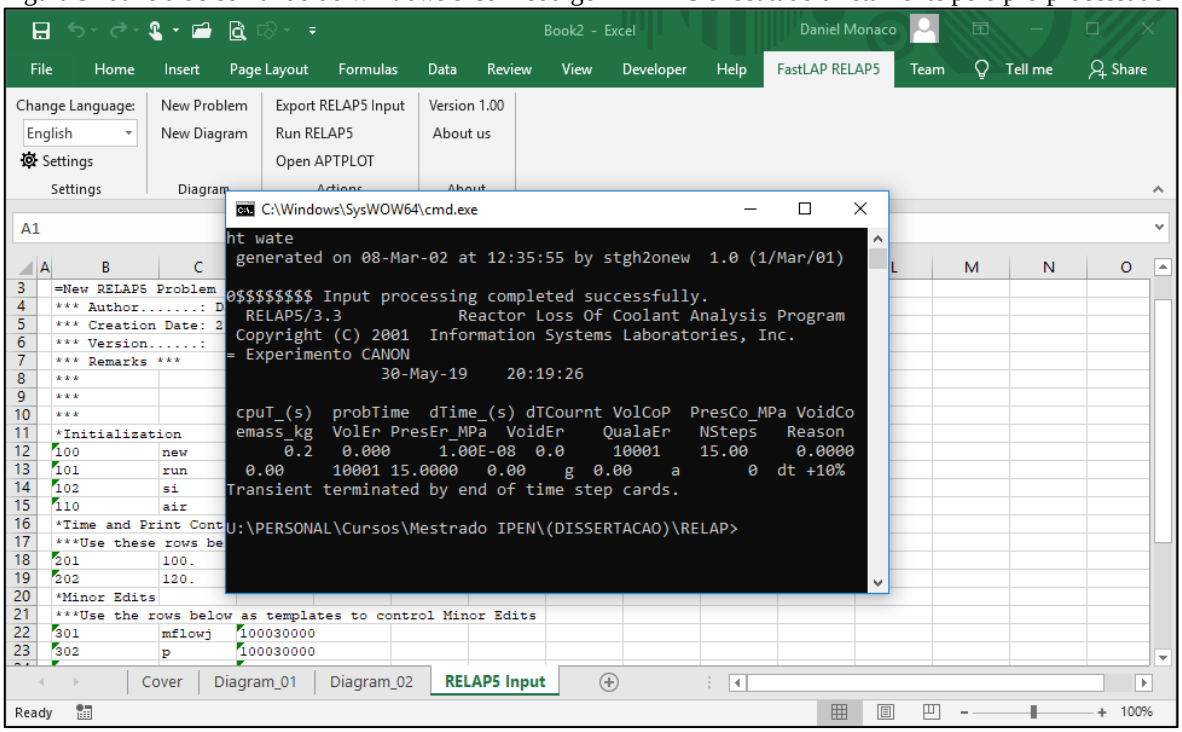
Enquanto o pré-processador não identificar que uma nova planilha problema foi criada, as funcionalidades de desenho de componentes termo-hidráulicos permanecem ocultas aos usuários, seguindo o padrão de interface já adotado pelo *MS Excel*®, por exemplo, na manipulação de imagens ou tabelas dinâmicas.

O grupo seguinte de funcionalidades referem-se a atalhos de acesso aos aplicativos, facilitando a execução do RELAP5 com os dados de entrada gerados, permitindo a exportação dos dados de entrada para um arquivo externo, em geral, com extensão ".inp", e um atalho que permite abrir o *AptPlot*. O acesso ao *SF Pressure Drop* se dá por meio das telas de criação de componentes termo-hidráulicos, uma vez que em geral é lá que o usuário fará uso dos resultados obtidos com os mesmos.

A interface *Ribbon*® do pré-processador também oferece integração direta com o código RELAP5. Para usar essa funcionalidade, basta primeiro informar, na janela de configuração (Figura 4), informar o caminho de instalação do código RELAP5, bem como o caminho para os arquivos de propriedades do fluido refrigerante "H_2O", "D_2O" e "H_2Onew", usados pelo código RELAP5 na simulação numérica de problemas termo-hidráulicos. Uma vez feito isso, o botão de integração com o RELAP5 fica habilitado para uso. Para usá-lo, basta clicar sobre o referido botão e, na caixa de diálogo de seleção de arquivos que se abrirá, selecionar um arquivo de entrada de dados válido para execução dentro do código RELAP5. Assim que esse procedimento for feito, uma janela de comando do sistema operacional irá se abrir e disparar a execução do código RELAP5 usando o arquivo de entrada de dados selecionado, conforme mostra a Figura 5.

Após o processamento, os arquivos de saída gerados ficarão salvos na mesma pasta onde se localiza o arquivo de entrada de dados usado, com o mesmo nome do arquivo de entrada, e com as extensões conforme configuradas na tela de configurações.

Figura 5 - Janela de comando do *Windows*® com código RELAP5 executado diretamente pelo pré-processador



Fonte: autor da dissertação.

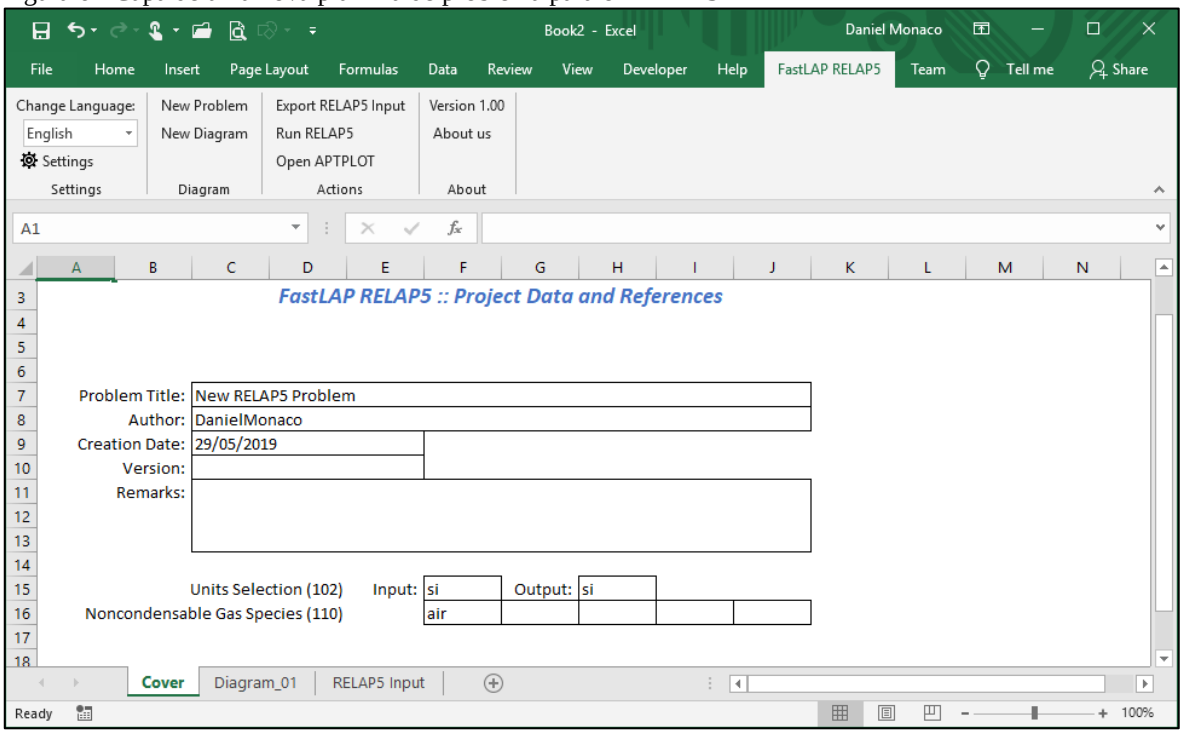
Na tela de configuração também pode ser informado se os arquivos de saída do RELAP5 das execuções anteriores podem ser sobrescritos nas execuções sucessivas.

3.2.4.2 Nova planilha de problema para o RELAP5

Ao clicar no botão "*New Problem*" uma nova planilha do *MS Excel*® é criada com uma formatação especial em suas abas, permitindo ao pré-processador identificar que essa planilha é por ele passível de manipulação. A planilha de problema é inicialmente criada com três abas especiais: uma capa, uma área de desenho de diagramas de nodalização e a área dos dados de entrada para o código RELAP5. As Figuras 6 a 8 mostram cada uma das abas citadas.

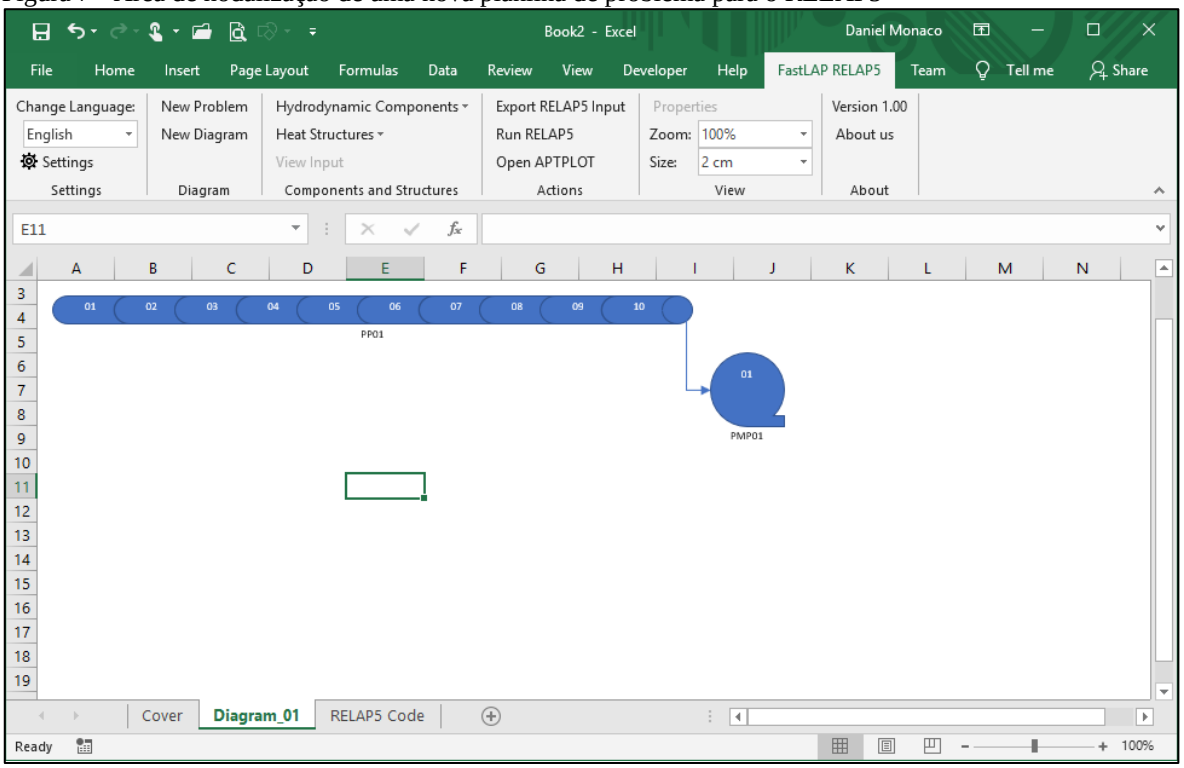
Um arquivo de problema RELAP5 criado com o pré-processador sempre conterá uma aba de capa, onde deve ser informado metadados do problema, como título, versão e data. Algumas dessas informações não são obrigatórias no RELAP5, mas boas práticas mostram que elas são importantes para manutenção e rastreamento da criação dos dados de entrada de um determinado problema.

Figura 6 - Capa de uma nova planilha de problema para o RELAP5



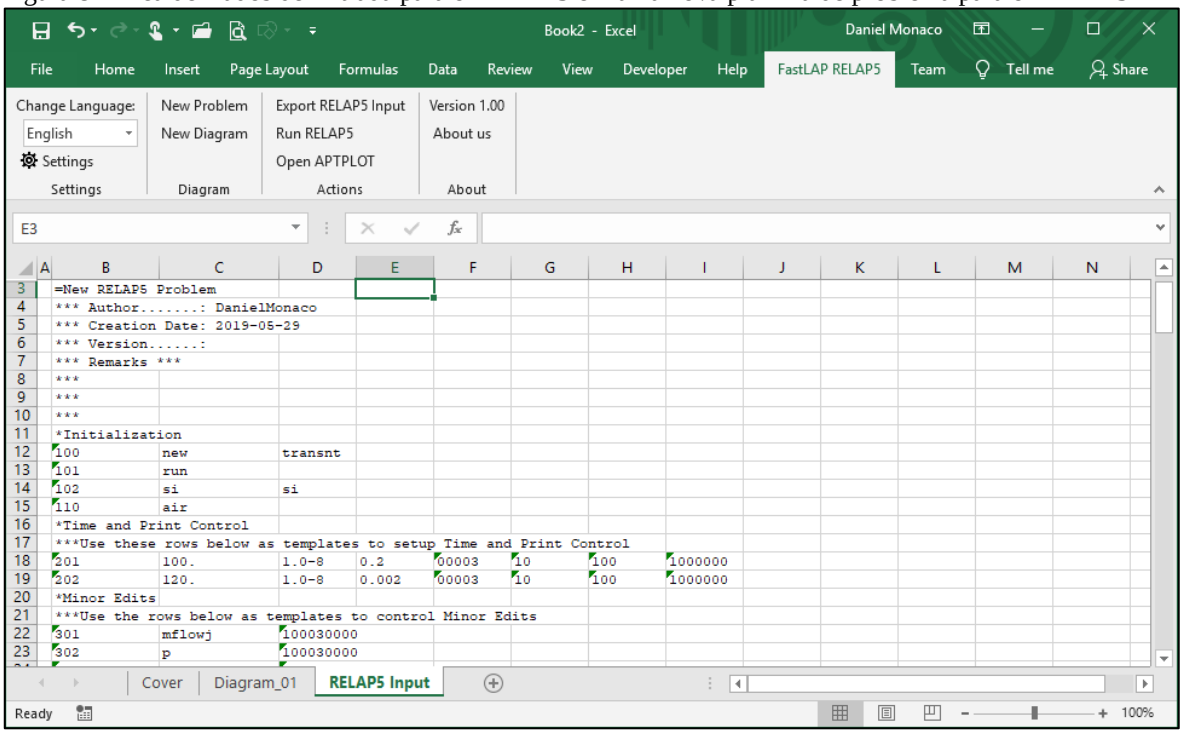
Fonte: autor da dissertação.

Figura 7 - Área de nodalização de uma nova planilha de problema para o RELAP5



Fonte: autor da dissertação.

Figura 8 - Área de Dados de Entrada para o RELAP5 em uma nova planilha de problema para o RELAP5

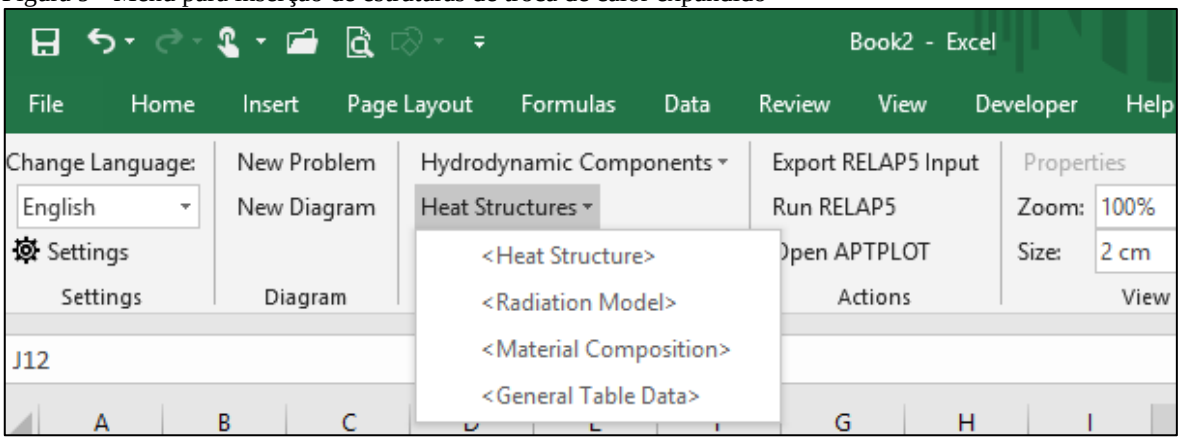


Fonte: autor da dissertação.

Apesar da área de dados de entrada ser totalmente alimentada com a utilização do pré-processador, é permitido que o usuário faça modificações em parâmetros diretamente nessa aba da planilha.

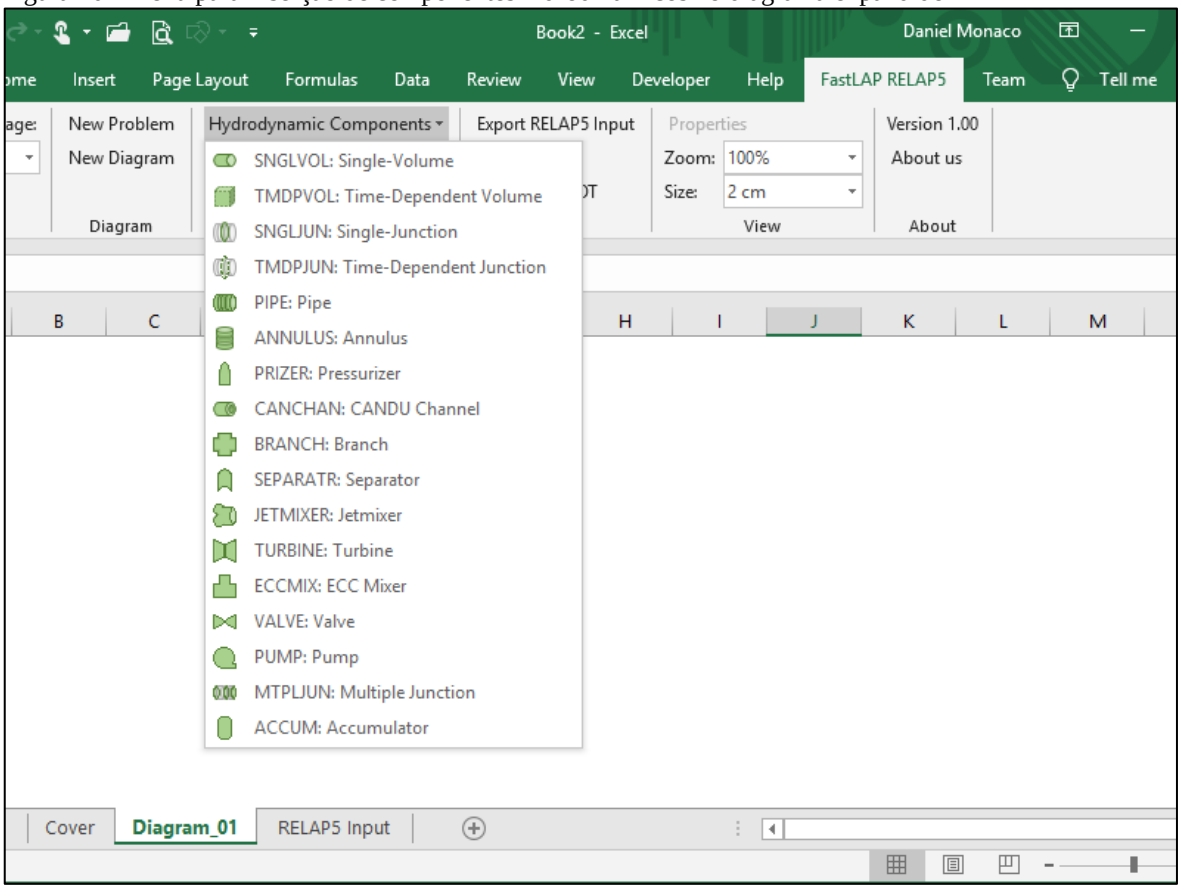
Como visto na Figura 7, ao selecionar a aba de nodalização, os botões de inserção de componentes hidrodinâmicos e de transferência de calor tornam-se visíveis para permitir a inserção dos mesmos no diagrama. As Figuras 9 e 10 mostram, respectivamente, os menus para inserção de estruturas de troca de calor e de componentes hidrodinâmicos expandidos.

Figura 9 - Menu para inserção de estruturas de troca de calor expandido



Fonte: autor da dissertação.

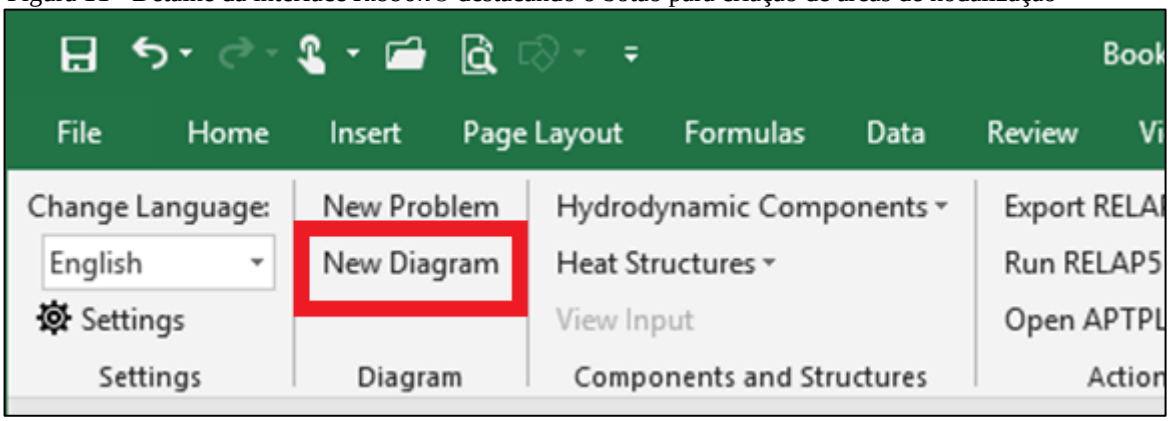
Figura 10 - Menu para inserção de componentes hidrodinâmicos no diagrama expandido



Fonte: autor da dissertação.

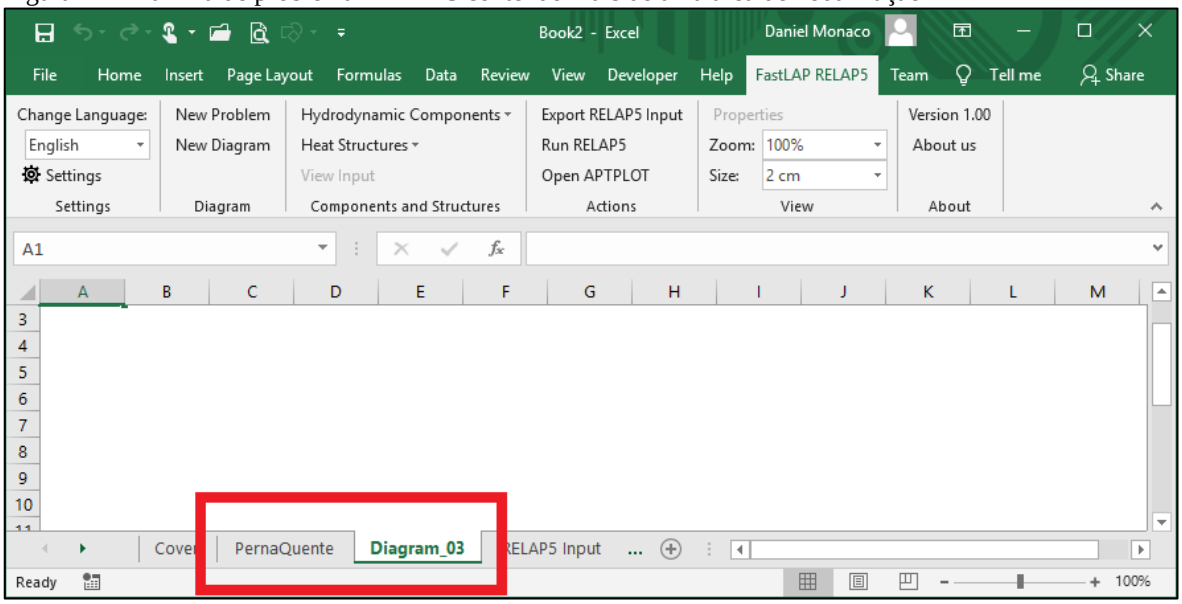
Para melhor diagramação de problemas complexos, o pré-processador permite que uma planilha de problema possa ter várias abas de nodalização. Assim o usuário pode organizar diversos diagramas de nodalização separadamente, mantendo-os em um único arquivo de entrada de dados. Para criar uma nova área de nodalização, basta que o usuário clique no botão "*New Diagram*" (Figura 11), e uma nova aba já preparada para receber diagramas de nodalização será criada na planilha de problema (Figura 12).

Figura 11 - Detalhe da interface *Ribbon*® destacando o botão para criação de áreas de nodalização



Fonte: autor da dissertação.

Figura 12 - Planilha de problema RELAP5 contendo mais de uma área de nodalização



Fonte: autor da dissertação.

O usuário pode criar quantas áreas de nodalização achar necessário para melhor organizar seus diagramas. Por exemplo, pode ser criada uma nodalização para a perna quente, uma para a perna fria, e outra para o vaso do reator: todos os componentes termo-hidráulicos, todavia, ficarão disponíveis num único arquivo de entrada de dados.

Dentro de uma planilha de problema, o usuário pode alterar o rótulo das abas (repare que, na Figura 12, uma das abas de nodalização teve seu nome alterado para “PernaQuente”), alterar a cor das abas, ou até mesmo excluir abas criadas indevidamente. Todavia, nodalizações excluídas com componentes nela desenhadas serão perdidas, porém os componentes permanecerão existindo na aba de entrada de dados.

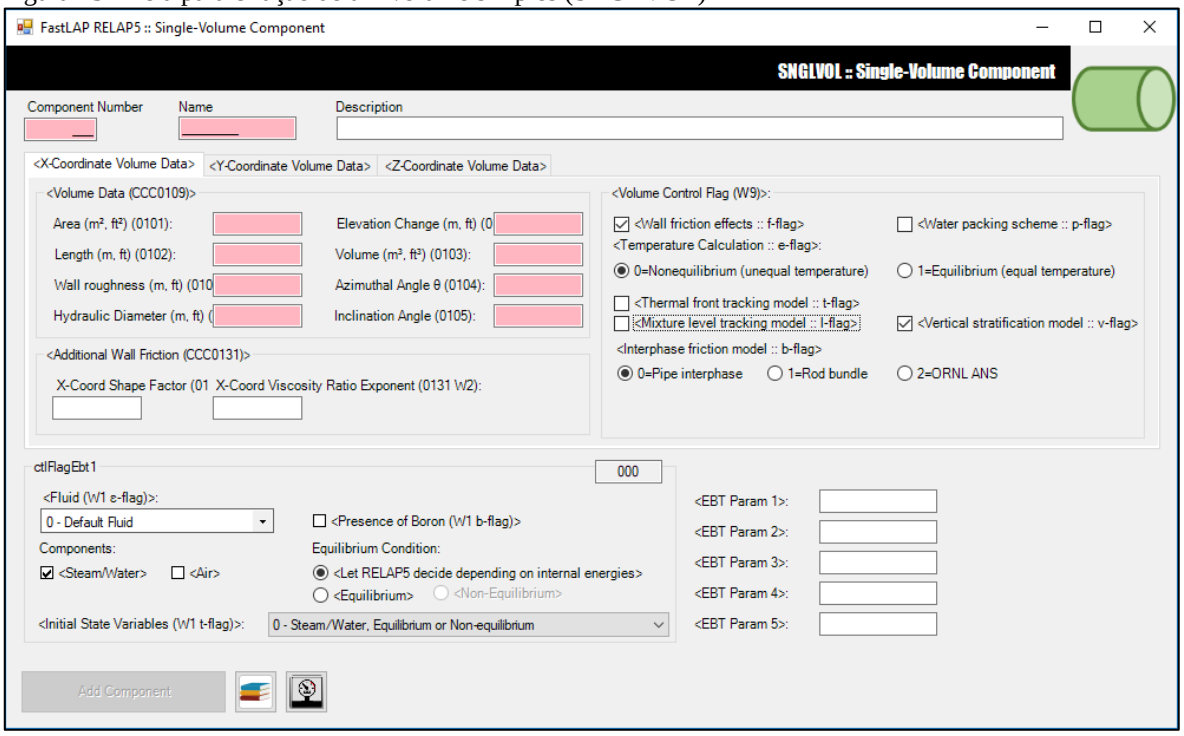
Os nomes da aba de capa e da aba de dados de entrada também podem ser alterados, bem como suas cores. Todavia, não é permitido ao usuário a exclusão dessas abas. Visando evitar a remoção acidental das mesmas, caso o usuário inadvertidamente tente excluir uma dessas abas, o pré-processador irá ativar a proteção de estrutura da planilha (com senha em branco), visando proteger a planilha de problemas de remoções acidentais.

3.2.4.3 Visão geral das telas de criação de componentes

Todas as telas para criação de componentes hidrodinâmicos e de transferência de calor possuem uma estrutura similar, com interface intuitiva, o que ajuda aos novos usuários do pré-processador encontrarem as informações que eles precisam.

A tela para criação de um volume simples (SNGLVOL) pode ser vista na Figura 13.

Figura 13 - Tela para criação de um volume simples (SNGLVOL)



Fonte: autor da dissertação.

Na imagem acima, destacam-se alguns elementos similares a todas as telas de componentes hidrodinâmicos, bem como de estruturas de troca de calor. A qualquer componente criado é necessário informar um número, um nome e uma descrição amigável. Durante o seu uso, essas informações são solicitadas já no início da tela, induzindo o usuário a preencher essas informações antes de prosseguir com os demais dados.

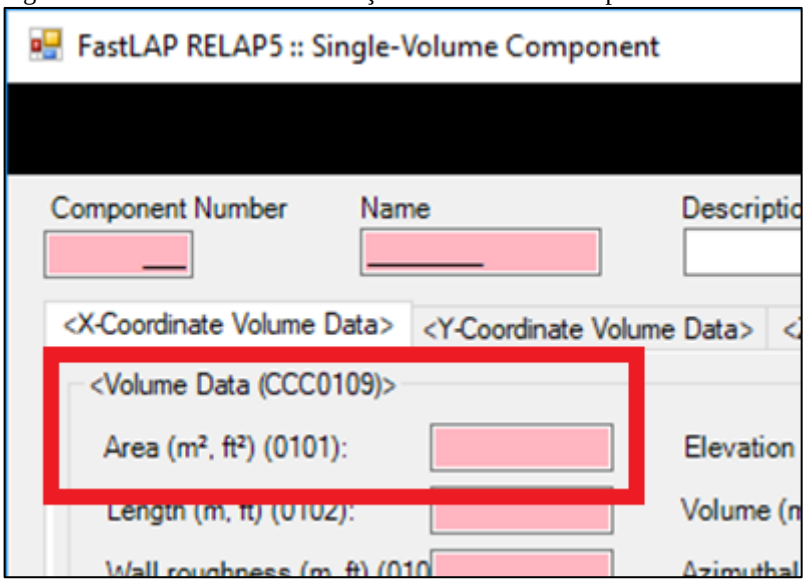
No RELAP5, todos os componentes usados nos cartões de entrada precisam ter um número a eles atribuído. É a partir desse número que o RELAP5 identifica cada componente durante a execução de seu código. Essa é uma propriedade obrigatória, uma vez que todos os demais cartões de entrada desse determinado componente usarão esse mesmo número em sua composição. O nome do componente também é outra informação obrigatória: um conjunto de até oito caracteres deve compor o nome do componente criado. Essas informações serão inseridas nos respectivos cartões de entrada do RELAP5 para o componente criado.

A descrição do componente, contudo, não é uma informação obrigatória ao RELAP5, na verdade o RELAP5 sequer solicita essa informação; contudo, boas práticas de uso do RELAP5 compartilhadas pelos usuários durante o trabalho informam que é importante acrescentar algum tipo comentário ou lembrete sobre a estrutura criada, para facilitar o entendimento e manutenção dos dados de entrada no futuro. Nesse caso, portanto,

essa informação será inserida em um cartão de comentário logo acima do cartão de criação do componente, como se fosse mais um metadado do problema.

Analisando o restante da tela exibida na Figura 13, pode ser visto que imediatamente ao lado de cada caixa de entrada de dado há um rótulo com uma informação amigável sobre o que deve ser inserido naquela caixa de entrada, bem como uma referência sobre que cartão de entrada do RELAP5 solicita essa informação. Por exemplo: a área de vazão da coordenada X do volume em questão (primeiro campo da tela), é solicitada pela *word* 1 do cartão CCC0109, como o rótulo ao lado do primeiro campo deixa claro (Figura 14). Dessa forma, caso o usuário tenha dúvidas sobre que informação deve ser entrada naquele campo, ele pode facilmente consultar o manual do RELAP5 e verificar mais detalhes técnicos sobre o mesmo.

Figura 14 - Detalhe da tela de criação de um volume simples mostrando o campo "área"



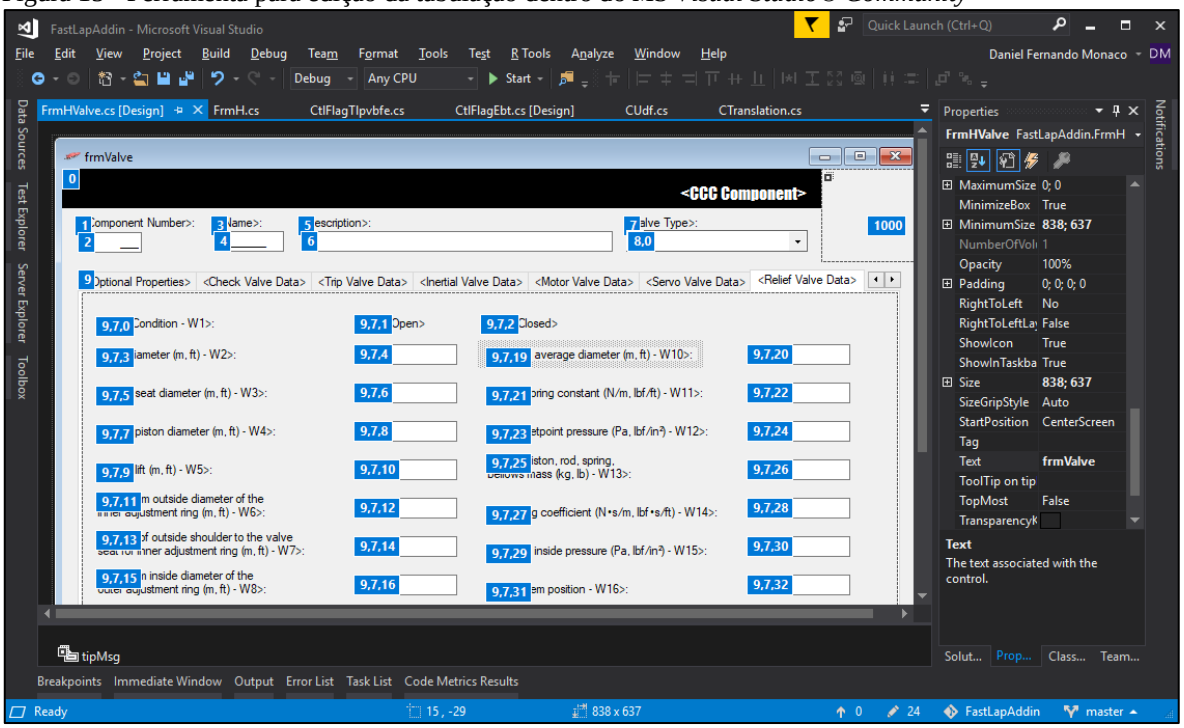
Fonte: autor da dissertação.

Todo o pré-processador possui uma abordagem fluida para a construção de sua interface. Dessa forma, o usuário dificilmente verá mensagens *popup* aparecendo na tela: toda a validação dos campos e ações permitidas se dão com alteração de cores, alteração da visibilidade de controles ou desativação dos mesmos.

O desenho das telas foi pensado de forma a tornar a navegação pelos parâmetros de entrada de forma evolutiva. Em algumas telas, como na da junção simples, as informações obrigatórias ficam na primeira tela, e as informações opcionais podem ser inseridas a partir de uma aba secundária dentro da tela; em outros casos, como na tela de troca de calor, as informações foram agrupadas em abas de acordo com sua natureza, e dentro de cada uma das abas as informações obrigatórias aparecem em uma subaba principal da tela e as

opcionais aparecem em subabas secundárias dentro da mesma tela. A ordem de navegação dos campos (também chamada de “tabulação”) também foi levada em consideração, para que os campos na tela possam ser navegados por meio do teclado (com uso da tecla TAB) de forma natural, como ocorre com a leitura. E, nesse aspecto, a utilização do *MS Visual Studio*® como IDE de programação foi interessante, pois seu editor de telas possui uma ferramenta bastante amigável para facilmente ajustar a tabulação, como mostrado na Figura 15.

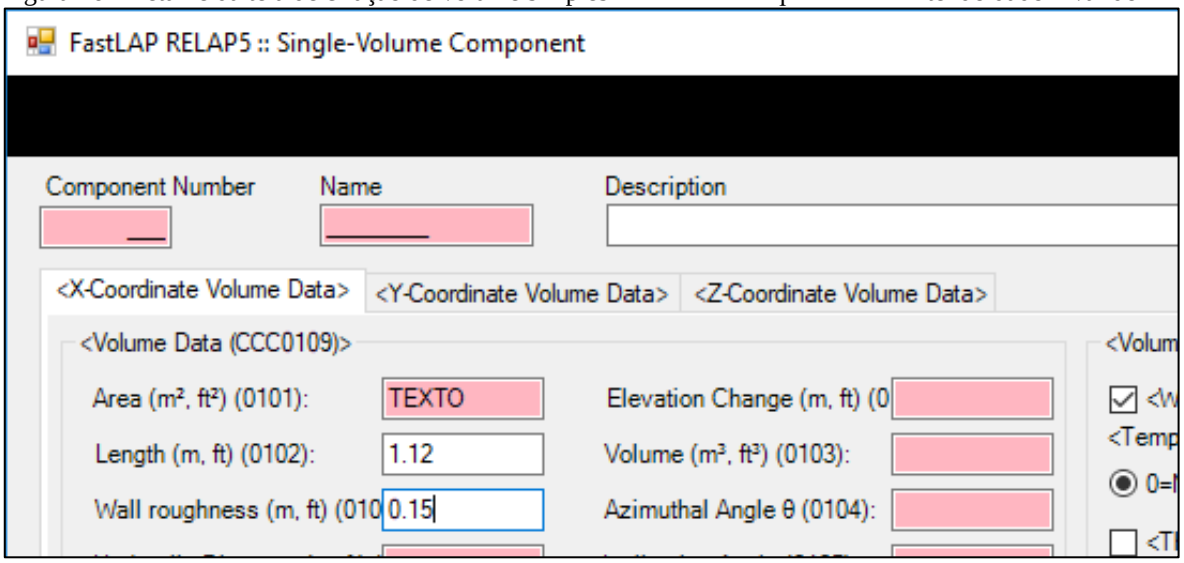
Figura 15 - Ferramenta para edição da tabulação dentro do *MS Visual Studio® Community*



Fonte: autor da dissertação.

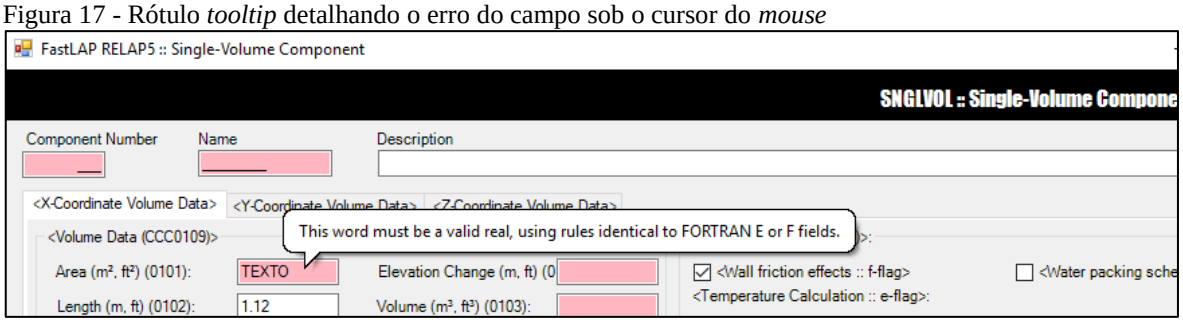
Para a validação do conteúdo dos campos, a abordagem adotada foi a seguinte: caso o usuário informe um valor inválido para determinado campo, esse campo tem sua cor de fundo alterada do branco para o vermelho, como pode ser visto na Figura 16, e o botão de adição do componente é desabilitado para impedir o usuário de inserir esse componente com dados inválidos. Nenhuma mensagem é exibida ao usuário: caso o usuário deseje entender o motivo do campo ter ficado vermelho, basta passar o cursor do *mouse* sobre o campo para que um rótulo (*tooltip*) apareça sobre o campo informando o erro, como na Figura 17. Após todos os erros da tela serem corrigidos, o botão de adição é novamente habilitado, permitindo assim a inserção do componente no diagrama.

Figura 16 - Detalhe da tela de criação de volume simples exibindo o campo “área” contendo dado inválido



Fonte: autor da dissertação.

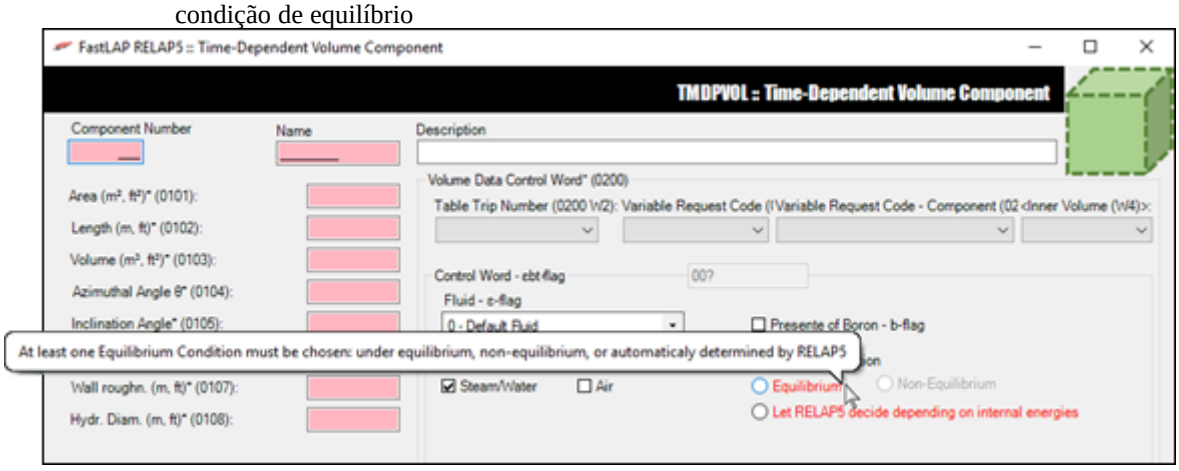
Figura 17 - Rótulo *tooltip* detalhando o erro do campo sob o cursor do *mouse*



Fonte: autor da dissertação.

Controles que não possuem cor de fundo branca em seus rótulos, como botões de rádio ou caixas de seleção, não ficam com o fundo vermelho: caso sejam obrigatórios e não sejam selecionados, eles ficarão com suas letras na cor vermelha, como pode ser visto na Figura 18.

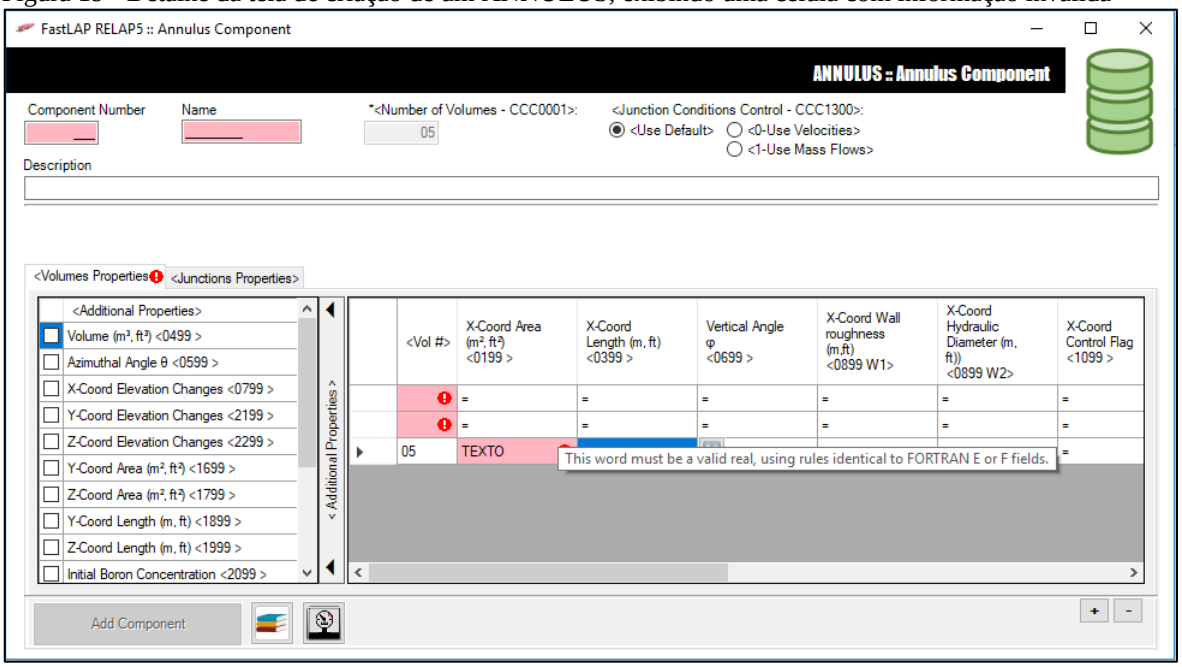
Figura 18 - Detalhe da tela de criação de TMDPVOL, informando obrigatoriedade de seleção de uma condição de equilíbrio



Fonte: autor da dissertação.

Até mesmo os controles do tipo *grid* possuem validação em cada uma de suas células: no caso particular das *grids*, as células inválidas aparecerão com o fundo em vermelho, além de um pequeno sinal de exclamação ao seu lado; ao passar o mouse sobre a exclamação o usuário é informado sobre qual erro o pré-processador identificou naquele dado informado. Tal comportamento pode ser visto na Figura 19.

Figura 19 - Detalhe da tela de criação de um ANNULUS, exibindo uma célula com informação inválida

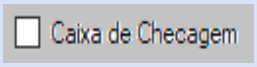
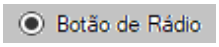
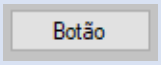
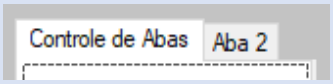
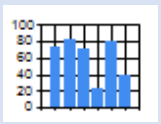
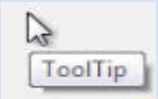
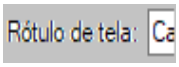


Fonte: autor da dissertação.

Campos obrigatórios já aparecerão em vermelho tão logo a tela é aberta pela primeira vez, mesmo que não tenham sido preenchidos. Campos não obrigatórios não ficarão em vermelho caso não sejam preenchidos; porém, caso sejam preenchidos com informações inválidas, eles também terão sua cor trocada para o vermelho e passarão a contar com o rótulo informativo, até que sejam corrigidos ou tenham seu conteúdo removido.

Apesar do controle de entrada de dados mais comum ser a famosa caixa de texto, a plataforma .NET® oferece diversos outros tipos de controle que podem ser usados para a confecção de uma interface de entrada de dados mais simpática, permitindo ao usuário uma melhor experiência durante o uso da ferramenta. Ao definir a abordagem de entrada de dados para este pré-processador, a adoção de diferentes controles para entrada de dados torna a inserção de informação mais fácil e menos sujeita a erros. A Tabela 2 enumera alguns exemplos de controles que foram usados neste pré-processador com esse objetivo.

Tabela 2 - Exemplos de controles de entrada de dados

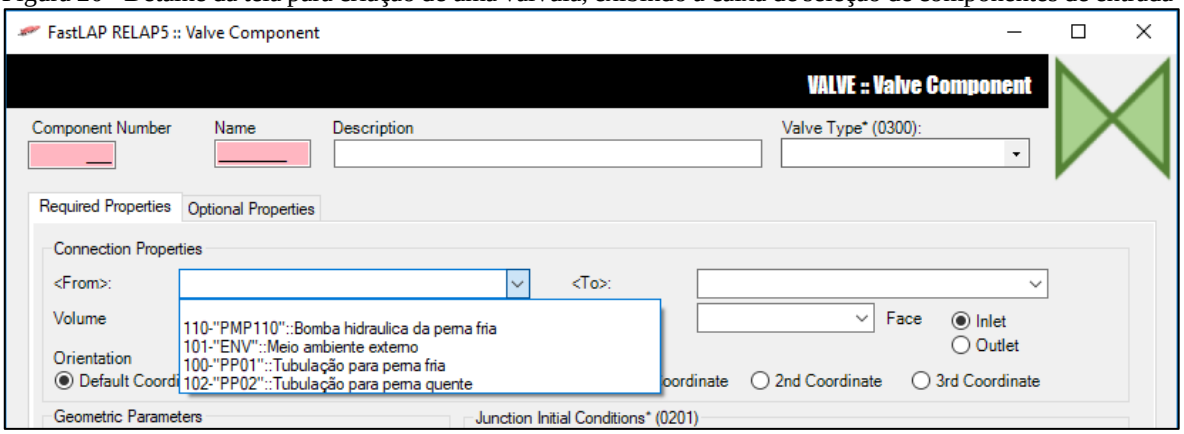
Nome do controle	Representação visual do controle
Caixa de texto (*textbox*)	
Caixa de seleção (*combobox*)	
Caixa de checagem (*checkbox*)	
Botão de rádio (*radio button*)	
Botão (*button*)	
Caixa incremental (*numeric up/down*)	
Controle de abas (*tab control*)	
Barra de tração (*trackbar*)	
Gráfico (*chart*)	
Barra de rolagem (*scroll bar*) horizontal ou vertical	
Rótulo de cursor (*tooltip*)	
Rótulo de tela (*label*)	

Fonte: autor da dissertação.

Quando o valor de uma determinada *word* precisa ser escolhida dentro de um conjunto finito de opções discretas já determinadas pelo manual do RELAP5, ou já são conhecidas pelos usuários (por terem sido previamente configuradas, por exemplo), uma caixa de entrada do tipo *combobox* acaba oferecendo uma melhor experiência ao usuário, uma vez que permite a ele selecionar o valor conhecido a partir de uma lista que se abre ao clique da seta à direita do controle. Em determinados cartões de entrada, o usuário é obrigado a informar o código de algum outro elemento previamente criado dentro dos dados de entrada do RELAP5. Tomando-se como exemplo a configuração de uma válvula: o RELAP5 exige que, ao configurar a válvula, o usuário informe o número dos volumes aos quais a entrada e a saída da válvula estarão conectadas, como exibido na Figura 20. Além disso, há cartões que solicitam que o número de variável lógica (*trip*) seja informado para abrir ou fechar a válvula, por exemplo. Para esses casos, o pré-processador usa controles de seleção, que

abrem uma lista para o usuário selecionar determinado valor a partir de valores previamente configurados pelo mesmo, evitando assim, possíveis erros no cálculo devido a informações inseridas de forma errada.

Figura 20 - Detalhe da tela para criação de uma válvula, exibindo a caixa de seleção de componentes de entrada



Fonte: autor da dissertação.

Todavia, quando essa seleção se restringe a apenas dois ou três valores distintos, ela é melhor representada ou evidenciada por meio de *checkboxes* ou *radioboxes*. As *checkboxes* são muito comuns para entrada de informação do tipo binária (sim/não, tem/não tem, ligado/desligado), enquanto que as *radioboxes* são comuns quando apenas um valor pode ser selecionado dentre algumas poucas opções.

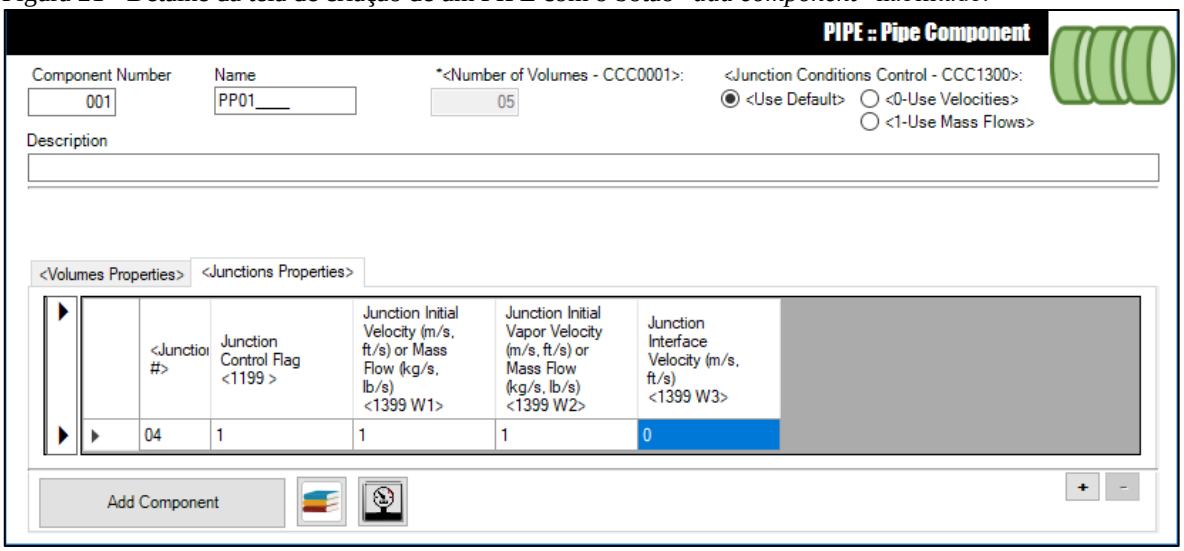
Essa simples alteração do tipo de controle usado para a entrada de dados já permite com que a validação destes dados inseridos pelo usuário torne-se mais amigável e agradável, ou até mesmo torne-se completamente desnecessária.

Para componentes que exigem a entrada de muitas informações distintas, o que tornaria a tela muito poluída visualmente, controles de aba mostram-se uma boa alternativa às barras de rolagem, que não são muito apreciadas pelos usuários. Os controles de aba oferecem uma alternativa simples e organizada para deixar a entrada de dados mais amigável, mesmo para componentes complexos.

Uma vez que todos os erros da tela são corrigidos, o botão "*add component*", localizado na parte inferior da tela, é habilitado para uso, conforme mostrado na Figura 21. Uma vez nele clicado, os cartões de dados de entrada são criados pelo pré-processador na respectiva aba, e uma representação visual do referido componente é desenhado com os *AutoShapes* na aba de diagrama selecionada, conforme mostra a Figura 22. Cada componente termo-hidráulico possui uma representação visual diferente, para facilitar o entendimento do diagrama. Elementos que possuem conexão com outros elementos, tais

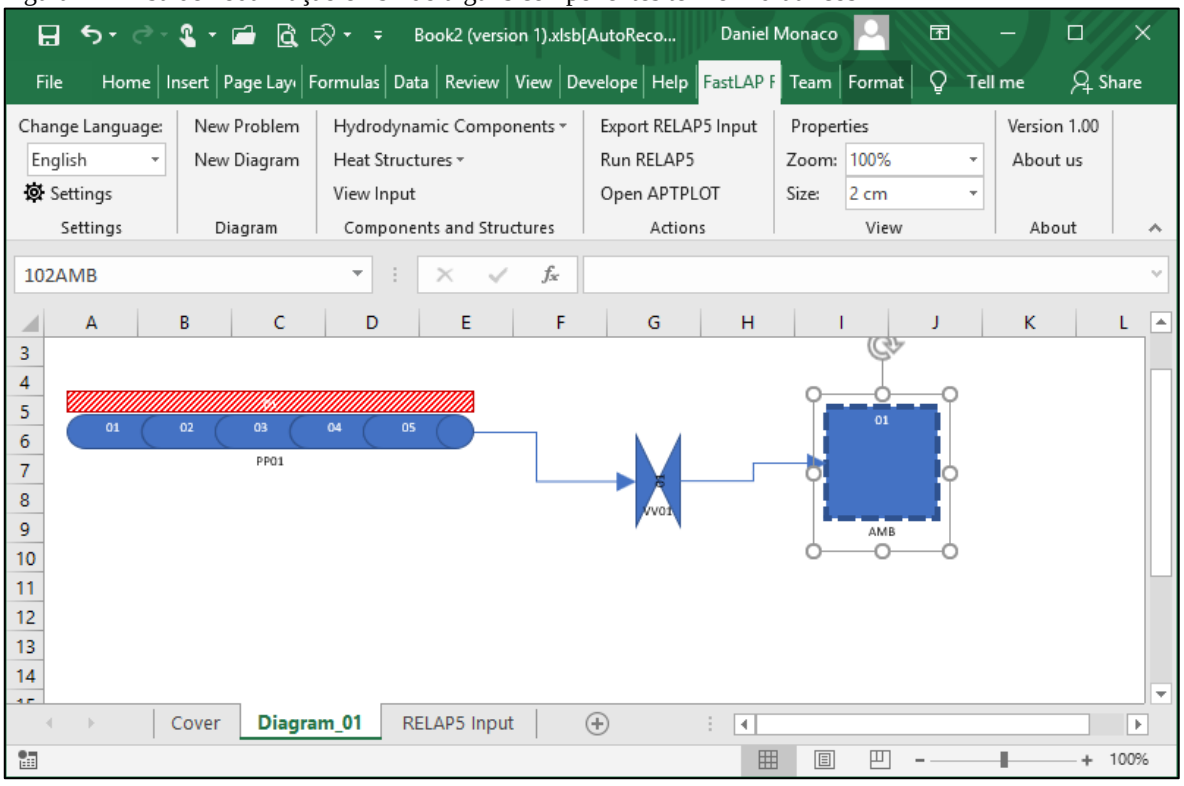
como junções, bombas, válvulas, entre outros, recebem junto a si, além de sua representação visual, setas que representam as conexões de entradas e saídas, conforme o caso.

Figura 21 - Detalhe da tela de criação de um PIPE com o botão "*add component*" habilitado.



Fonte: autor da dissertação.

Figura 22 - Aba de nodalização exibindo alguns componentes termo-hidráulicos



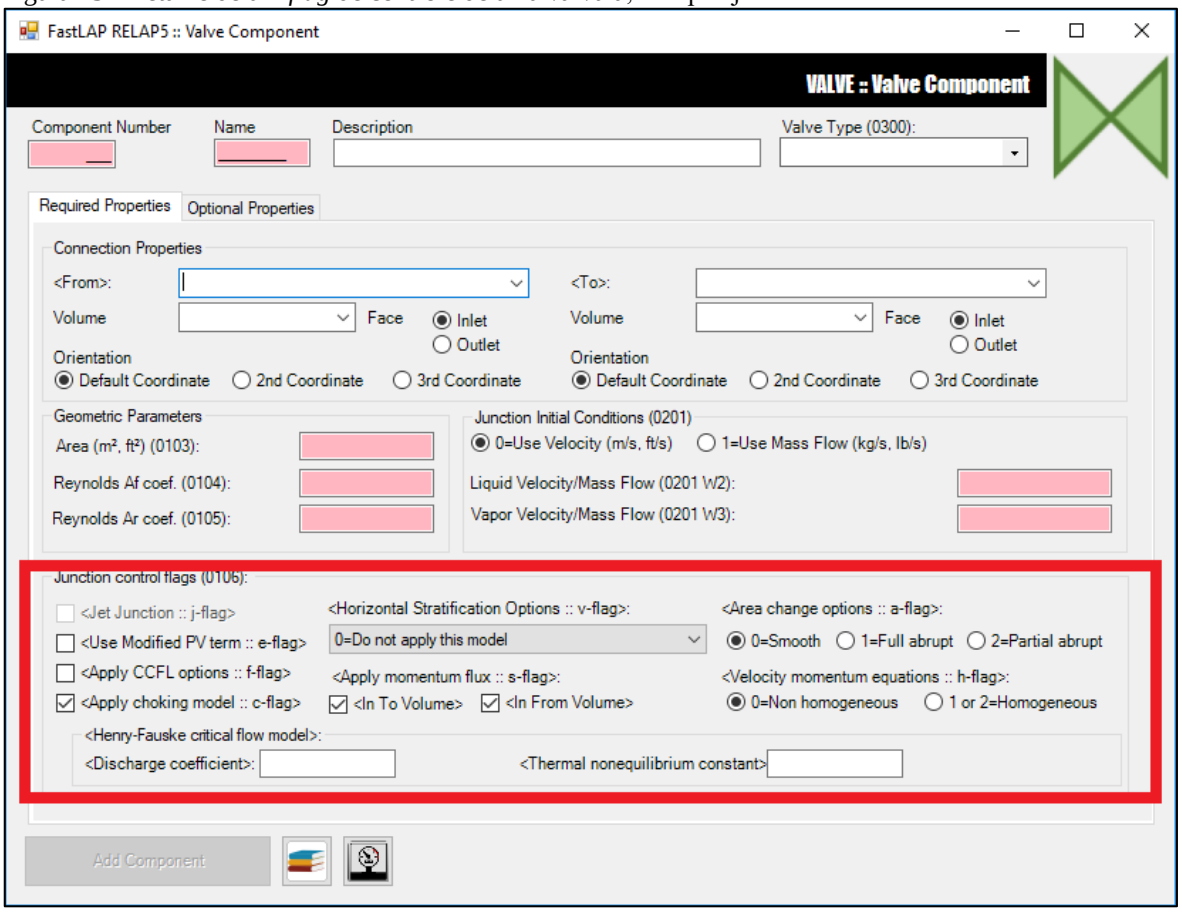
Fonte: autor da dissertação.

3.2.4.4 Visão geral dos *flags* de controle

Outro tipo de parâmetro muito comum nos componentes do RELAP5 são os *flags* de controle. Os *flags* nada mais são que um conjunto de parâmetros com diferentes

significados, mas que são informados ao código RELAP5 por meio de uma única *word*. Um exemplo de *flag* é o *flag* de controle da junção de uma válvula. Em uma única *word* de entrada do RELAP5, o usuário deve informar se a válvula deverá empregar algum *choking model*, ou seja, se o modelo se aplicará aos volumes de entrada, de saída ou de ambos, se a alteração de abertura da válvula será abrupta ou suave, entre outros; tudo isso num único parâmetro de entrada. Para esses casos, o pré-processador conta com uma expansão desse parâmetro em um conjunto de controles de entrada, onde cada uma das opções é inserida separadamente, como pode ser visto na Figura 23. Essa abordagem evita que o usuário possa esquecer uma informação necessária para seu problema em estudo, e facilmente identifique as combinações de configuração possíveis.

Figura 23 - Detalhe de um *flag* de controle de uma válvula, do tipo "jefvcahs"



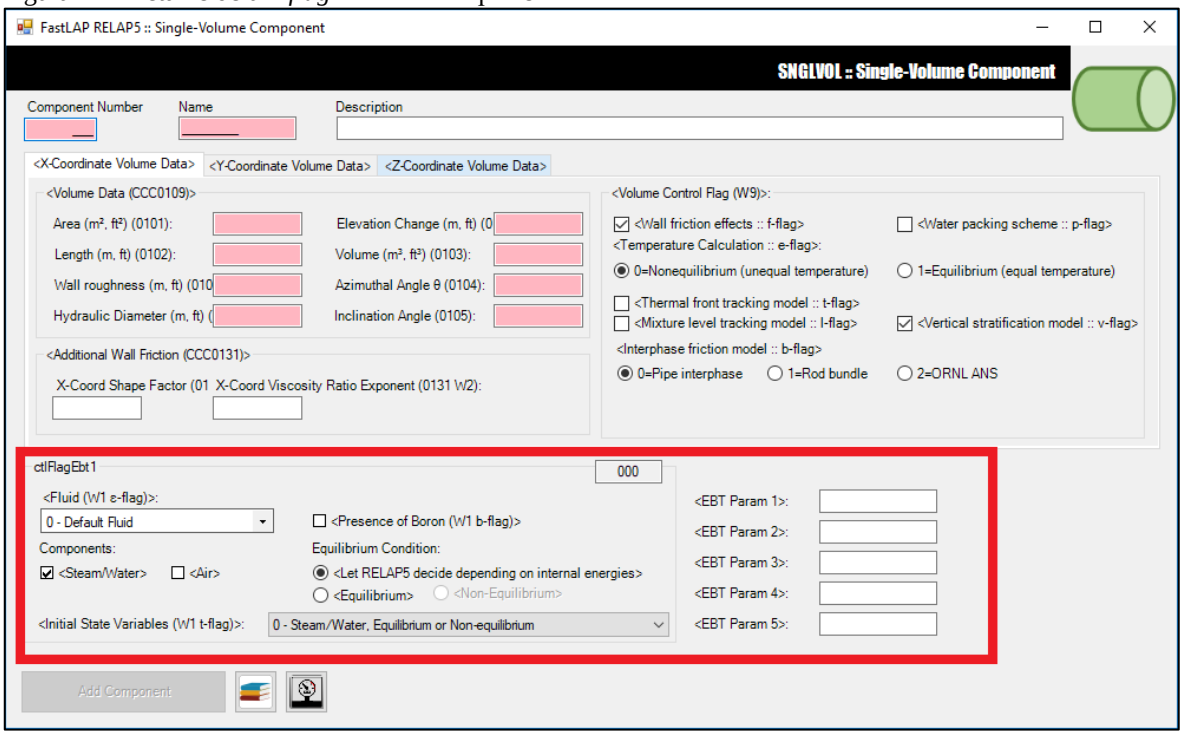
Fonte: autor da dissertação.

Os *flags* de controle criados foram três:

- *flags* do tipo "jefvcahs", em geral para condição inicial de junções (Figura 23);
- *flags* do tipo "εbt", para condição inicial dentro dos volumes (Figura 24);

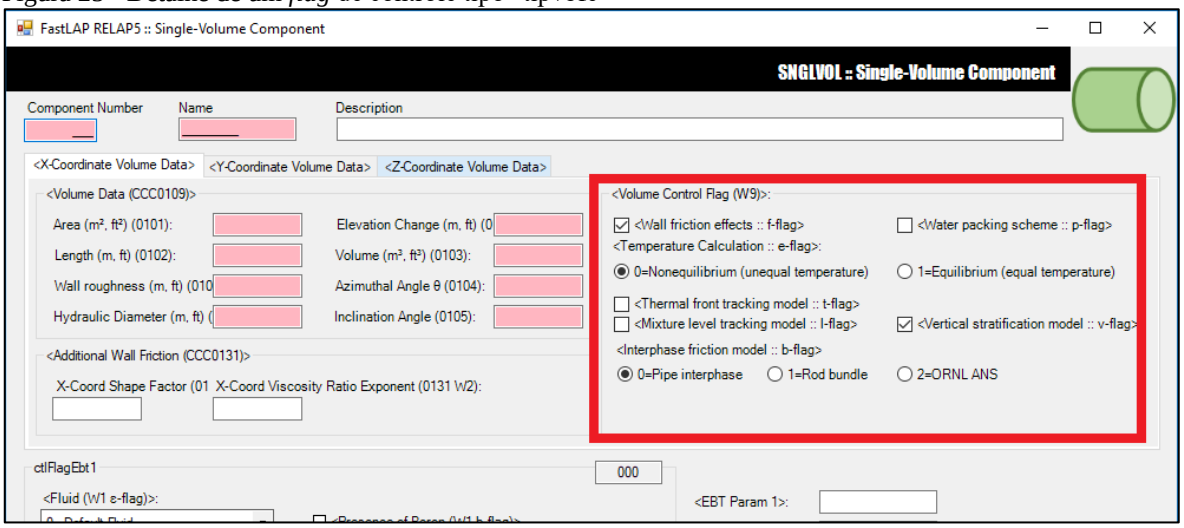
- *flags* do tipo “tlpvbfe”, para condição inicial de volumes, orientados aos eixos (Figura 25).

Figura 24 - Detalhe de um *flag* de controle tipo “εbt”



Fonte: autor da dissertação.

Figura 25 - Detalhe de um *flag* de controle tipo “tlpvbfe”



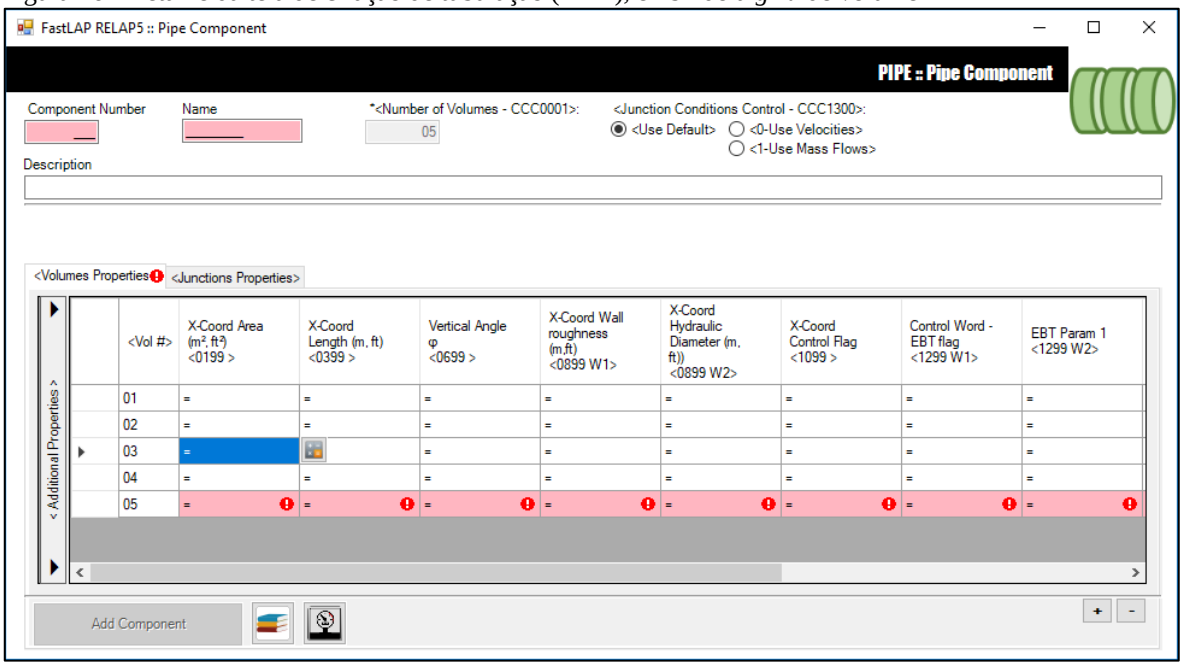
Fonte: autor da dissertação.

3.2.4.5 Entrada de dados por tabelas (*grids*)

Por fim, algumas informações de entrada são inseridas no código RELAP5 no formato de tabelas. O termo “cartões do tipo tabela” se refere a cartões que podem ser informados mais de uma vez para determinado componente. Em geral, esses cartões são

referenciados no manual do RELAP5 dentro de um intervalo numérico. Por exemplo, para a configuração de uma tubulação (PIPE), o manual do RELAP5 nos informa que as áreas de escoamento de cada um dos volumes de controle da tubulação devem ser informadas por meio dos cartões de CCC0201 até CCC0299. Isso dá um total de até 99 cartões com uma numeração semelhante que aparecerá no arquivo de dados de entrada. Para facilitar a inserção desse tipo de informação, o pré-processador conta com elementos de tela no formato de planilhas muito similares às planilhas do *MS Excel®*, conhecidos como *grids*. Abaixo, a Figura 26 mostra um detalhe da tela de criação de uma tubulação, onde um controle tipo *grid* é usado para informar um conjunto de dados da tubulação que precisam ser informados em cartões de tabela.

Figura 26 - Detalhe da tela de criação de tubulação (PIPE), exibindo a *grid* de volume

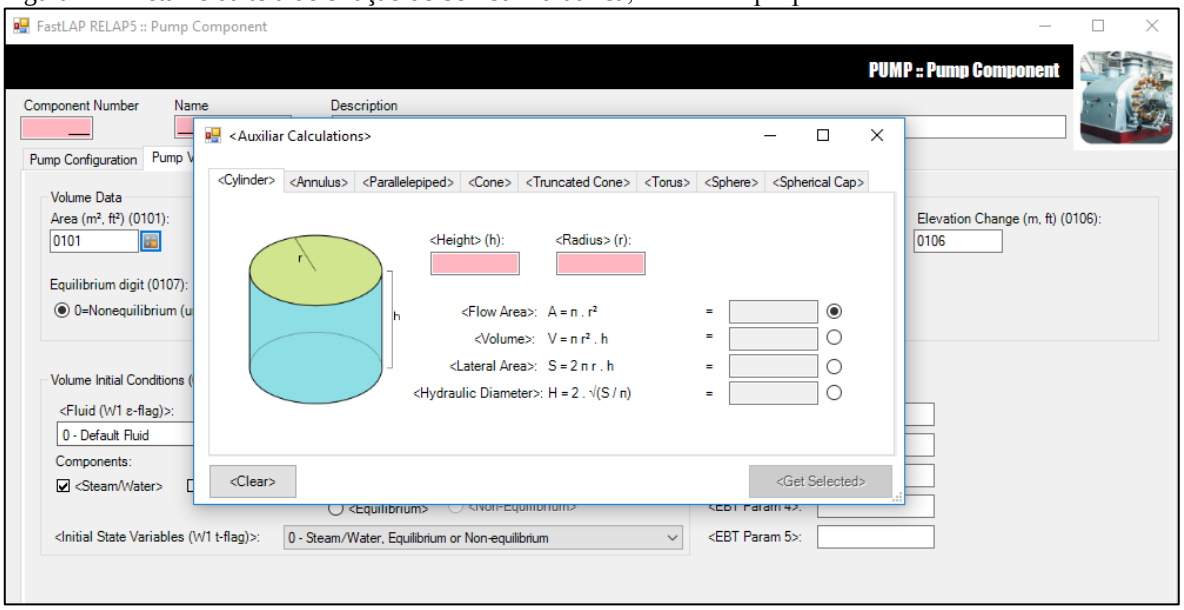


Fonte: autor da dissertação.

3.2.4.6 Botões auxiliares de tela

Em todas as telas de entrada, o usuário notará em alguns campos o aparecimento de um pequeno botão lateral que lembra uma pequena calculadora, conforme exibido na Figura 27. Esse botão aparece para permitir ao usuário o acesso a uma tela onde é possível calcular algumas dimensões físicas de elementos hidrodinâmicos para diferentes geometrias, e já transportar o resultado para a caixa de texto previamente selecionada pelo usuário. Em campos onde não há a entrada desse tipo de grandeza, todavia, a calculadora não aparece.

Figura 27 - Detalhe da tela de criação de bomba hidráulica, com destaque para a “calculadora”



Fonte: autor da dissertação.

A “calculadora” permite o cálculo das seguintes grandezas:

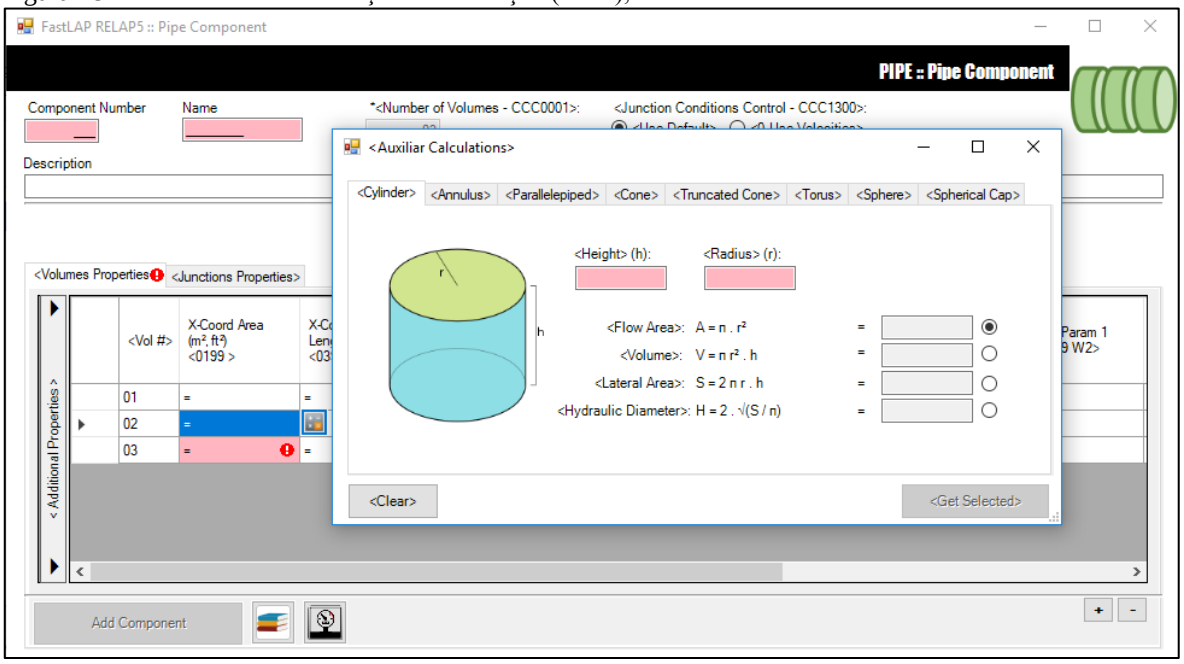
- Área de escoamento;
- Volume;
- Área de transferência de calor;
- Diâmetro hidráulico.

E todos esses cálculos podem ser feitos para as seguintes geometrias:

- Cilindro;
- Anel;
- Paralelepípedo;
- Cone;
- Cone truncado;
- Tora de raio interno e externo;
- Esfera;
- Calota esférica.

Nas telas onde há a entrada de dados por tabela, a “calculadora” também aparece ao selecionar qualquer célula. Nesse caso, dependendo do tipo de informação da célula selecionada, o botão pode abrir a tela de cálculo de grandezas, ou alguma das telas de entrada de *flags* (Figura 28), ajudando o usuário a informar adequadamente esse tipo de *flag*.

Figura 28 - Detalhe da tela de criação de tubulação (PIPE), exibindo a “calculadora”



Fonte: autor da dissertação.

Além do botão de “calculadora”, dois outros botões estão disponíveis em todas as telas de criação de componentes. São eles:

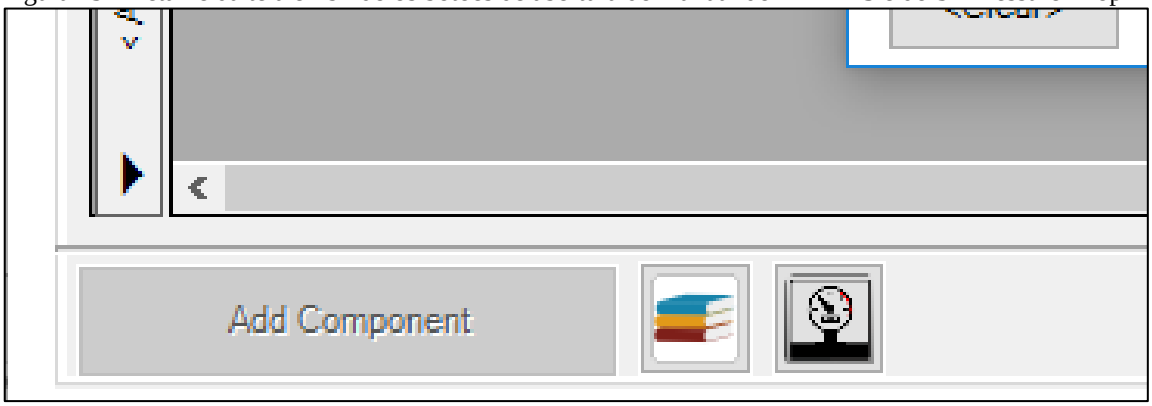
- botão para abrir manual do RELAP5;
- botão para abrir o *SF Pressure Drop*.

Os botões ficam localizados imediatamente ao lado do botão de adicionar componentes, formando uma espécie de barra de atalhos disponível em qualquer tela de criação de componentes do pré-processador, como pode ser visto na Figura 29.

O botão de abertura do manual do RELAP5 abre um arquivo em PDF contendo o manual, conforme configuração prévia na tela de ajustes, e procura o componente que está sendo gerado na página correspondente à sua descrição, facilitando assim a busca de informações dentro do manual.

Já o botão para abertura do *SF Pressure Drop* funciona como um atalho, facilitando o acesso ao referido programa, mediante configuração prévia na tela de configurações.

Figura 29 - Detalhe da tela exibindo os botões de abertura do manual do RELAP5 e do SF Pressure Drop



Fonte: autor da dissertação.

3.2.4.7 Entrada de dados por meio de fórmulas

Um dos atributos mais importantes de um programa de planilhas eletrônicas como o *MS Excel*® é a sua capacidade de lidar com fórmulas. Por meio das fórmulas, o usuário do *MS Excel*® ganha uma grande flexibilidade na atualização de informações dentro dos documentos, uma vez que os dados inseridos na planilha são atualizados, as fórmulas são automaticamente recalculadas.

Outro uso menos comum, mas nem por isso menos importante, das fórmulas é que elas permitem registrar uma memória de cálculo, mesmo quando seu resultado independe de outros dados de entrada. Caso o usuário volte a dar manutenção em uma determinada planilha, e não se lembre mais do motivo de ter inserido determinada constante dentro da mesma, se a constante foi inserida como uma fórmula, há chances de que ao ler a expressão que calcula a constante o usuário se lembre do motivo pelo qual inseriu determinado valor.

O usuário do RELAP5 se depara constantemente com situação similar. A título de exemplo, uma série de componentes hidrodinâmicos no RELAP5 solicita que seja informada a área de escoamento como propriedade. Como boa parte da tubulação usada em plantas termo-hidráulicas possui geometria cilíndrica, a situação mais comum é de que sempre a área de escoamento seja dada como uma função do diâmetro da tubulação em questão, que é calculada pela equação (1):

$$\mathrm{Area} := \pi \cdot \left(\frac{\mathrm{diametro}}{2} \right)^2 \qquad (1)$$

Caso a tubulação em questão tenha, por exemplo, 30cm de diâmetro (ou 0,3m), a área de escoamento pode ser obtida pela equação (2), abaixo:

$$Area = \pi \cdot \left(\frac{0,3}{2} \right)^2 \qquad Area = \pi \cdot \left(\frac{diametro}{2} \right)^2 \tag{2}$$

A expressão acima resultará num valor aproximado de 0,070685m^2, e esse valor será informado nos cartões de entrada do código RELAP5. Se o usuário não deixar algum lembrete usando cartões de entrada de comentários, no futuro ele poderá não se lembrar de qual foi o diâmetro utilizado para aquela área de escoamento. Claro que, no caso ilustrado aqui, o usuário sempre pode fazer a conta inversa e descobrir o diâmetro original da tubulação.

Entretanto, a área de escoamento pode vir a depender de mais de um parâmetro além do diâmetro da tubulação. Usando ainda o exemplo anterior, caso o usuário do código RELAP5 queira representar uma determinada junção de uma tubulação que sofre uma obstrução de 10% de sua área, a nova área de escoamento dessa junção teria que ser calculada por meio de uma expressão que depende agora de dois parâmetros: diâmetro da tubulação e percentual de obstrução, o que resultaria em algo como o mostrado na equação (3), abaixo:

$$Area = \pi \cdot \left(\frac{diametro}{2} \right)^2 \cdot (1 - obstrucao\%) = \pi \cdot \left(\frac{0,3}{2} \right)^2 \cdot 0,9 \tag{3}$$

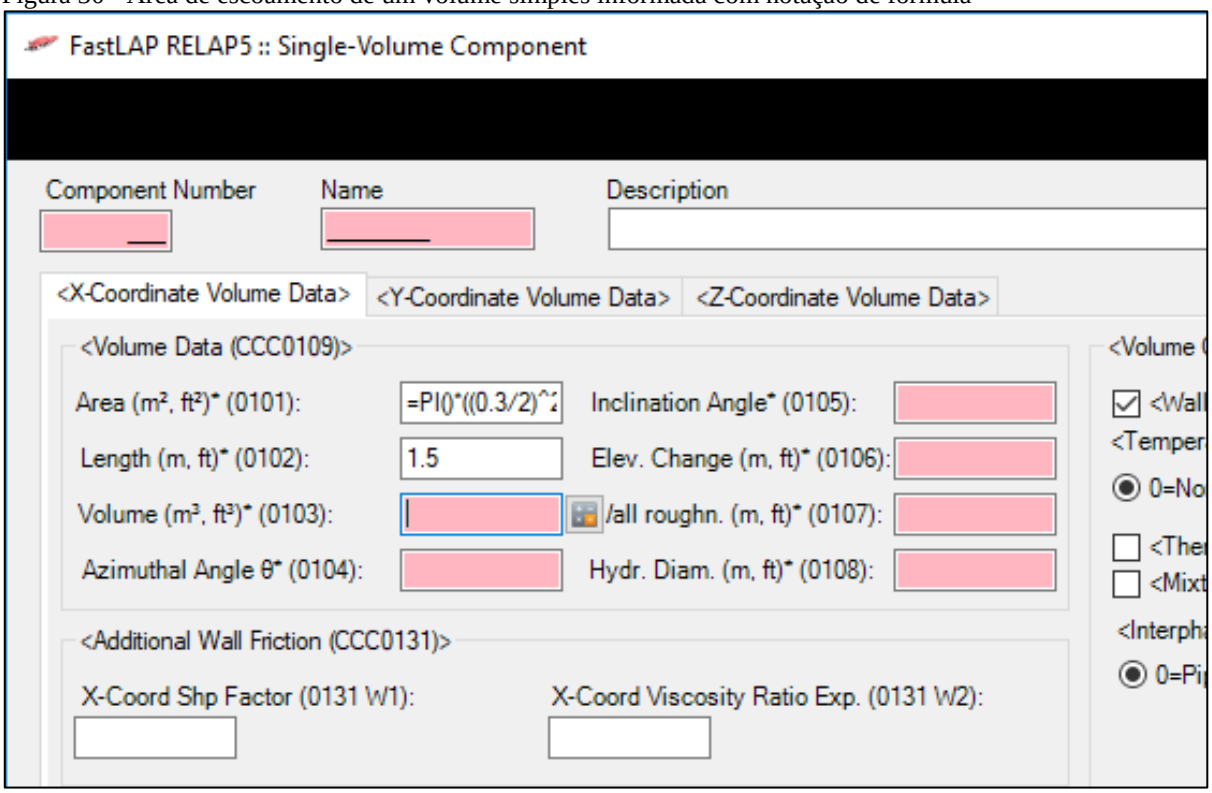
A expressão acima resultaria algo próximo a 0,063617m^2, e esse valor é que será informado pelo usuário para o código RELAP5, por meio dos cartões de entrada. Partindo desse cenário, caso no futuro o usuário RELAP5 queira lembrar o diâmetro original da tubulação em questão e obter o valor usando somente a expressão da área, sem levar em consideração a obstrução outrora determinada, ele descobrirá que sua tubulação possui diâmetro aproximado de 28,46cm.

Para reduzir as ocorrências deste tipo problema, o pré-processador permite que as constantes de cada um dos parâmetros inseridos para os componentes termo-hidráulicos sejam informadas tanto como valores como fórmulas. Caso o usuário deseje informar um valor no formato de fórmula, basta que ele inicie a digitação do campo usando o sinal

de “=”, da mesma forma que ele faz quando digita uma fórmula em uma célula de uma planilha eletrônica do *MS Excel*®.

A notação usada para fórmulas obedece ao mesmo padrão de sintaxe para entrada de fórmulas no *MS Excel*®. Todas as fórmulas disponíveis no *MS Excel*®, como “Sum()”, “If()”, “And()”, “Or()”, “Pi()” e “Sqrt()”, por exemplo, podem ser usadas em todos os parâmetros de entrada dos componentes. A única ressalva é que as fórmulas devem ser informadas no formato do *MS Excel*® norte-americano, ou seja, valores com decimais precisam usar o ponto como separador, e não a vírgula, e o nome das fórmulas predefinidas do *MS Excel*® devem ser informadas em inglês, independente de que idioma esteja ativo no ambiente do usuário, quer para o pré-processador ou mesmo para o *MS Excel*®. A Figura 30 ilustra o exemplo dado acima, onde a área de escoamento de um volume simples é informada como no exemplo, usando notação de fórmula, enquanto que o comprimento é informado como valor.

Figura 30 - Área de escoamento de um volume simples informada com notação de fórmula

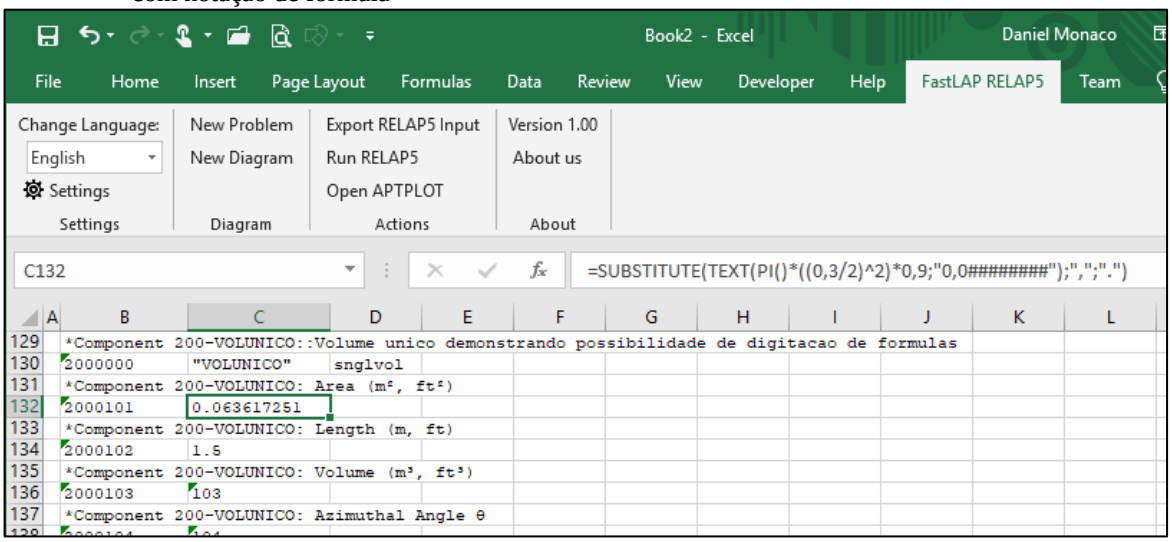


Fonte: autor da dissertação.

Já a Figura 31 mostra como essa fórmula ficará representada na aba de dados de entrada do problema. Nessa figura pode-se notar que, como é de praxe no MS *Excel*®, a célula correspondente à *word* de área de escoamento exibe o valor calculado pela fórmula digitada, enquanto a barra de fórmula exibe a fórmula digitada (acrescida de um

encapsulamento para garantir que o resultado da fórmula seja sempre um número que usa ponto como separador decimal, como é exigido para entrada de dados no RELAP5).

Figura 31 - Detalhe dos dados de entrada de um volume simples, mostrando área de escoamento informada com notação de fórmula



Fonte: autor da dissertação.

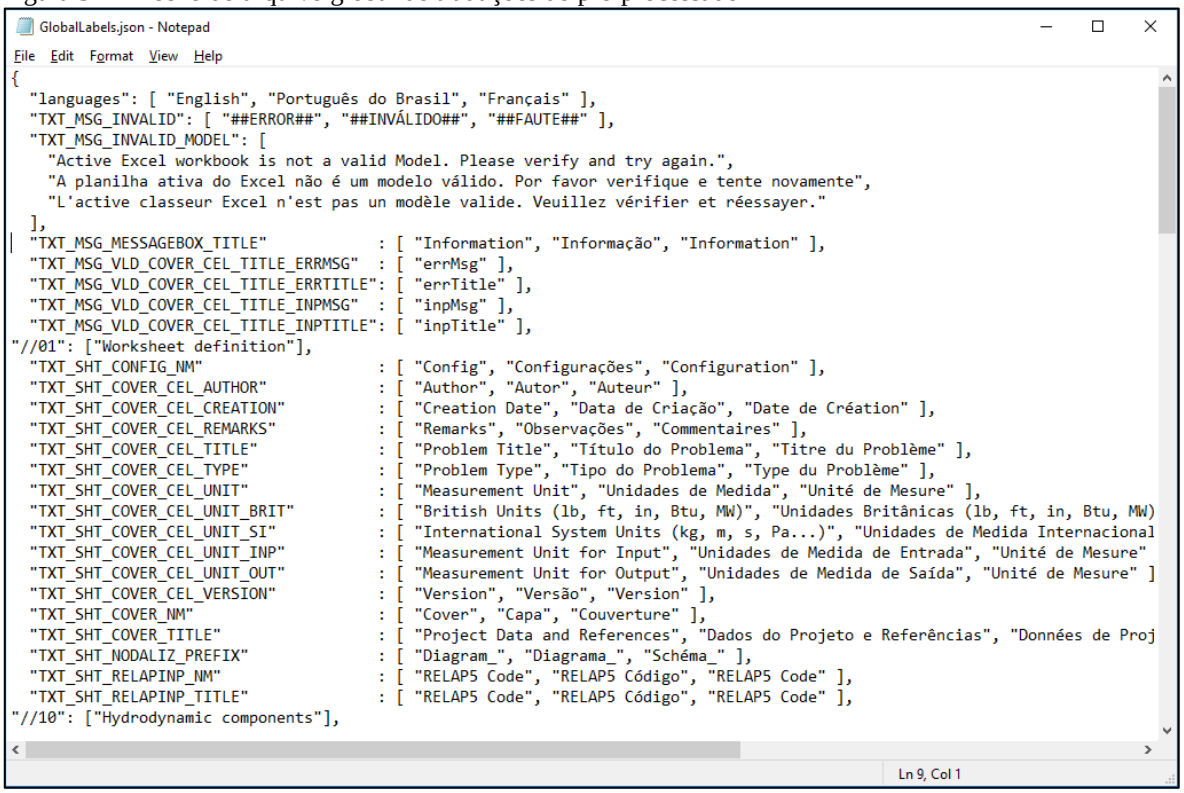
3.2.4.8 Novos idiomas para o pré-processador

Inicialmente, o pré-processador é distribuído em inglês, por ser o mesmo idioma no qual o RELAP5 está escrito; porém, um arquivo externo contendo todas as entradas passíveis de tradução são instaladas na máquina do usuário junto ao o pré-processador. A partir de um simples editor de texto, qualquer usuário pode abrir esse arquivo e fazer as traduções de todo o programa para qualquer idioma sem necessidade de acesso ao programa-fonte. O arquivo de traduções é instalado na mesma pasta selecionada pelo usuário para a instalação do pré-processador.

O arquivo de traduções está no formato JSON, que é uma notação mais sucinta para transcrição de dados de objetos para intercâmbio de informações entre diferentes sistemas, independente de linguagem ou tecnologia. Isso deixa esse arquivo menos prolixo e menos poluído que um arquivo XML, por exemplo, porém o mantém fácil de ser entendido e alterado. Um detalhe do arquivo de traduções pode ser visto na Figura 32.

O arquivo inicialmente disponibilizado é composto de uma coluna representando os rótulos passíveis de tradução, e três colunas onde são informadas as traduções desses rótulos para os idiomas português do Brasil, inglês e francês. Mais colunas com mais idiomas podem ser inseridas no arquivo de traduções, se o usuário assim o desejar. Foram disponibilizadas traduções para o português e para o francês apenas para os elementos das telas principais.

Figura 32 - Trecho do arquivo global de traduções do pré-processador



Fonte: autor da dissertação.

Os rótulos passíveis de tradução referem-se a todos os rótulos de telas existentes dentro do pré-processador: etiquetas de botões, rótulos de telas, títulos de telas, e até mesmo mensagens de erro.

Caso o arquivo de traduções seja atualizado com mais traduções ou com mais idiomas, será necessário reiniciar o pré-processador para que o arquivo de traduções seja novamente lido pelo pré-processador; entretanto não é necessário reiniciar o computador. Caso o usuário deseje alterar o idioma em uso durante a sua utilização, não é necessário reiniciar o pré-processador: basta que o usuário selecione o idioma desejado na caixa de seleção de idiomas disponível na *Ribbon*® do pré-processador, no canto esquerdo da faixa de opções.

Com relação às planilhas de problemas, as traduções são escritas nas mesmas no momento em que são geradas pelo pré-processador, não sendo posteriormente alteradas. Sendo assim, uma planilha criada com o idioma francês permanecerá com seus rótulos de capa em francês, mesmo que o idioma ativo seja alterado. Contudo, ela pode ser usada normalmente com o pré-processador ativo em outro idioma.

3.2.5 Componentes hidrodinâmicos

Na versão atual, os seguintes componentes foram desenvolvidos como resultado deste trabalho: Volume simples (SNGLVOL), Volume dependente do tempo (TMDPVOL), Junção simples (SNGLJUN), Junção dependente do tempo (TMDPJUN), Tubulação (PIPE), Anel (ANNULUS), Pressurizador (PRIZER), Canal CANDU (CANCHAN), Ramificação (BRANCH), *Jetmixer* (JETMIXER), Válvula (VALVE) e Bomba (PUMP).

Além dos componentes hidrodinâmicos, foram também criadas as estruturas de troca de calor (HEAT TRANSFER), propriedades dos materiais das estruturas de troca de calor (HEAT STRUCTURE THERMAL PROPERTIES) e a tabela geral de dados das estruturas de troca de calor (GENERAL TABLE DATA).

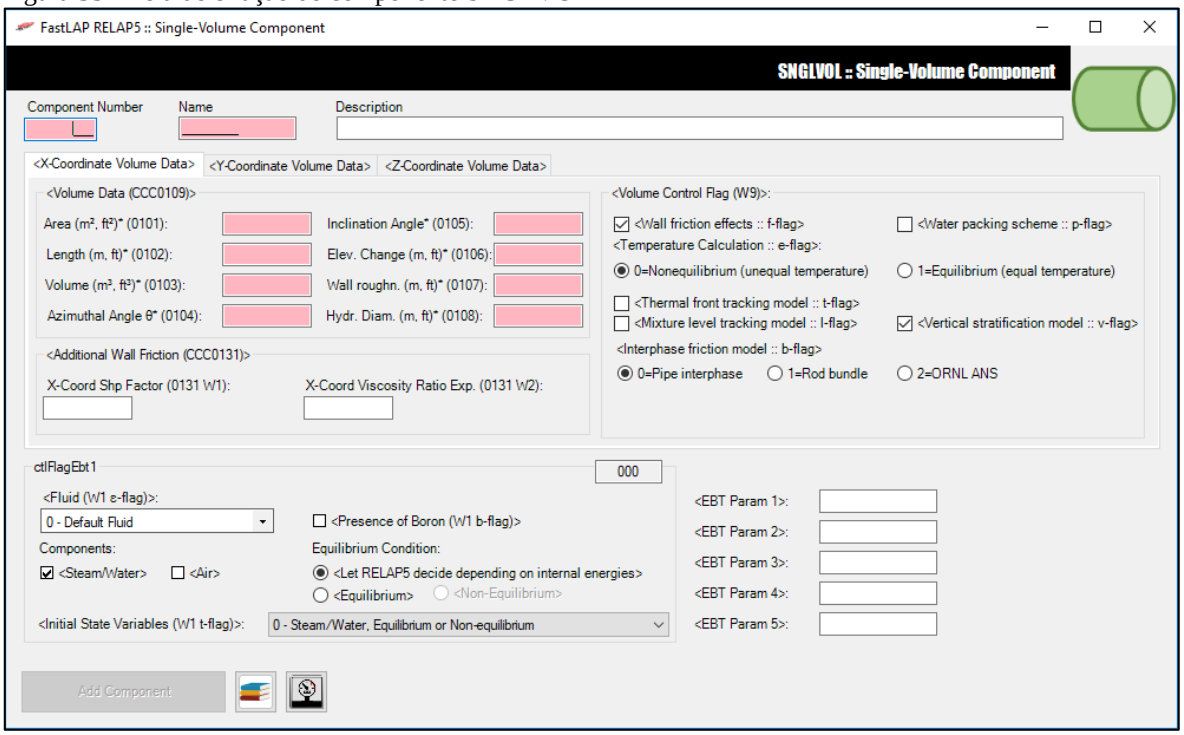
Nos próximos itens pode ser visto cada uma dessas telas desenvolvidas com mais detalhes.

3.2.5.1 Volume simples (SNGLVOL)

O volume simples, ou SNGLVOL, é usado para representar um volume hidrodinâmico genérico. Em geral, a vazão dentro do volume ocorre em um único eixo (X), porém o RELAP5 permite a representação de vazão também nos demais eixos. Como praticamente todos os cartões relacionados ao volume e ao eixo X são obrigatórios, e as informações dos demais eixos são opcionais, as informações dessa tela foram organizadas em abas separadas pelos eixos cartesianos, tornando mais intuitiva a configuração do componente. A Figura 33 mostra a tela principal de configuração de um componente SNGLVOL, enquanto as Figuras 34 e 35 mostram a aba de configuração das informações relacionadas aos eixos Y e Z, respectivamente.

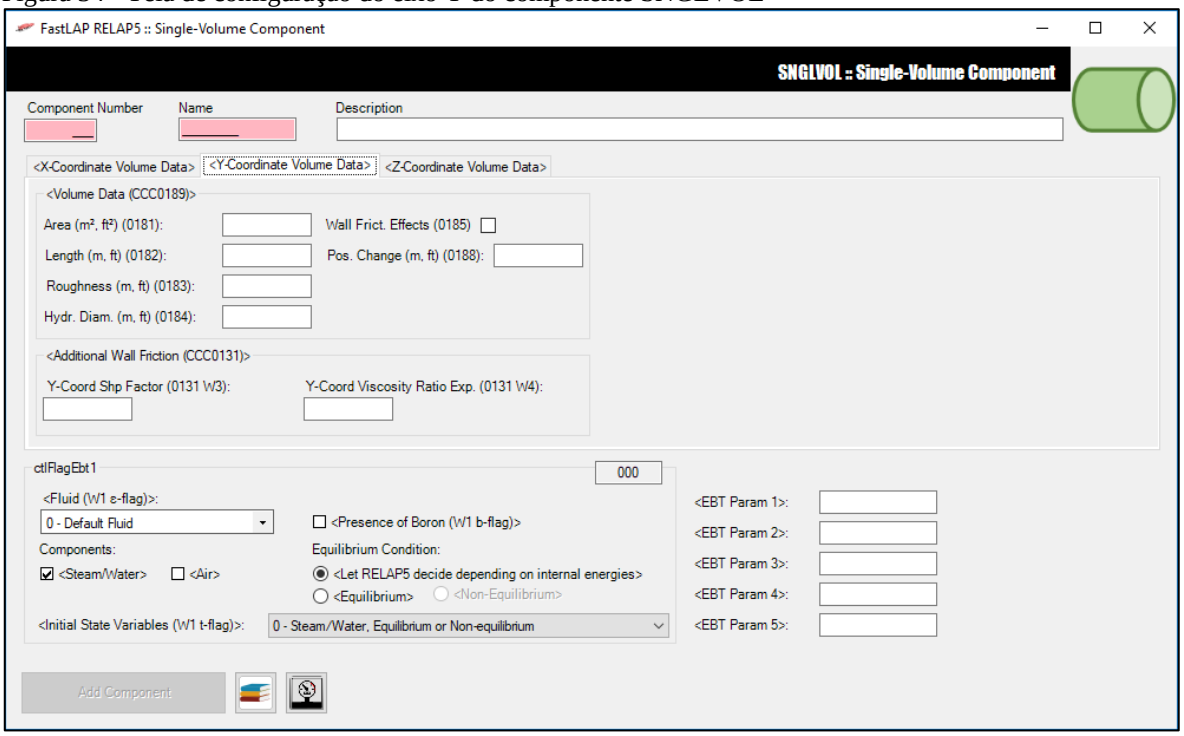
O *flag* de controle é uma informação relacionada a todo o volume, e não a um determinado eixo; por essa razão o controle desse *flag* fica visível sempre, não importando qual eixo está sendo configurado no momento.

Figura 33 - Tela de criação do componente SNGLVOL



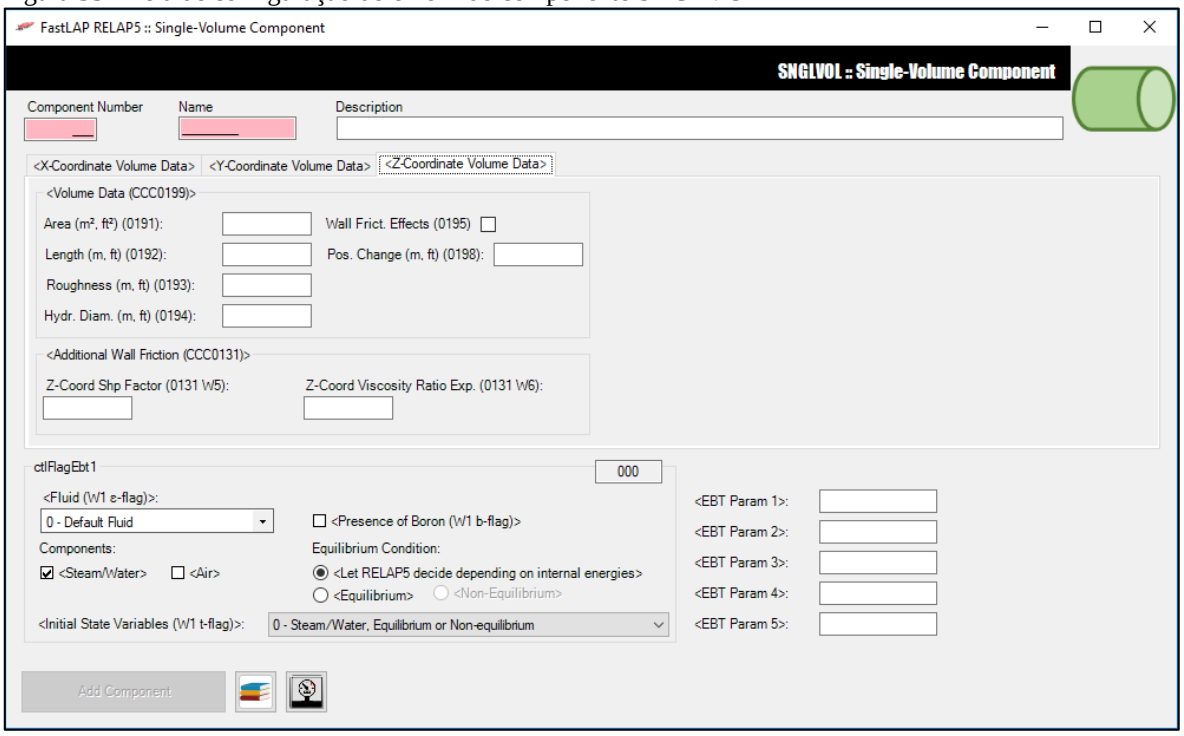
Fonte: autor da dissertação.

Figura 34 - Tela de configuração do eixo Y do componente SNGLVOL



Fonte: autor da dissertação.

Figura 35 - Tela de configuração do eixo Z do componente SNGLVOL



Fonte: autor da dissertação.

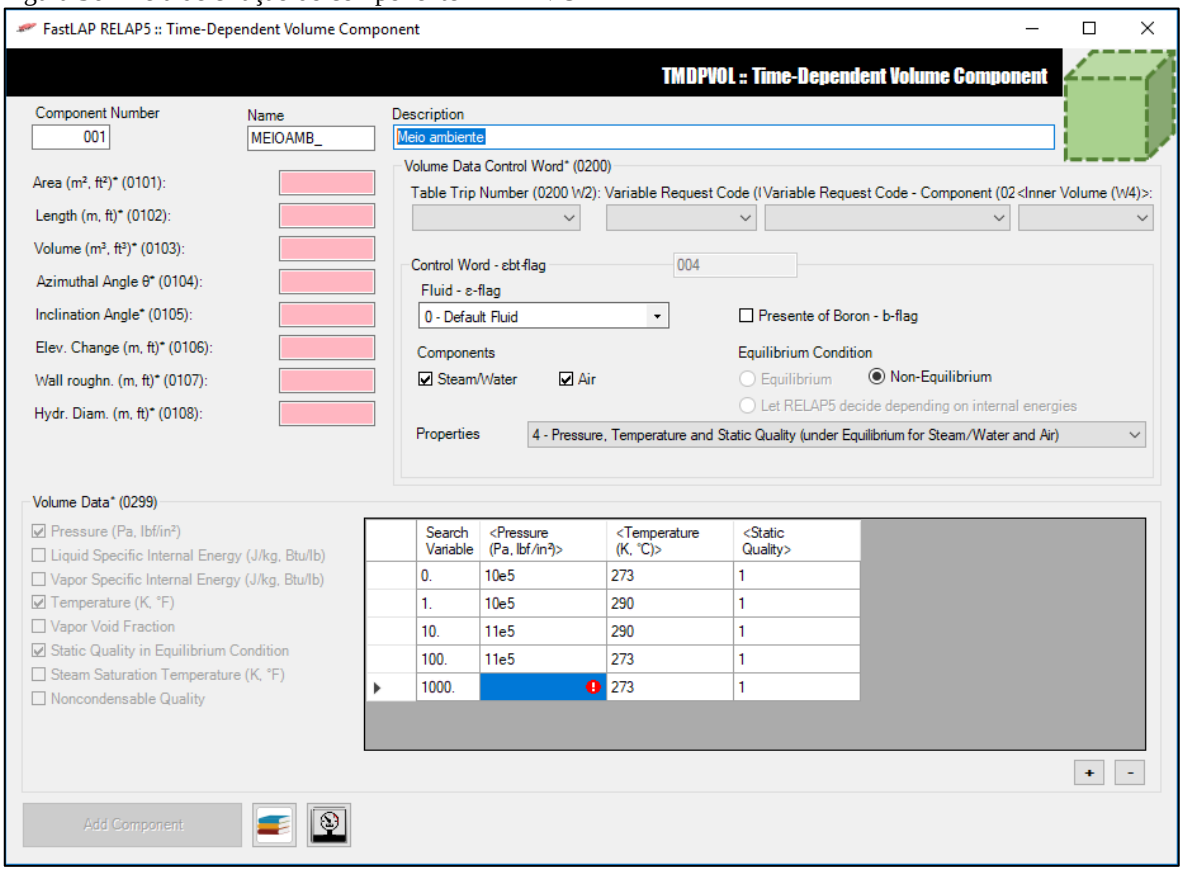
3.2.5.2 Volume dependente do tempo (TMDPVOL)

O volume dependente do tempo, ou TMDPVOL, é um componente usado em geral para criar volumes de condições de contorno, como o meio ambiente, por exemplo. Nesse tipo de volume é possível informar como as suas condições operacionais irão evoluir conforme o decorrer do tempo da simulação com o RELAP5.

Para configurar o componente adequadamente, o usuário do RELAP5 precisa informar as medidas do volume, além de informar nos cartões do tipo tabela como determinadas condições operacionais variam com o tempo, tais como temperatura, fração de vazio, pressão, entre outras. Para esse fim, a tela de criação do TMDPVOL conta com um controle *grid*, permitindo ao usuário uma melhor visualização dessa série de dados que serão inseridos nos cartões de tabela.

A Figura 36 exibe a tela de criação de um TMDPVOL.

Figura 36 - Tela de criação do componente TMDPVOL



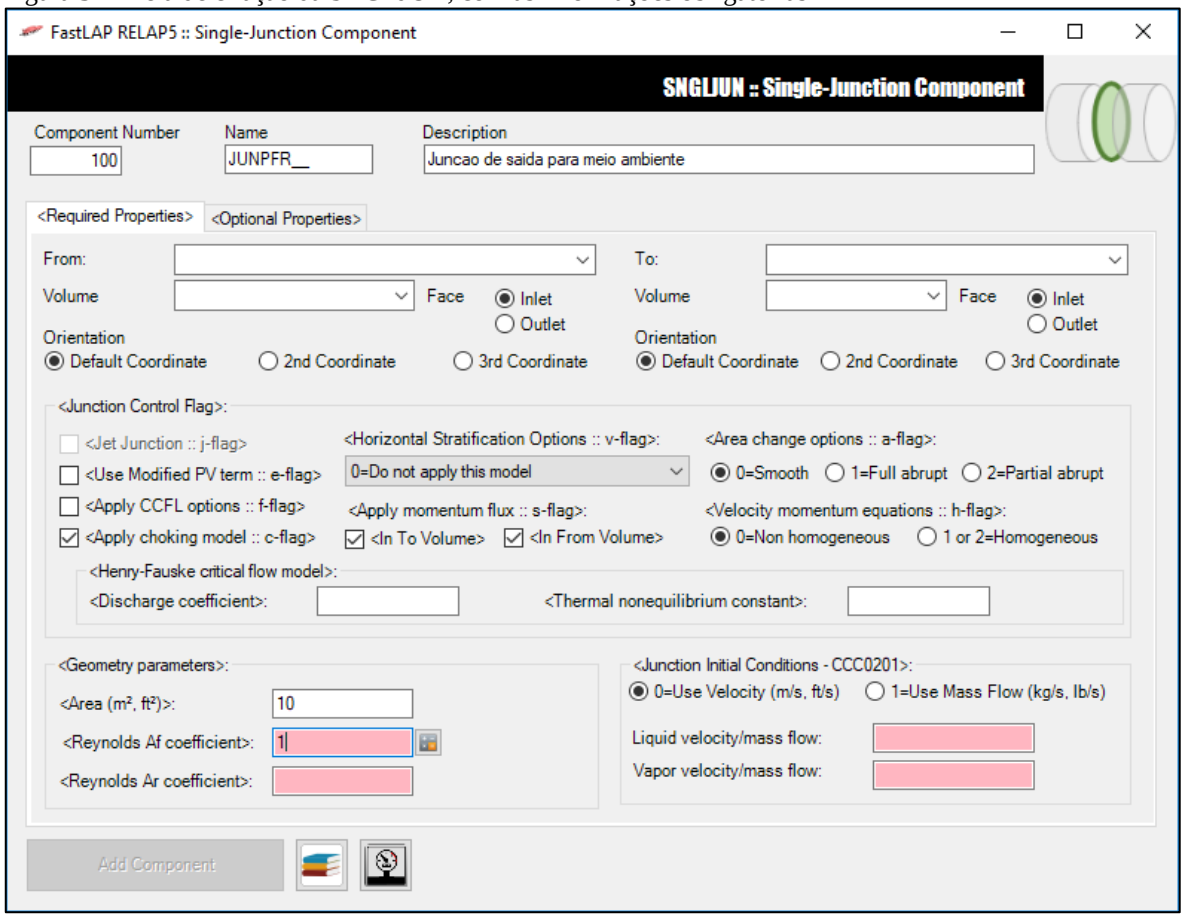
Fonte: autor da dissertação.

3.2.5.3 Junção simples (SNGLJUN)

A junção simples, ou SNGLJUN, é um elemento que representa a conexão entre dois volumes. Nele são configuradas informações relacionadas à sua geometria e a vazão de fluido através dele.

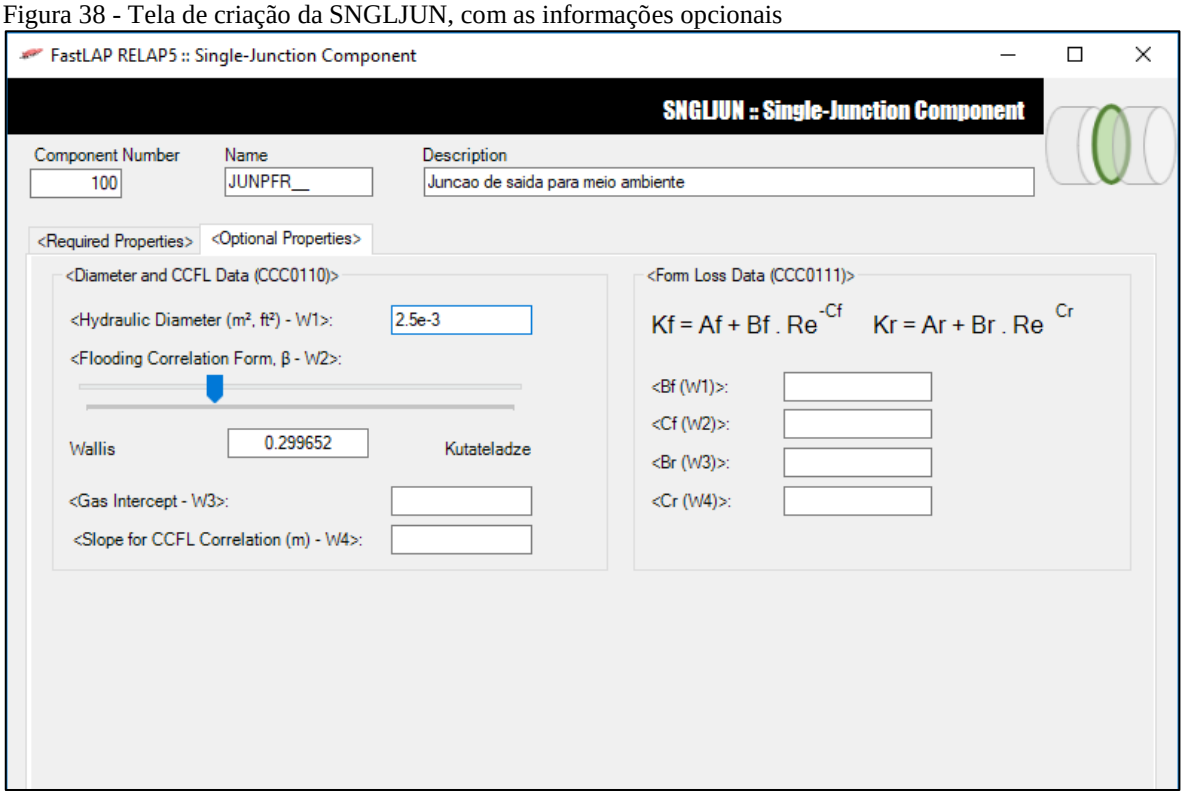
Por ser um componente do tipo conector, duas das informações mais importantes são os volumes de origem e de destino da junção. Ao informar esses dois parâmetros, o pré-processador faz, por meio de setas, a representação gráfica da vazão do fluido entre os componentes relacionados. A Figura 37 mostra a tela com as informações de preenchimento obrigatório para criação de uma junção, enquanto a Figura 38 apresenta a tela com as informações de preenchimento opcional.

Figura 37 - Tela de criação da SNGLJUN, com as informações obrigatórias



Fonte: autor da dissertação.

Figura 38 - Tela de criação da SNGLJUN, com as informações opcionais



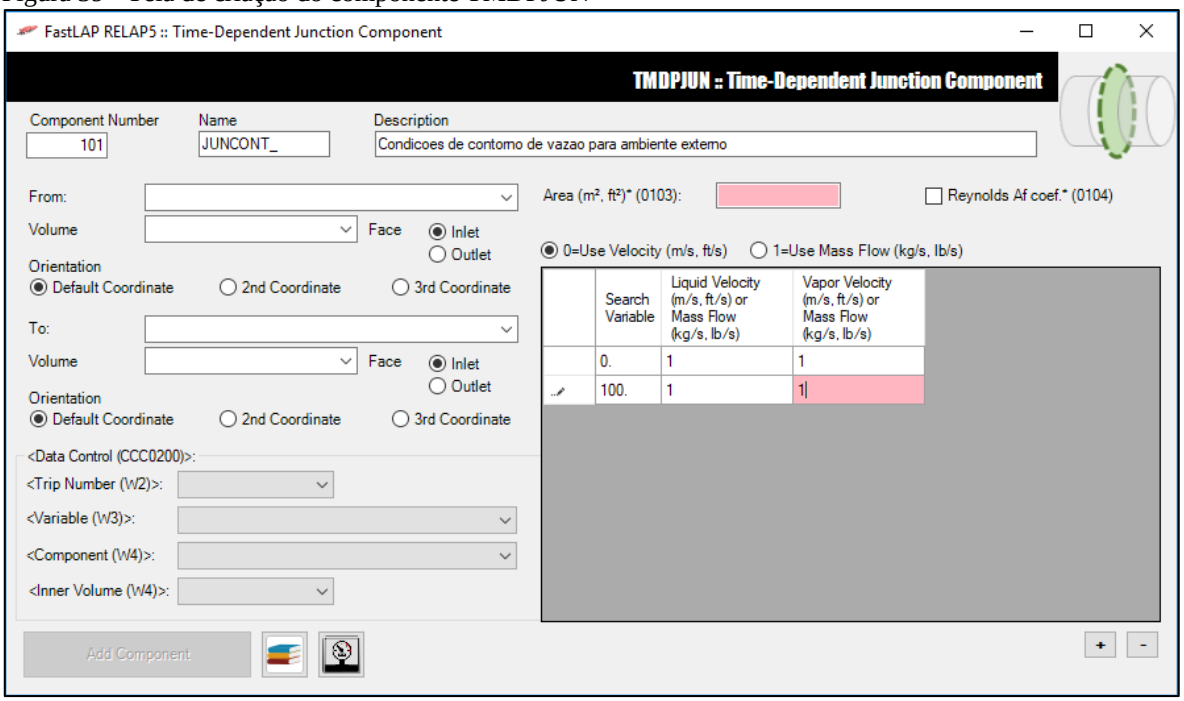
Fonte: autor da dissertação.

3.2.5.4 Junção dependente do tempo (TMDPJUN)

De maneira análoga ao que ocorre com o volume, a junção também possui um componente hidrodinâmico no RELAP5 para representar condições de contorno para junções. Esse componente é a junção dependente do tempo, ou TMDPJUN.

A tela de criação desse componente possui, portanto, também um controle do tipo *grid* muito similar ao existente na criação do TMDPVOL, como pode ser visto na Figura 39.

Figura 39 - Tela de criação do componente TMDPJUN



Fonte: autor da dissertação.

3.2.5.5 Tubulação (PIPE)

Um dos componentes mais utilizados para simulação de transientes no código RELAP5 é o componente de tubulação, conhecido como PIPE. Por meio desse componente é possível configurar uma tubulação inteira, informando em um único componente todos os volumes internos e junções internas. O resultado é equivalente à criação de inúmeros SNGLVOL e SNGLJUN; porém, o componente PIPE permite essa criação de maneira bem mais enxuta e eficiente, facilitando a manutenção dessas informações no futuro.

O recurso que permite ao PIPE tamanha flexibilidade na criação de múltiplos volumes e junções integrados em um único componente é chamado pelo RELAP5 de “expansão”. Com o recurso da expansão, o RELAP5 compreende que informações comuns a um ou mais volumes conjugados serão informadas em um único cartão. No início na execução do problema, o código RELAP5 obtém essa informação do cartão de entrada e faz

a expansão dessa propriedade compartilhada, replicando-a para todos os volumes ou junções aos quais a propriedade se aplica.

Para facilitar o entendimento do critério de expansão utilizado pelo RELAP5, vamos tomar como exemplo a tubulação da "Experiência SUPER CANON", um tubo único de 4,389m. Um usuário do RELAP5 poderia decidir subdividir essa tubulação em trinta volumes de controle da seguinte maneira: os dez primeiros volumes terão 10cm de comprimento, os oito volumes seguintes terão 20cm de comprimento, os próximos cinco volumes com novamente 10cm de comprimento, seguidos de seis volumes de 15cm cada e, para finalizar, um último volume de 38,9cm.

Os comprimentos dos volumes de controle de um PIPE são informados ao código RELAP5 por meio do conjunto de cartões CCC0301-0399. Para a configuração do experimento mencionado, todavia, não é necessário informar para o RELAP5 trinta cartões de entrada contendo os comprimentos volume a volume, mas informar apenas cinco cartões. Cada um desses cinco cartões irá informar ao código RELAP5 o comprimento do volume desejado e o número do último volume do grupo. No caso, tem-se o primeiro cartão CCC0301 informando que o volume 10 possui 0,1m, o segundo cartão informando que o volume 18 possui 0,2m, terceiro cartão informando que o volume 23 possui 0,1m, o quarto cartão informando que o volume 29 possui 0,15m, e o último cartão, CCC0305, informando que o volume 30 possui 0,389m. O código RELAP5 processa esses cartões de entrada e identifica que não foram informados os comprimentos de todos os volumes configurados no PIPE, e expande as informações dos cartões subsequentes para os volumes cujas informações foram suprimidas. No caso, como o primeiro cartão de comprimento informa o comprimento do volume 10, sem informar o comprimento de nenhum volume anterior, o RELAP5 entende que o cartão em questão serve para informar o comprimento de todos os volumes compreendidos do número 1 ao 10; e assim sucessivamente.

Essa característica se aplica a cada um dos conjuntos de cartões de entrada de dados de volumes e junções separadamente. Ainda seguindo o raciocínio exposto anteriormente, o código RELAP5 recebe a informação das áreas de escoamento dos volumes de controle através do conjunto de cartões CCC0101-0199. Para o usuário informar que a área de escoamento de todos os volumes de controle dessa mesma tubulação é de $10cm^2$, por exemplo, não será necessário criar outros cinco subconjuntos de cartões de área; uma vez que a expansão é feita pelo RELAP5 para cada subconjunto de cartões de propriedade, bastaria o usuário informar num único cartão CCC0101 que o volume 30 possui área de

escoamento de $0.001m^2$, e o código RELAP5 expandiria esse valor para todos os outros vinte e nove volumes que não foram mencionados nesse conjunto de cartões de área.

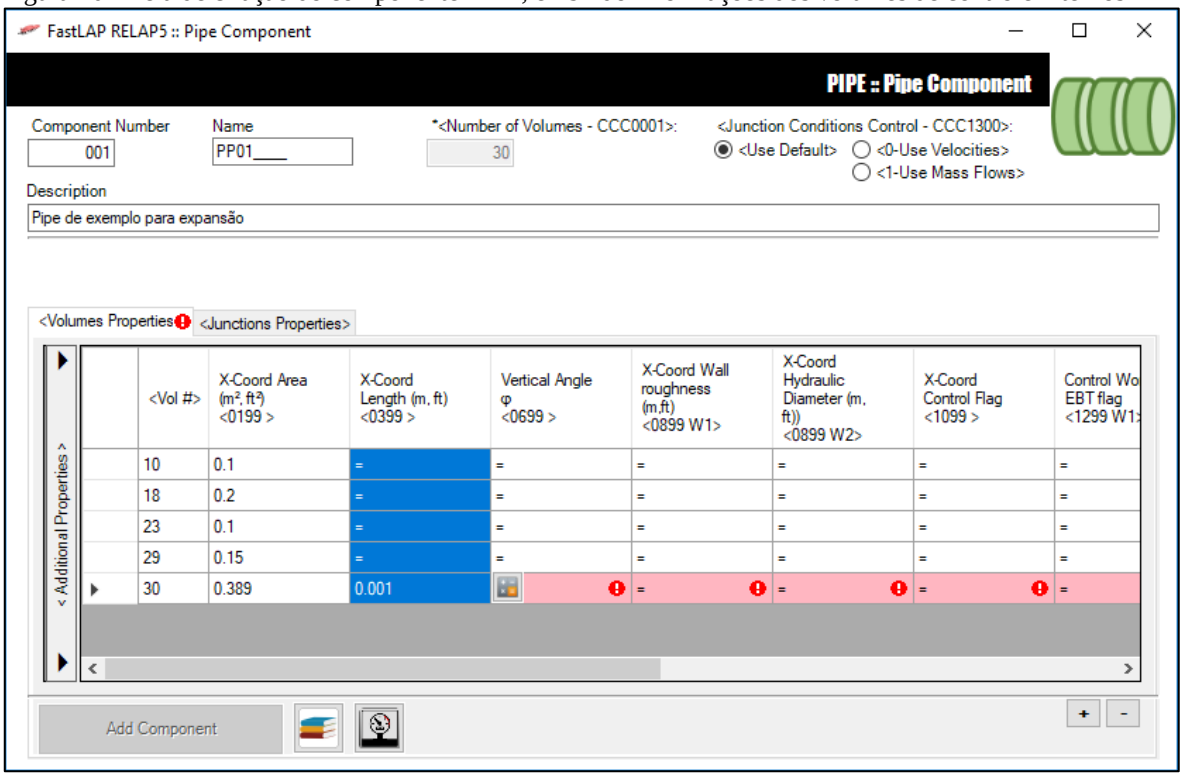
Se, por um lado, essa característica de expansão facilita a configuração de problemas no RELAP5, por outro, ela dificulta em muito a criação de uma interface que possa ser simples, mas ao mesmo tempo possa oferecer o mesmo grau de comodidade que o usuário RELAP5 está acostumado. Como cada conjunto de cartões de propriedade é tratado pelo código RELAP5 individualmente, o mais provável seria uma interface com uma infinidade de tabelas, uma para cada propriedade de volumes e de junções. Porém, o mais provável é que fazendo isso a interface ficaria completamente poluída, e o usuário teria mais dificuldade em usar essa tela do que ele já tem diretamente no RELAP5, e não é essa a intenção por trás deste pré-processador.

Uma outra alternativa seria uma tabela única, onde todos os dados seriam informados para todos os volumes: nesse caso, apesar desse fato simplificar a confecção da tela, o usuário acabaria perdendo um recurso importante do RELAP5 que reduz a complexidade dos cartões de entrada.

A alternativa identificada pelo autor desse trabalho foi a de criar um único controle *grid* para as informações relacionadas com os volumes de controle, e outro controle *grid* para as informações das junções. Nesse *grid*, cada linha representa um volume (ou junção, na *grid* correspondente), e cada coluna representa uma propriedade. Cartões de entrada que possuem apenas uma *word* além da informação do número do volume ou junção serão representados apenas em uma coluna na *grid*, cartões que possuem mais *words* terão uma coluna para cada *word* adicional.

Além disso, foi criado no pré-processador um recurso que imita o recurso de expansão usado pelo RELAP5, mas que permite o usuário informar todos os dados de um determinado volume ou junção em uma única estrutura de tabelas. Para atingir essa feita, criou-se o caractere de controle “=” (sinal de igual). Ao digitar o sinal de igual em determinada coluna, o usuário informa ao pré-processador que a propriedade para aquele volume representado naquela linha é igual à mesma propriedade do volume subsequente, ou seja, da linha abaixo da mesma *grid*. Configurando o PIPE desta forma, o usuário irá ter um conjunto de cartões de entrada muito similar ao que teria caso estivesse criando os cartões manualmente, sem ocorrência de informações repetidas desnecessariamente. A Figura 40 ilustra a tela de criação do componente PIPE, mostrando os dados de comprimento e área preenchidas conforme o exemplo citado acima.

Figura 40 - Tela de criação do componente PIPE, exibindo informações dos volumes de controle internos



Fonte: autor da dissertação.

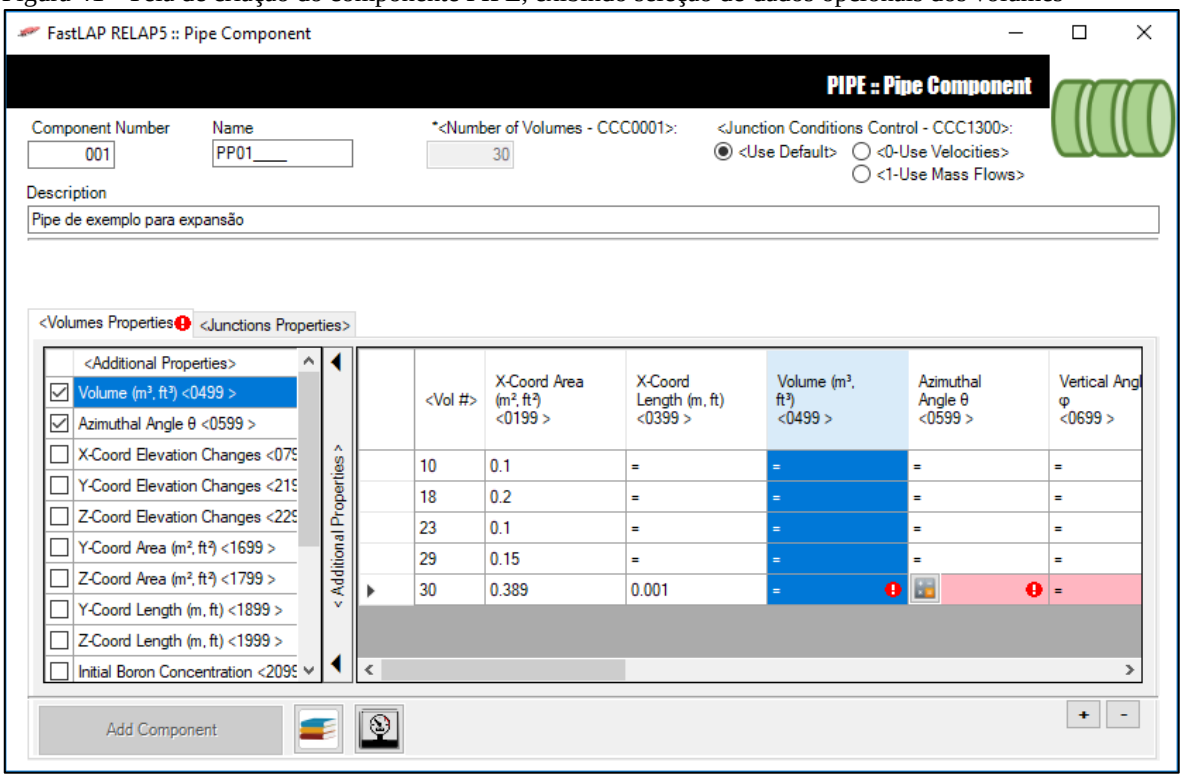
Na criação do PIPE, algumas informações são deduzidas pelo pré-processador, e o próprio se encarrega de informar isso no código de entrada do RELAP5. Por exemplo, ao preencher a *grid* de volumes informando um volume 30 como o maior número de volume, o pré-processador já entende que o PIPE em questão terá trinta volumes de controle e vinte e nove junções: essa informação será criada automaticamente pelo pré-processador no código de entrada, não sendo necessário informá-la, ficando disponível na tela para facilitar a consulta e conferência.

Inicialmente, ao abrir a tela para criação de um PIPE, somente os dados obrigatórios para criação do componente são carregados na *grid*; todavia, todos os dados do PIPE estão disponíveis para os usuários preencherem. Para isso, basta que o usuário expanda a caixa de seleção de dados adicionais, usando o botão vertical ao lado esquerdo da *grid*. Fazendo isso, uma caixa de seleção se abrirá, e cada dado opcional selecionado é automaticamente inserido na *grid* ao lado, para ser digitado e ter seu cartão de entrada gerado, como pode ser visto na Figura 41.

O tamanho da caixa de seleção de dados opcionais pode ser facilmente alterado, bastando somente que o usuário clique sobre o botão de dados adicionais e arraste-o sobre a *grid* até que a caixa obtenha o tamanho desejado.

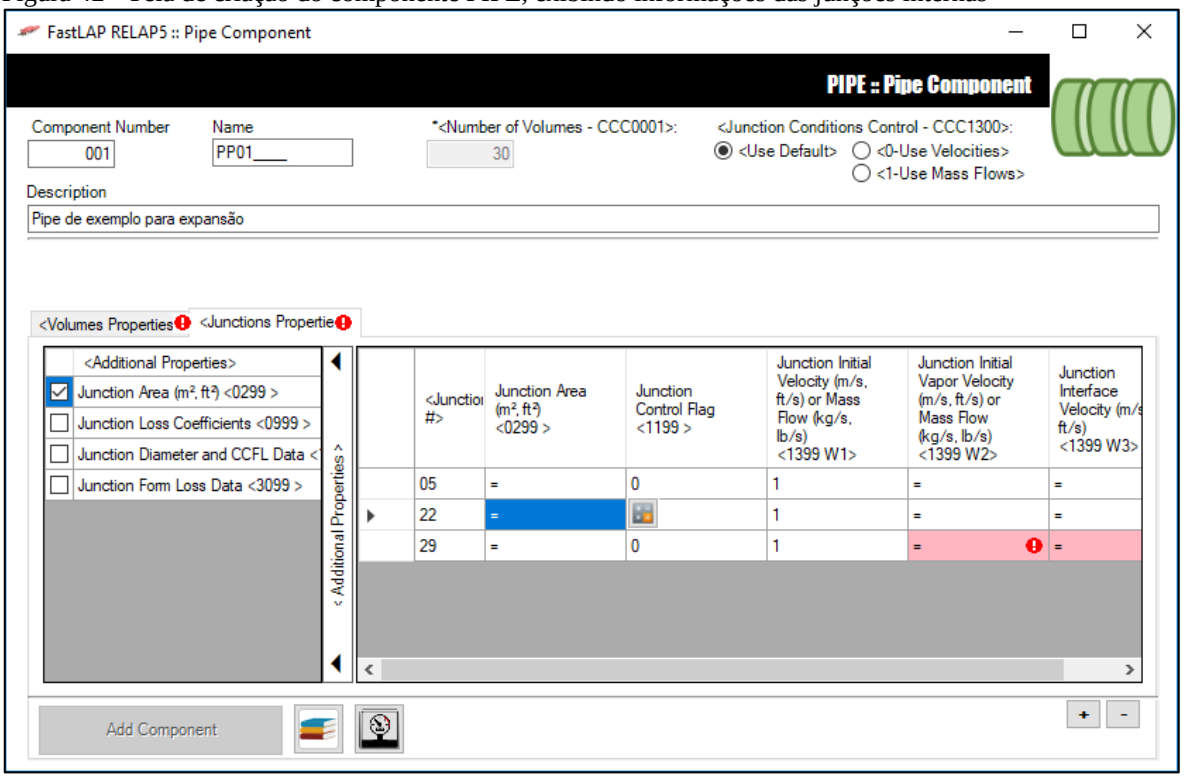
Os dados das junções irão funcionar de maneira muito similar à que ocorre com os volumes, como pode ser visto na Figura 42.

Figura 41 - Tela de criação do componente PIPE, exibindo seleção de dados opcionais dos volumes



Fonte: autor da dissertação.

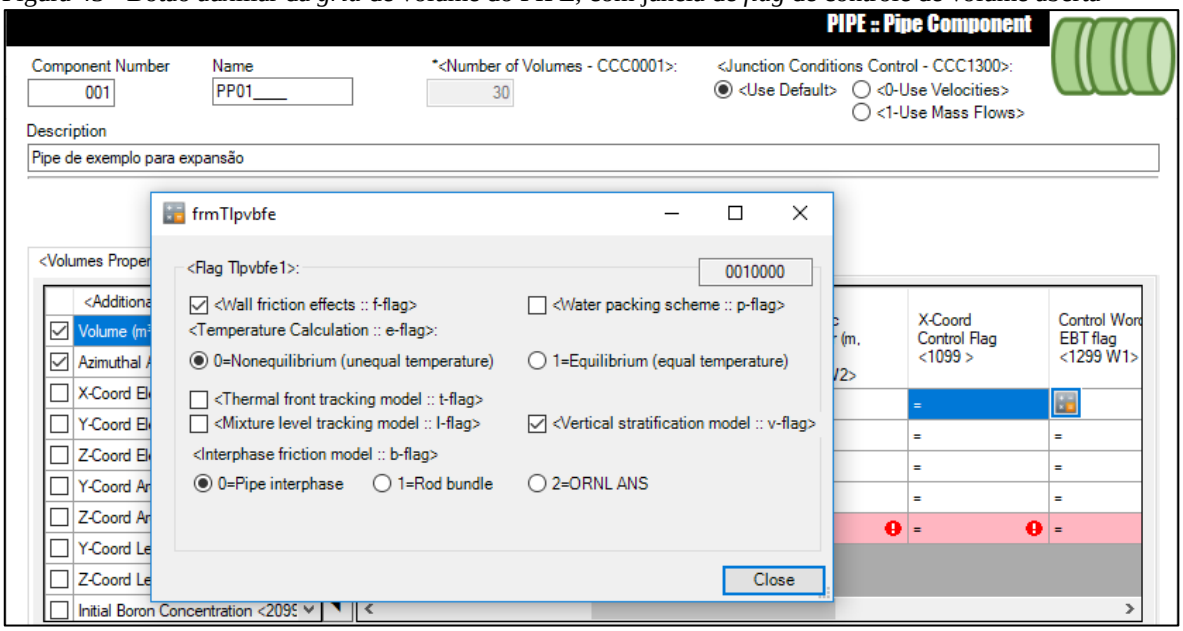
Figura 42 - Tela de criação do componente PIPE, exibindo informações das junções internas



Fonte: autor da dissertação.

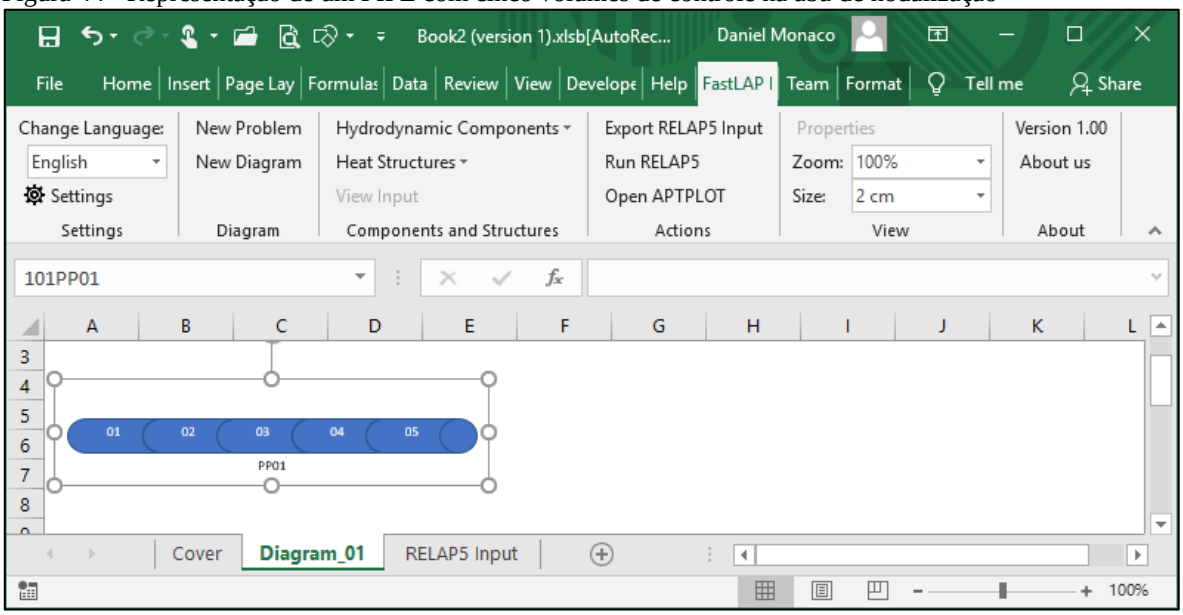
Tanto a *grid* de dados de volume quanto de junção possui a funcionalidade do botão “calculadora”. Nas colunas que permitem inserção de valores numéricos, ao clicar no botão “calculadora” é aberta a calculadora de grandezas geométricas, conforme descrito no item 3.2.4.6. Entretanto, caso a coluna selecionada seja uma coluna de entrada de um *flag* de controle, por exemplo, ao clicar na “calculadora” será aberta uma janela de composição do parâmetro em questão, ajudando o usuário no preenchimento adequado do mesmo, como demonstrado na Figura 43.

Figura 43 - Botão auxiliar da *grid* de volume do PIPE, com janela de *flag* de controle de volume aberta



Fonte: autor da dissertação.

Figura 44 - Representação de um PIPE com cinco volumes de controle na aba de nodalização



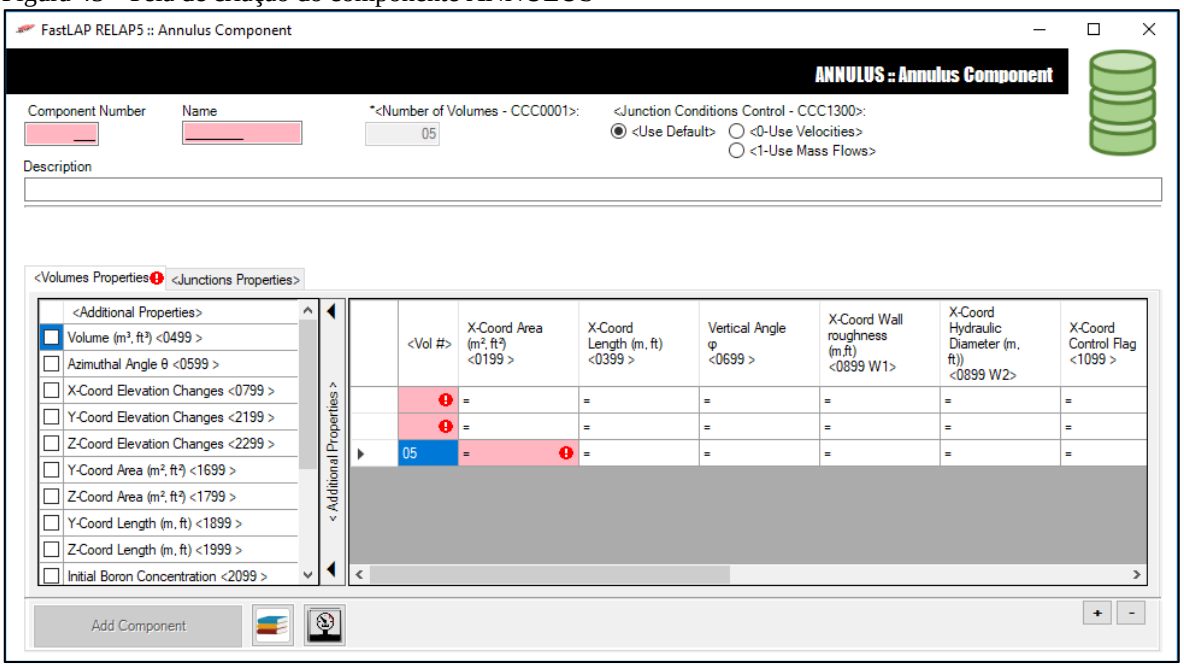
Fonte: autor da dissertação.

Uma vez que todos os campos obrigatórios são corrigidos, o botão para inserção do componente no problema é habilitado para uso e, uma vez clicado, o PIPE configurado é desenhado na tela de diagrama. Uma observação interessante é que o PIPE inserido no diagrama é representado com a mesma quantidade de *AutoShapes* que ele possui de volumes de controle, agrupados em um único elemento. A Figura 44 mostra um PIPE com cinco volumes de controle representados na área de nodalização.

3.2.5.6 Anel (ANNULUS)

O componente do tipo Anel, ou ANNULUS, é muito similar ao componente PIPE, com exceção de que esse componente obrigatoriamente precisa estar posicionado na vertical e com água contida dentro dele (ou seja, sem transbordamento). Dessa forma, toda a implementação da criação de um ANNULUS foi feita utilizando a mesma tela criada para o PIPE, como pode ser visto na Figura 45.

Figura 45 - Tela de criação do componente ANNULUS

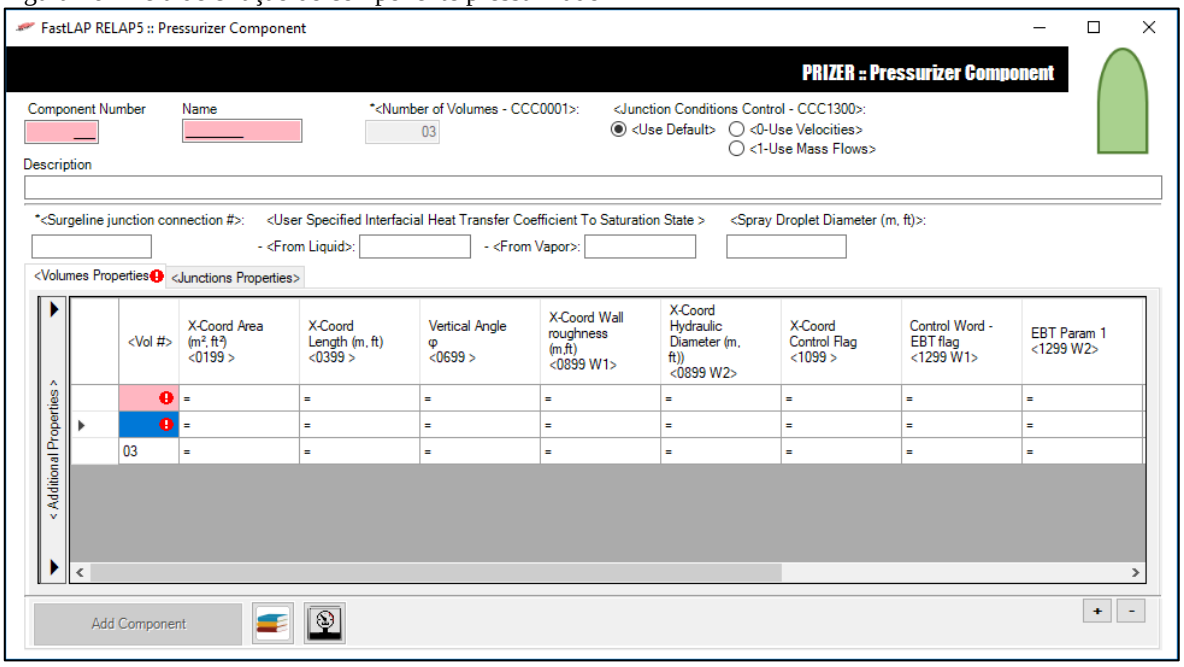


Fonte: autor da dissertação.

3.2.5.7 Pressurizador (PRIZER)

O componente pressurizador também possui atributos muito semelhantes aos do PIPE, e por isso foi implementado usando a mesma janela de criação do PIPE, como pode ser visto na Figura 46.

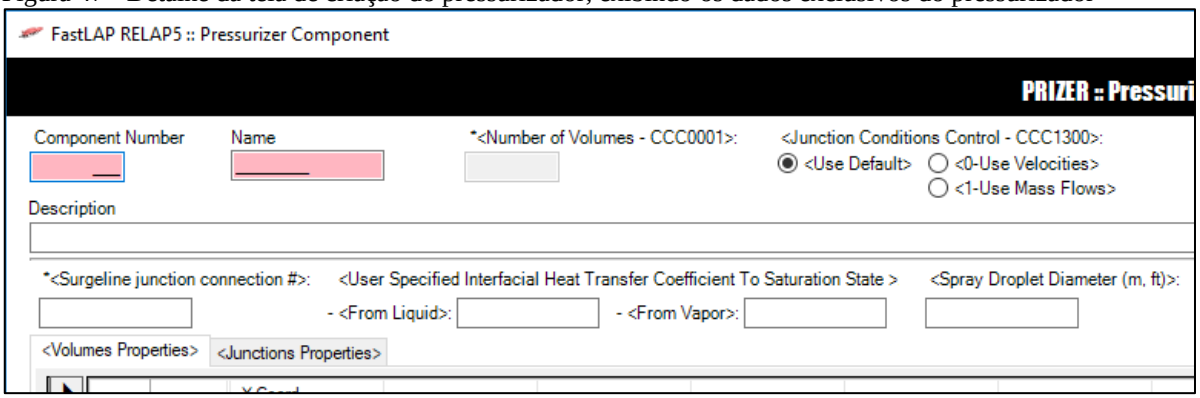
Figura 46 - Tela de criação do componente pressurizador



Fonte: autor da dissertação.

Como esse componente possui, todavia, alguns dados exclusivos de um pressurizador que não estão vinculados a nenhum volume ou junção em particular, ao abrir essa janela as caixas de texto para digitação desses dados aparecem para o usuário; fato esse que não ocorre quando o usuário deseja inserir um PIPE. Os dados exclusivos do pressurizador podem ser vistos em detalhe na Figura 47.

Figura 47 - Detalhe da tela de criação do pressurizador, exibindo os dados exclusivos do pressurizador



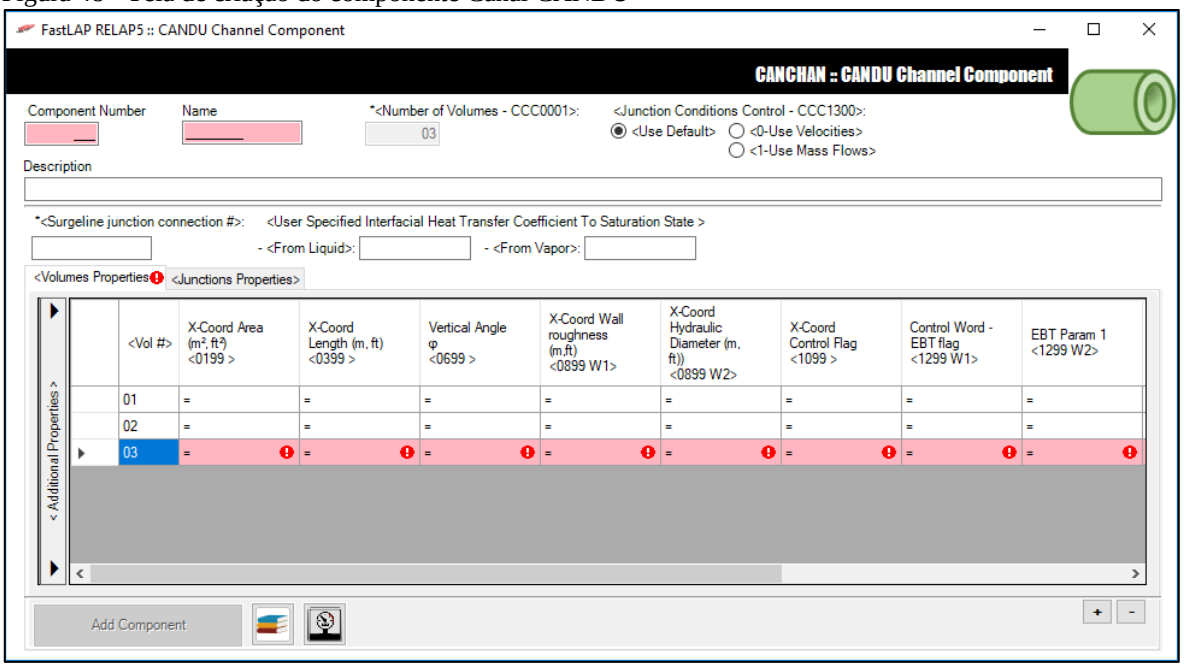
Fonte: autor da dissertação.

3.2.5.8 Canal CANDU (CANCHAN)

De maneira análoga ao que ocorre na criação de um pressurizador, um componente do tipo CANCHAN (Canal CANDU) também é um componente muito semelhante ao pressurizador, porém com a exceção de possuir alguns dados exclusivos.

Desta forma, sua criação se dá por meio de uma janela muito semelhante à do pressurizador, como ilustra a Figura 48.

Figura 48 - Tela de criação do componente Canal CANDU



Fonte: autor da dissertação.

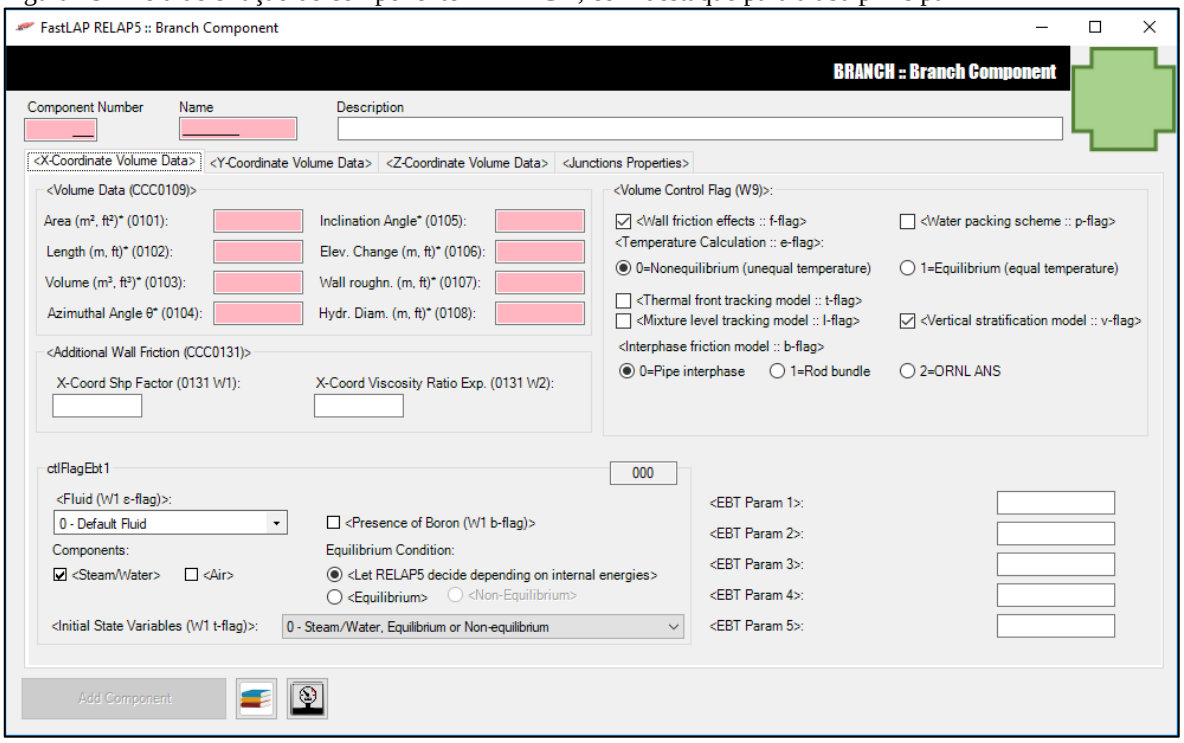
3.2.5.9 Ramificação (BRANCH)

Um componente de BRANCH (ramificação) nada mais é para o RELAP5 que um componente SNGLVOL, mas que permite a configuração de inúmeras junções dentro dele. O número máximo de junções que podem ser criadas num BRANCH é de nove junções.

Devido a esse fator, a tela de criação de um BRANCH foi pensada para ajudar o usuário a raciocinar desta forma enquanto modela seu BRANCH:

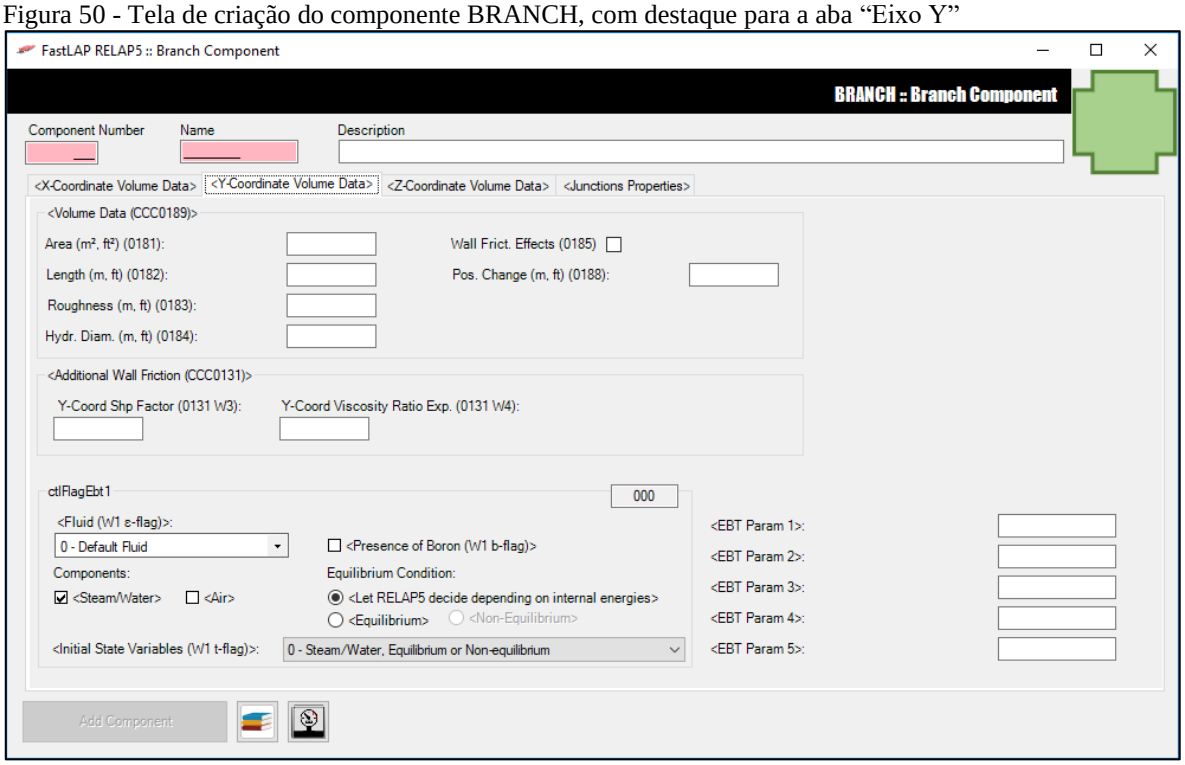
- uma aba inicial contendo as informações do volume do BRANCH relacionadas ao eixo X, que em geral são obrigatórias na configuração desse componente, como pode ser visto na Figura 49;
- duas abas seguintes, com informações opcionais para o volume orientadas aos eixos Y e Z, como pode ser visto nas Figuras 50 e 51;
- uma última aba com as informações para configuração de junções, com interface muito similar à configuração de junções para um componente do tipo PIPE (porém, aqui, limitado à criação de nove junções). Essa tela pode ser vista na Figura 52.

Figura 49 - Tela de criação do componente BRANCH, com destaque para a aba principal



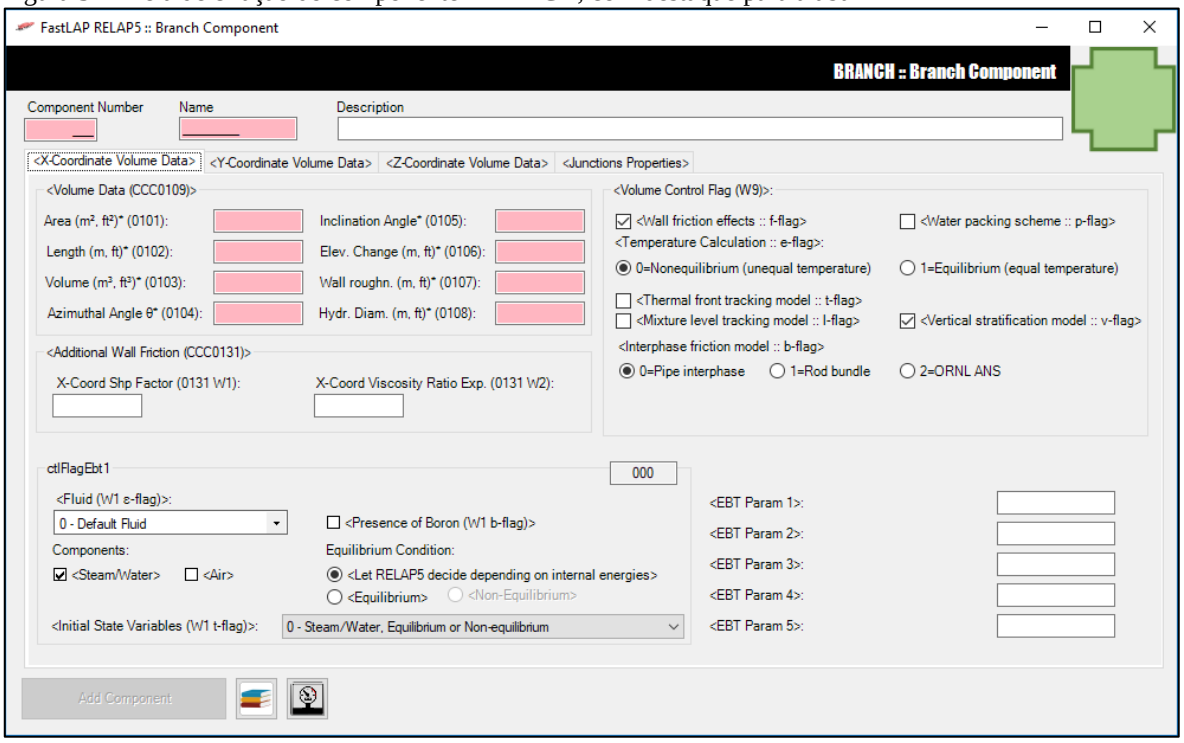
Fonte: autor da dissertação.

Figura 50 - Tela de criação do componente BRANCH, com destaque para a aba “Eixo Y”



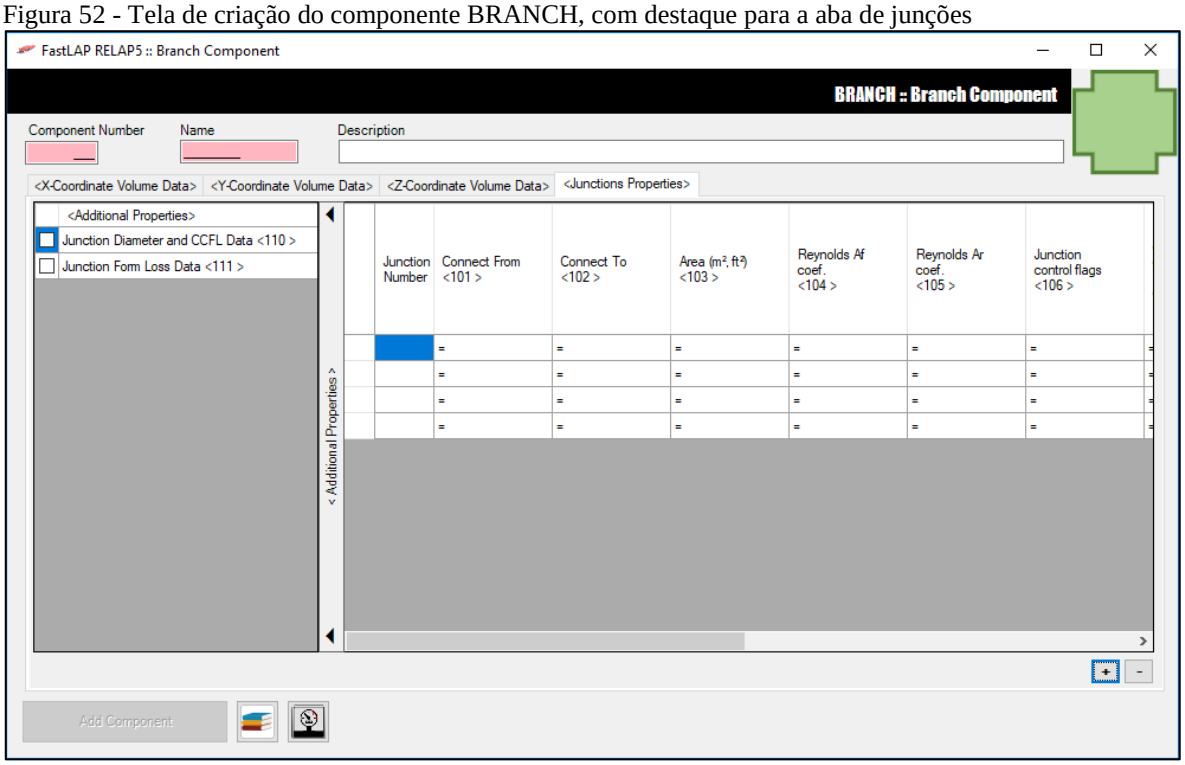
Fonte: autor da dissertação.

Figura 51 - Tela de criação do componente BRANCH, com destaque para a aba “Eixo Z”



Fonte: autor da dissertação.

Figura 52 - Tela de criação do componente BRANCH, com destaque para a aba de junções



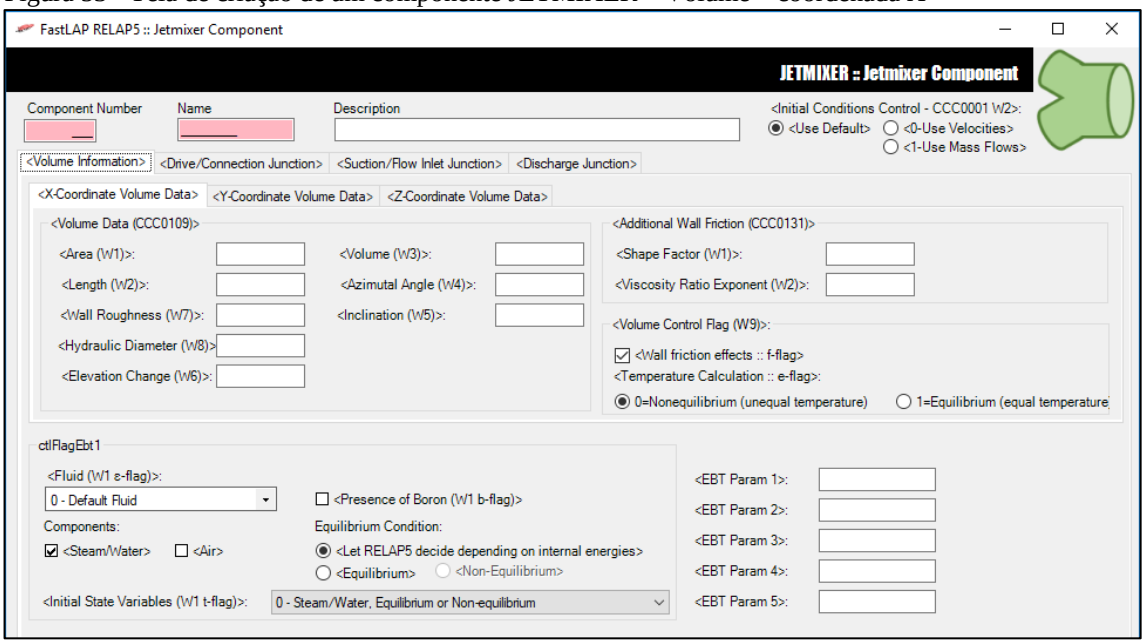
Fonte: autor da dissertação.

3.2.5.10 *Jetmixer* (JETMIXER)

O componente JETMIXER é um tipo especializado de BRANCH que usa três junções. A primeira junção representa a junção *drive*, a junção dois representa a junção de sucção, e a terceira junção é a junção de descarga (*discharge junction*).

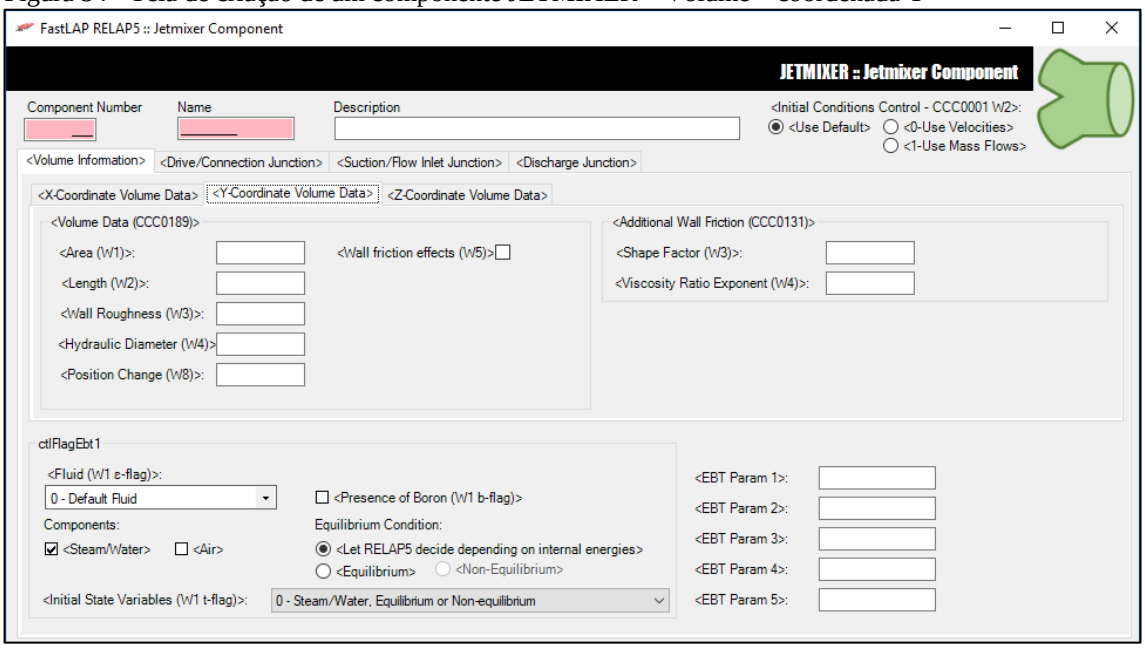
Como a tela de criação do BRANCH permite a criação de até nove junções, porém junções essas sem significado próprio, optou-se por desenhar uma tela específica para a criação do JETMIXER, como pode ser visto nas Figuras 53 a 58.

Figura 53 - Tela de criação de um componente JETMIXER – Volume – coordenada X



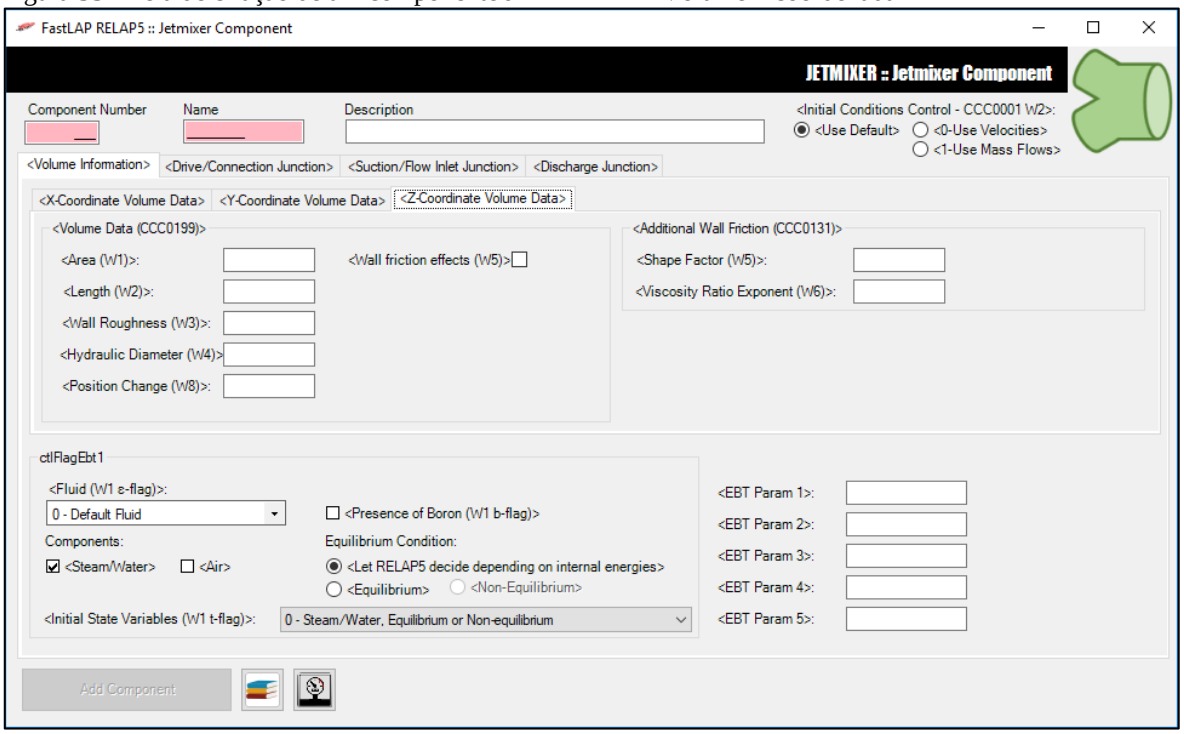
Fonte: autor da dissertação.

Figura 54 - Tela de criação de um componente JETMIXER – Volume – coordenada Y



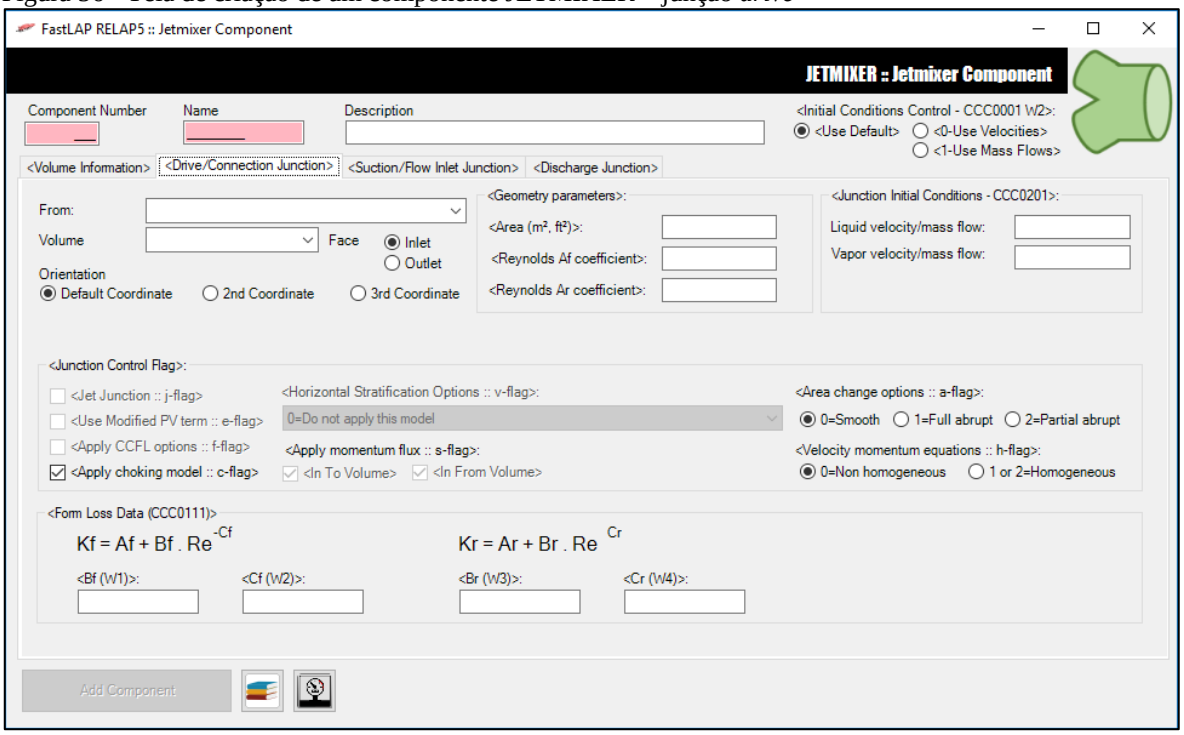
Fonte: autor da dissertação.

Figura 55 - Tela de criação de um componente JETMIXER – Volume – coordenada Z



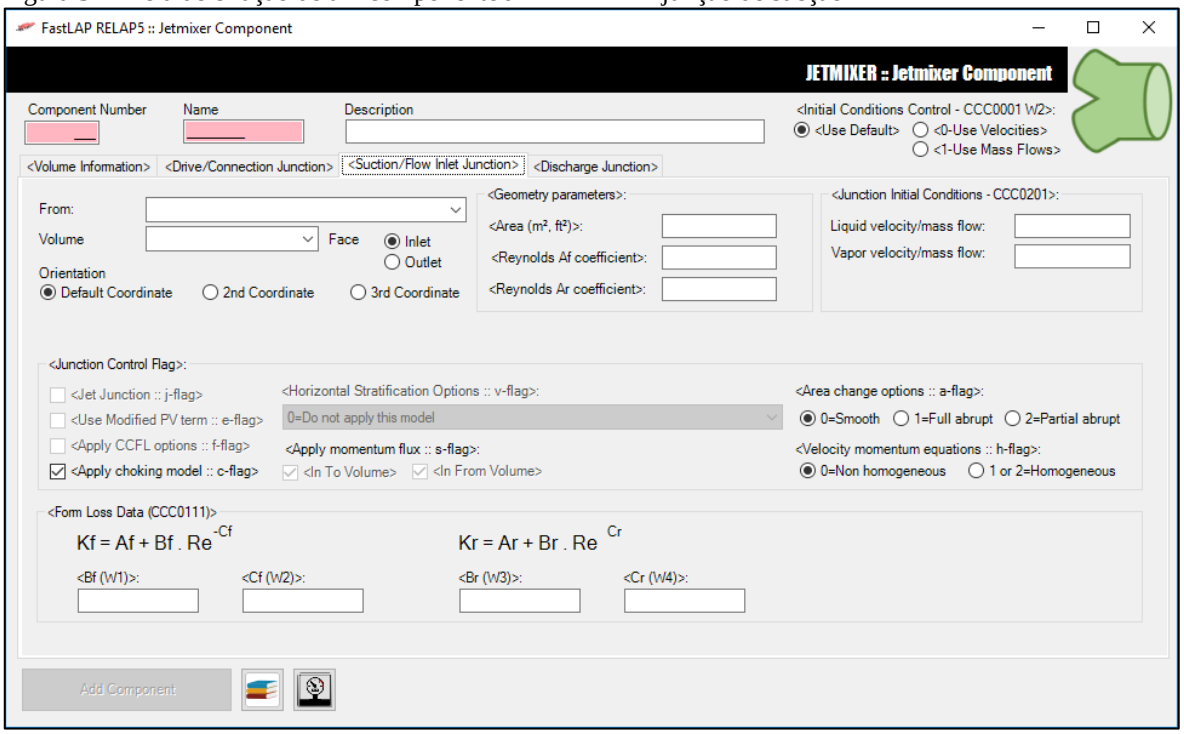
Fonte: autor da dissertação.

Figura 56 - Tela de criação de um componente JETMIXER – junção *drive*



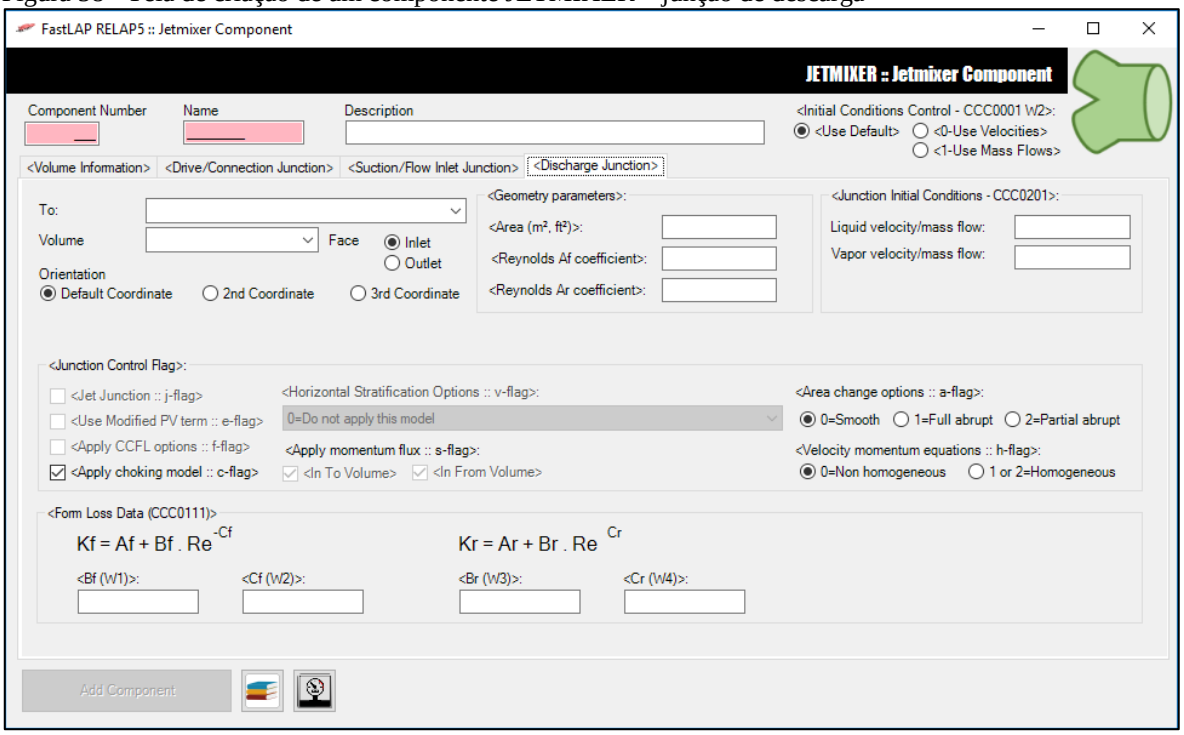
Fonte: autor da dissertação.

Figura 57 - Tela de criação de um componente JETMIXER – junção de sucção



Fonte: autor da dissertação.

Figura 58 - Tela de criação de um componente JETMIXER – junção de descarga



Fonte: autor da dissertação.

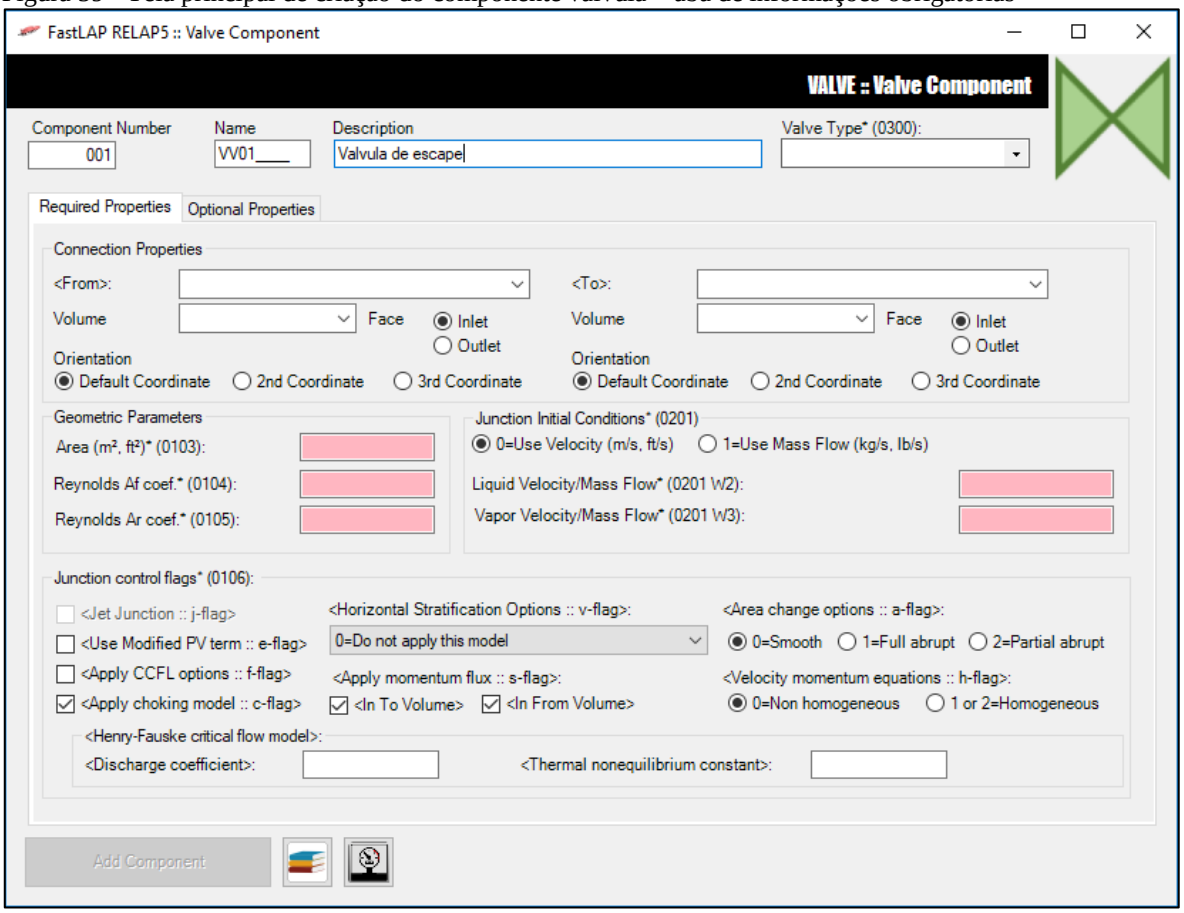
Repare nas Figuras 56 a 58 que, apesar dessas telas terem sido modeladas como junções, apenas uma conexão é solicitada para essas junções, e não duas (origem e destino) como de costume para um componente do tipo junção. O que ocorre é que para modelar

apropriadamente um JETMIXER, as junções *drive* e de sucção precisam ter seu destino conectados à entrada (*inlet end*) do JETMIXER; já a junção de descarga precisa obrigatoriamente ter sua origem na saída (*outlet end*) do JETMIXER que está sendo criado. O pré-processador, portanto, sequer irá solicitar essa informação ao usuário, pois ele já possui essa informação internamente, gerando os cartões de entrada e já levando em consideração essa particularidade, e solicitando aos usuários que informem somente os volumes externos que estarão conectados ao JETMIXER.

3.2.5.11 Válvula (VALVE)

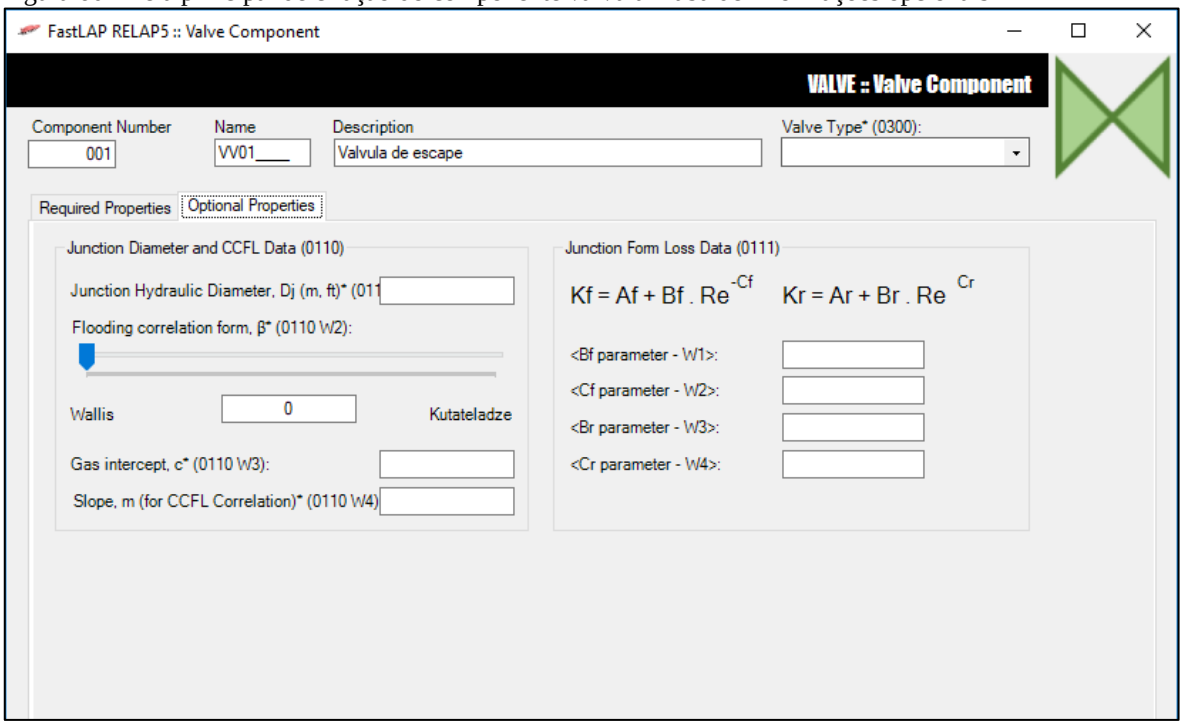
O componente do tipo válvula é usado para controlar a vazão de fluido entre os componentes hidrodinâmicos dentro do problema. Dessa forma, muitas das propriedades de uma válvula são semelhantes às das junções. Sendo assim, o pré-processador implementa essa tela com leiaute muito similar ao usado na tela de criação do componente SNGLJUN. Nas Figuras 59 e 60 são apresentadas as telas desse tipo de componente.

Figura 59 - Tela principal de criação do componente válvula – aba de informações obrigatórias



Fonte: autor da dissertação.

Figura 60 - Tela principal de criação do componente válvula – aba de informações opcionais

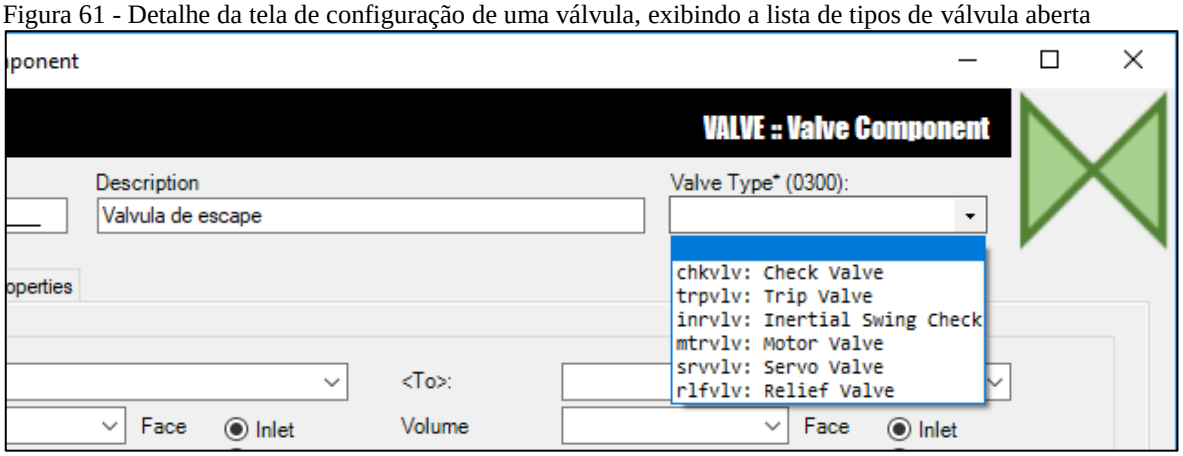


Fonte: autor da dissertação.

O código RELAP5 permite representar seis tipos diferentes de válvulas, cada uma com suas características peculiares. São elas: válvula *Check* (CHKVLV), válvula *Trip* (TRPVLV), válvula de checagem de balanço inercial (INRVLV), válvula motor (MTRVLV), válvula *servo* (SRVVLV) e válvula de alívio (RLFVLV). Todas essas válvulas foram contempladas no pré-processador, como mostram as Figuras 61 a 67.

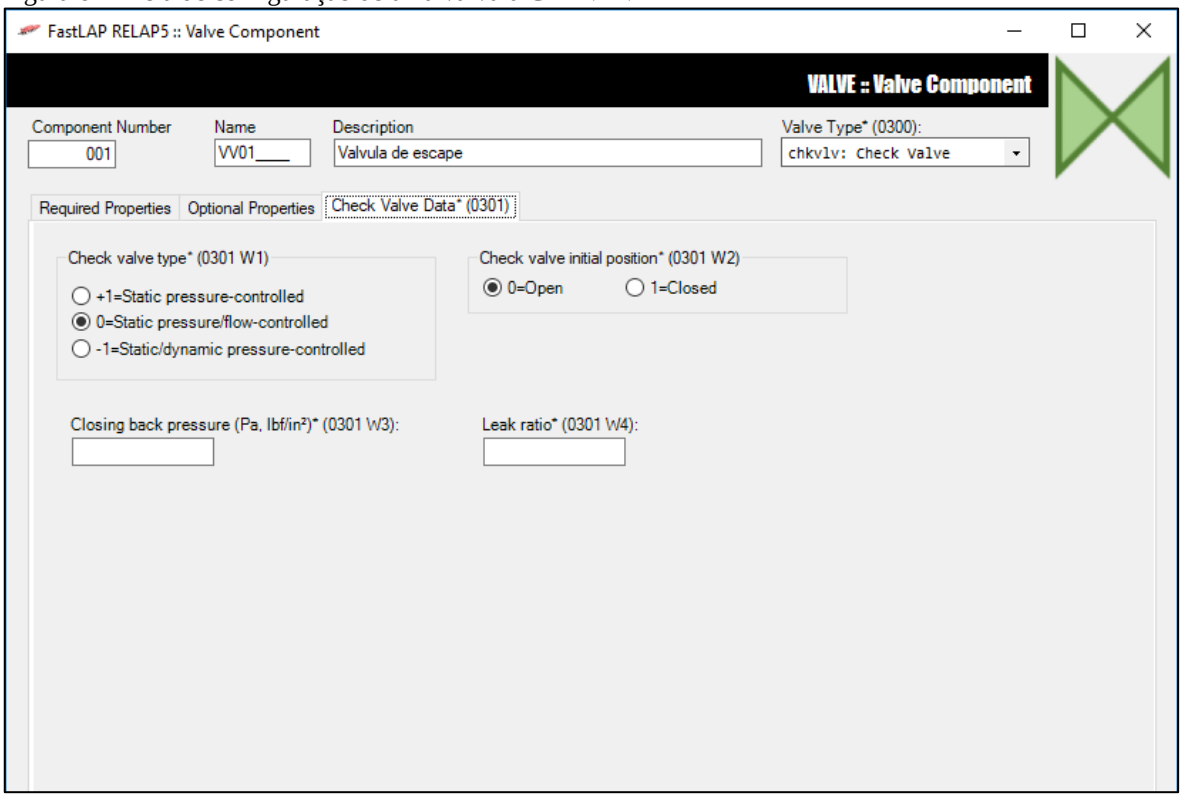
Ao abrir a janela do componente válvula, o usuário não verá nenhuma aba para configurar nenhuma dessas válvulas: essas informações só aparecerão depois que o usuário selecionar o tipo de válvula na lista correspondente, mantendo a tela com um visual limpo e evitando exibição de abas desnecessárias.

Figura 61 - Detalhe da tela de configuração de uma válvula, exibindo a lista de tipos de válvula aberta



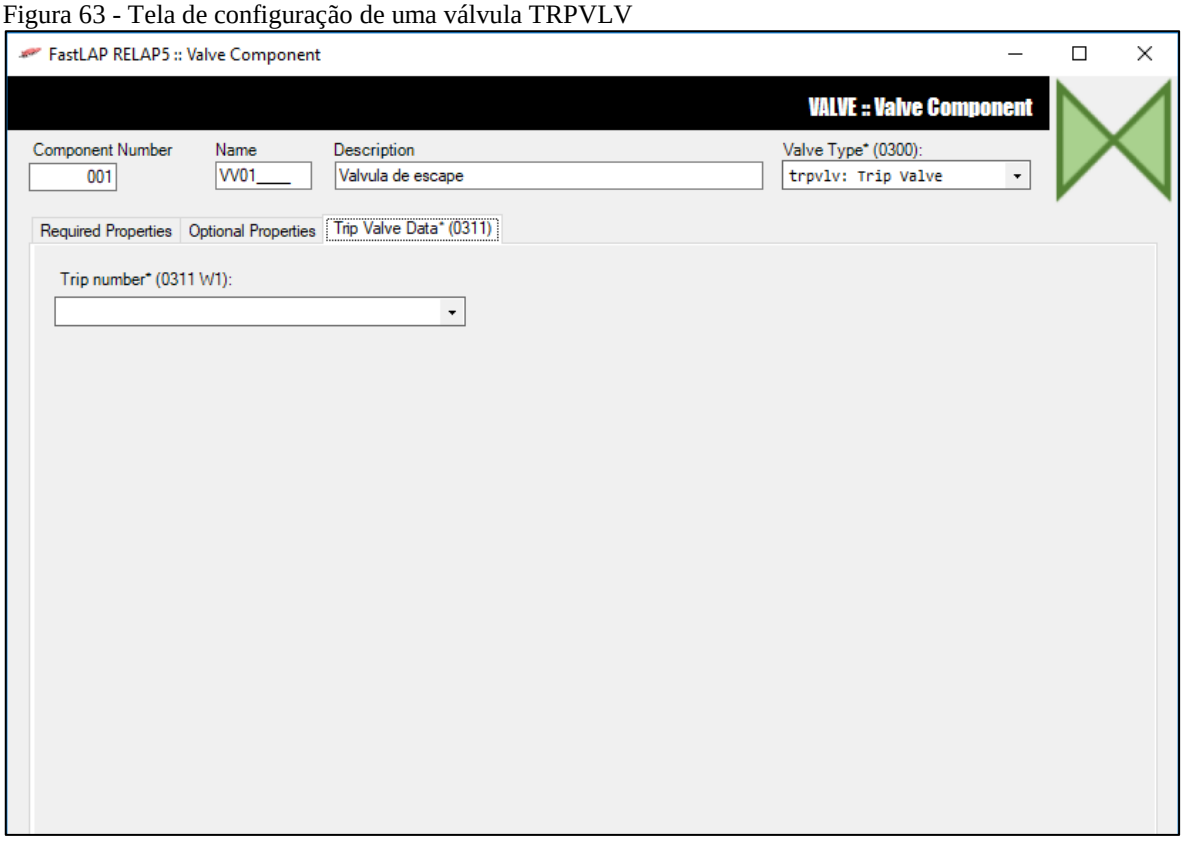
Fonte: autor da dissertação.

Figura 62 - Tela de configuração de uma válvula CHKVLV



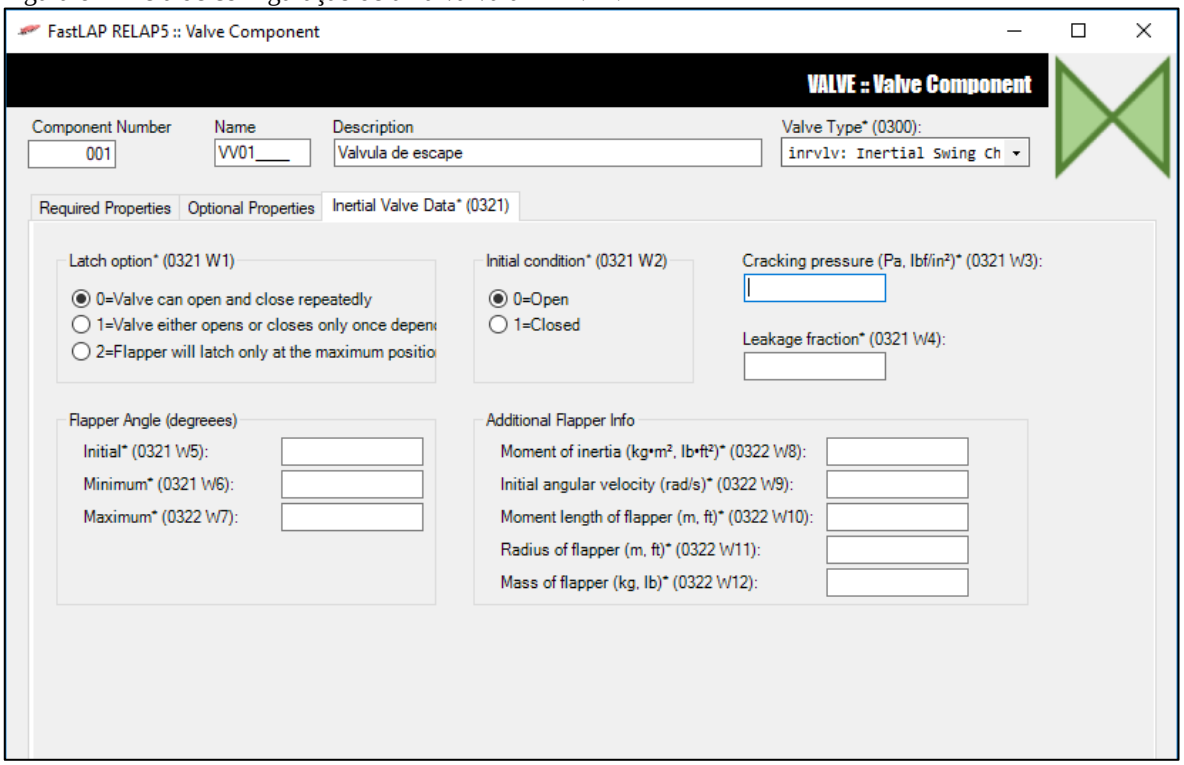
Fonte: autor da dissertação.

Figura 63 - Tela de configuração de uma válvula TRPVLV



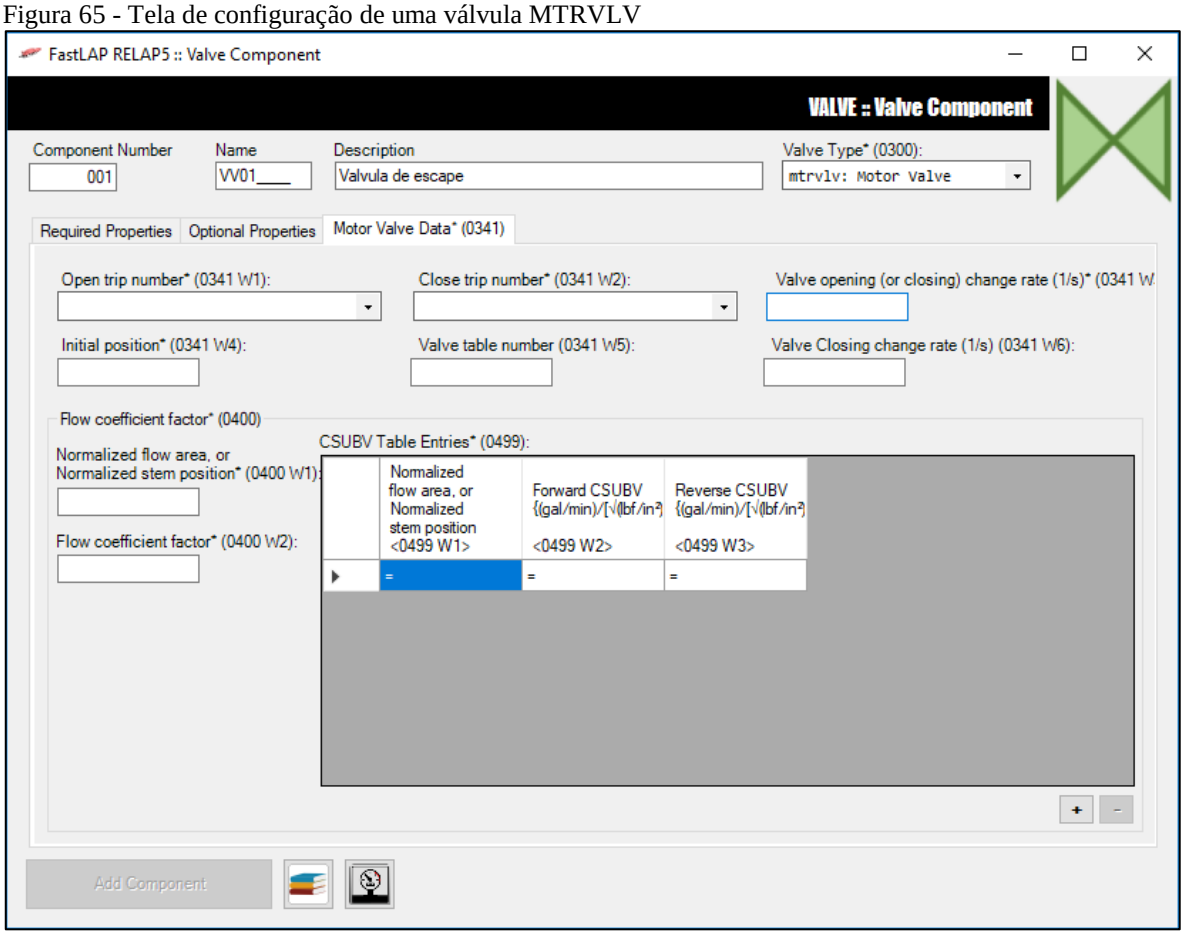
Fonte: autor da dissertação.

Figura 64 - Tela de configuração de uma válvula INRVLV



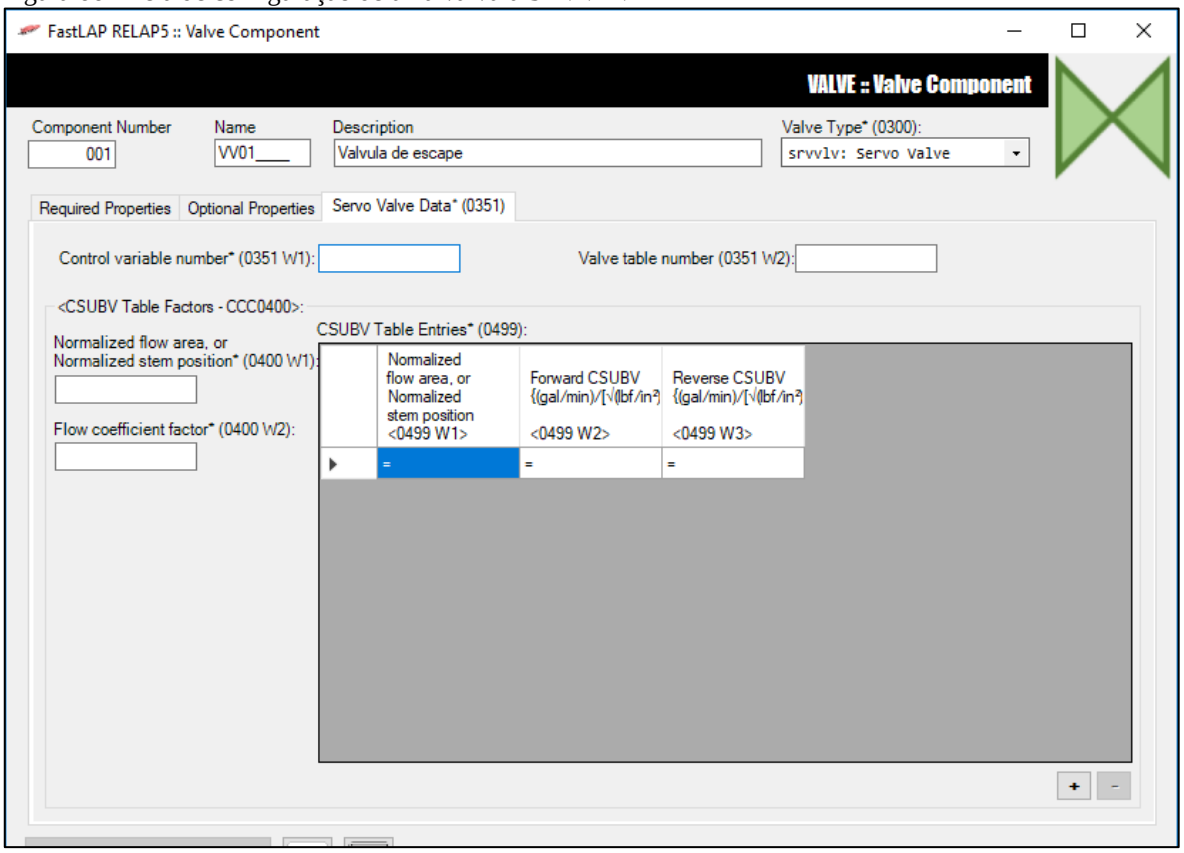
Fonte: autor da dissertação.

Figura 65 - Tela de configuração de uma válvula MTRVLV



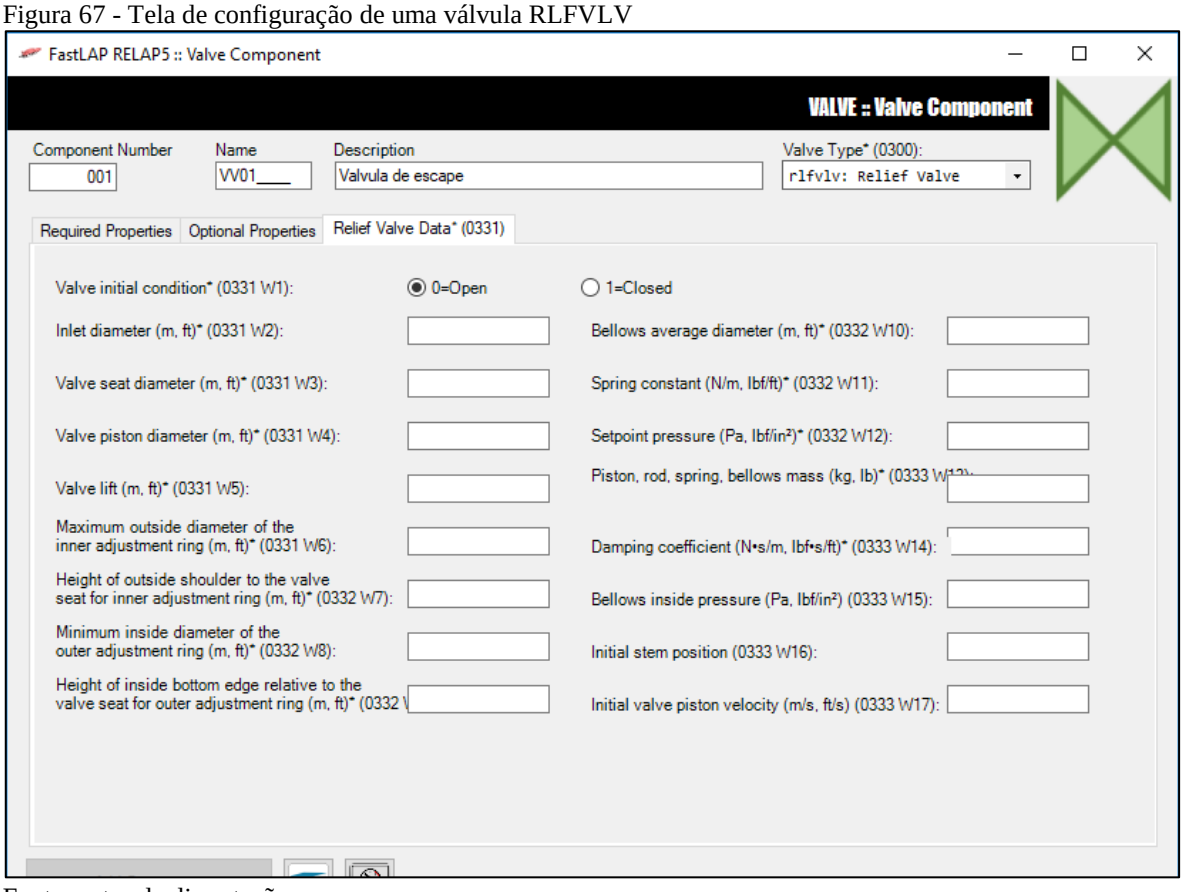
Fonte: autor da dissertação.

Figura 66 - Tela de configuração de uma válvula SRVVLV



Fonte: autor da dissertação.

Figura 67 - Tela de configuração de uma válvula RLFVLV



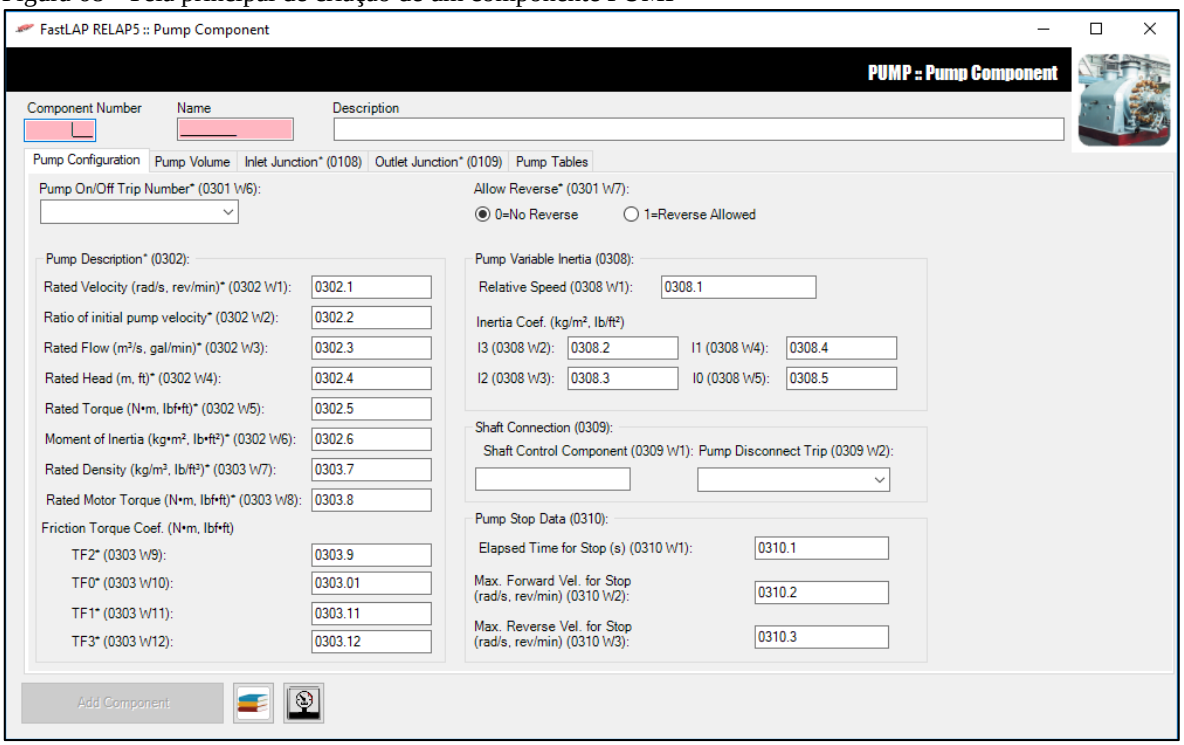
Fonte: autor da dissertação.

3.2.5.12 Bomba hidráulica (PUMP)

O componente do tipo bomba hidráulica, ou PUMP, é um componente bastante complexo. Via de regra, trata-se de um componente que gera uma carga adicional de trabalho no transporte de fluido entre outros componentes hidrodinâmicos. Como uma bomba possui um volume interno, e por ligar dois componentes hidrodinâmicos, a tela de criação de um PUMP possui tanto informações de configuração do volume quanto configurações das junções de entrada e de saída.

Para tornar a entrada de dados mais coesa, o pré-processador organizou as informações de uma bomba numa ordem ligeiramente diferente dos cartões de entrada do RELAP5. A primeira tela que o usuário terá que preencher ao criar uma bomba são informações obrigatórias e opcionais pertinentes à bomba em si, conforme exibido na Figura 68. No caso das bombas, o primeiro conjunto de cartões de dados da bomba chama-se CCC0101-0107. Já o primeiro conjunto de cartões de entrada que se referem exclusivamente a uma bomba é o cartão CCC0301; todavia, nesse trabalho optou-se por manter na primeira tela de criação de uma bomba as informações relacionadas exclusivamente a uma bomba, e mantendo os dados do volume e das junções da bomba em abas secundárias.

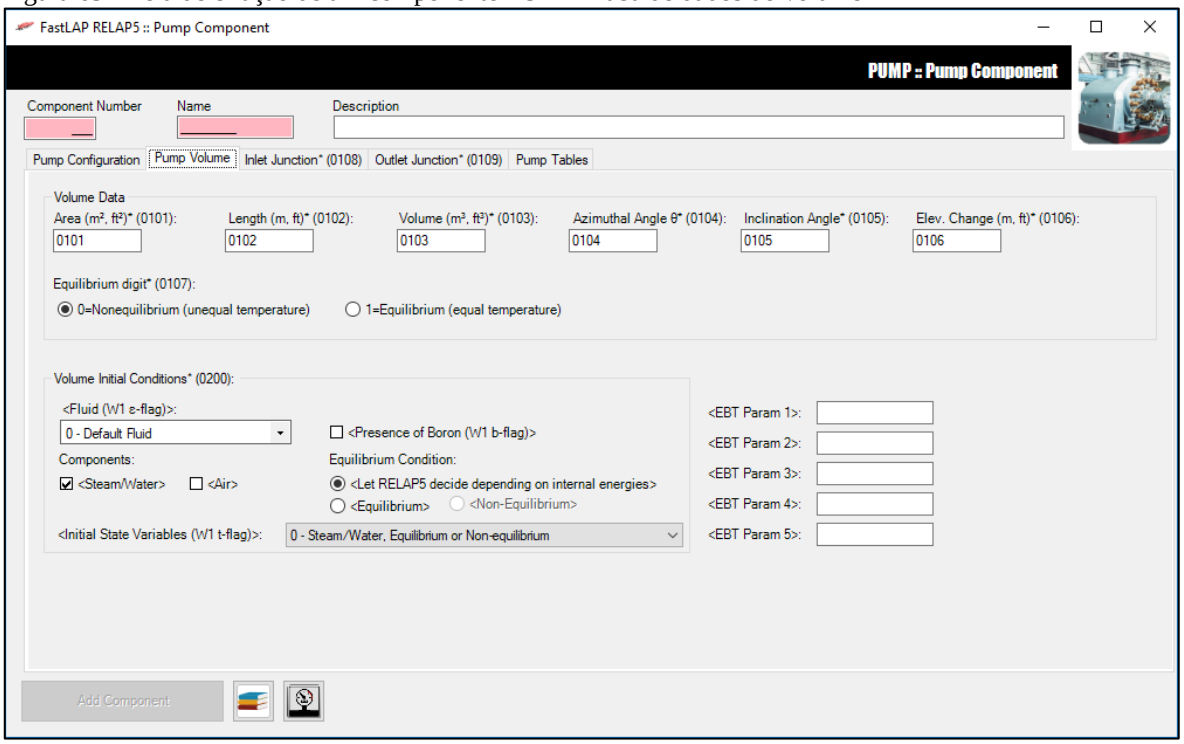
Figura 68 - Tela principal de criação de um componente PUMP



Fonte: autor da dissertação.

A Figura 69 mostra a aba seguinte, que permite ao usuário informar os dados da bomba que está sendo configurada.

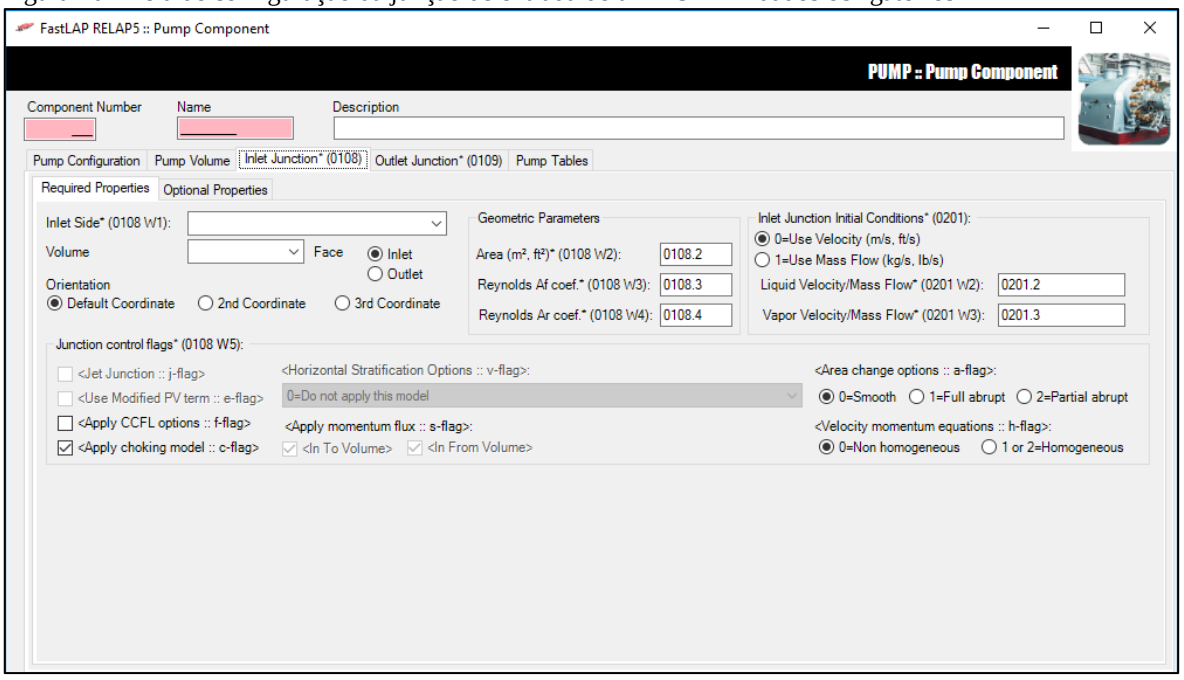
Figura 69 - Tela de criação de um componente PUMP – aba de dados do volume



Fonte: autor da dissertação.

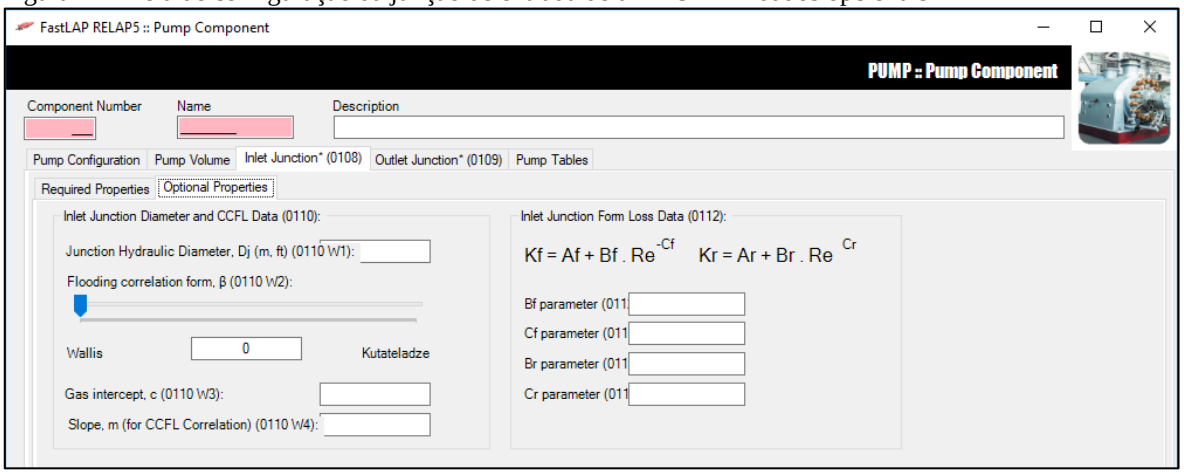
Uma bomba possui duas junções: uma de entrada na bomba, e uma de saída. A configuração dessas junções é muito similar à configuração de uma SNGLJUN. Desta forma, o pré-processador implementou as propriedades de entrada e saída da bomba usando interface muito semelhante à citada, como pode ser notado nas Figuras 70 a 73.

Figura 70 - Tela de configuração da junção de entrada de um PUMP – dados obrigatórios



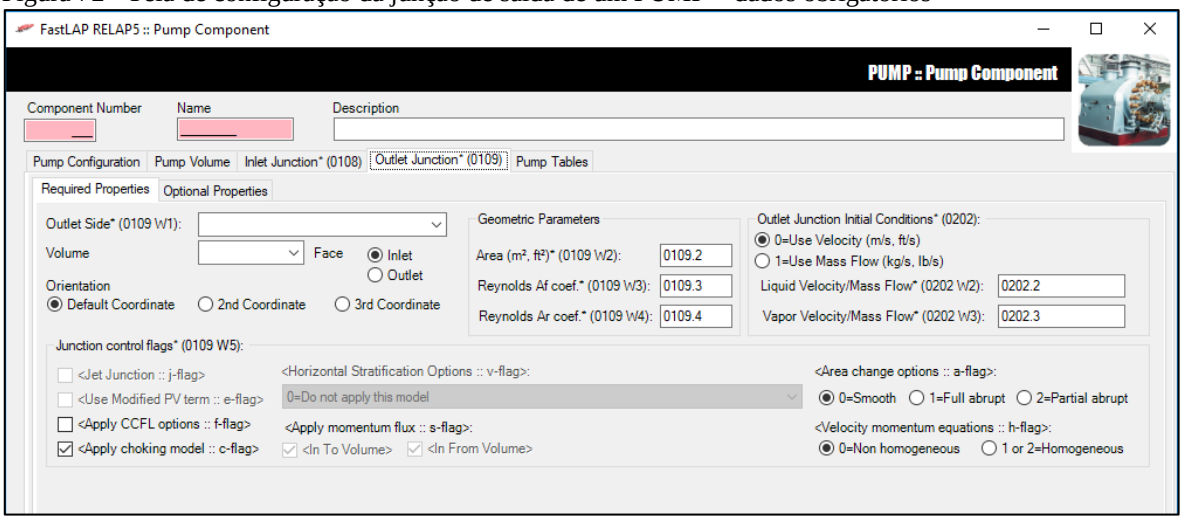
Fonte: autor da dissertação.

Figura 71 - Tela de configuração da junção de entrada de um PUMP – dados opcionais



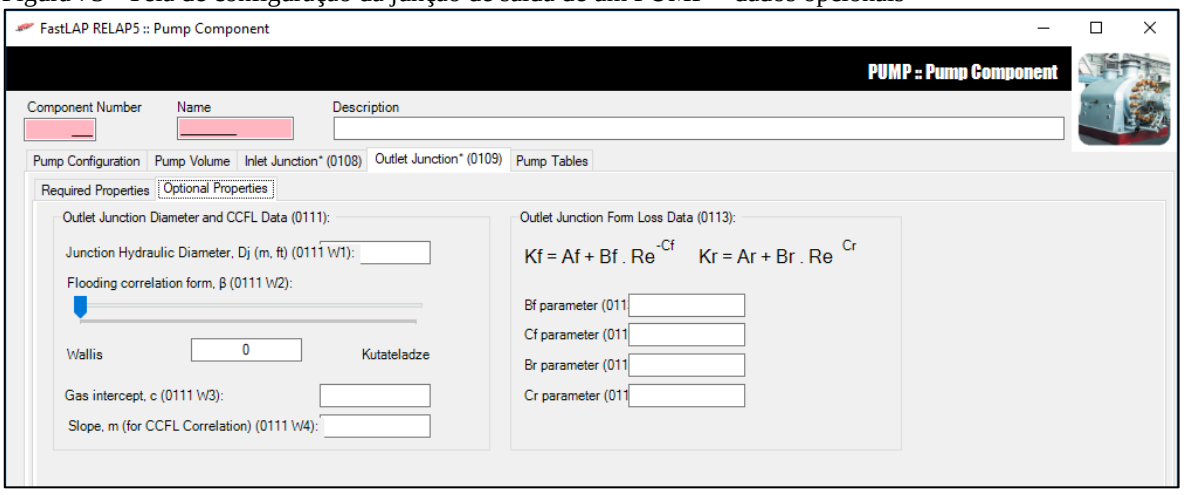
Fonte: autor da dissertação.

Figura 72 - Tela de configuração da junção de saída de um PUMP – dados obrigatórios



Fonte: autor da dissertação.

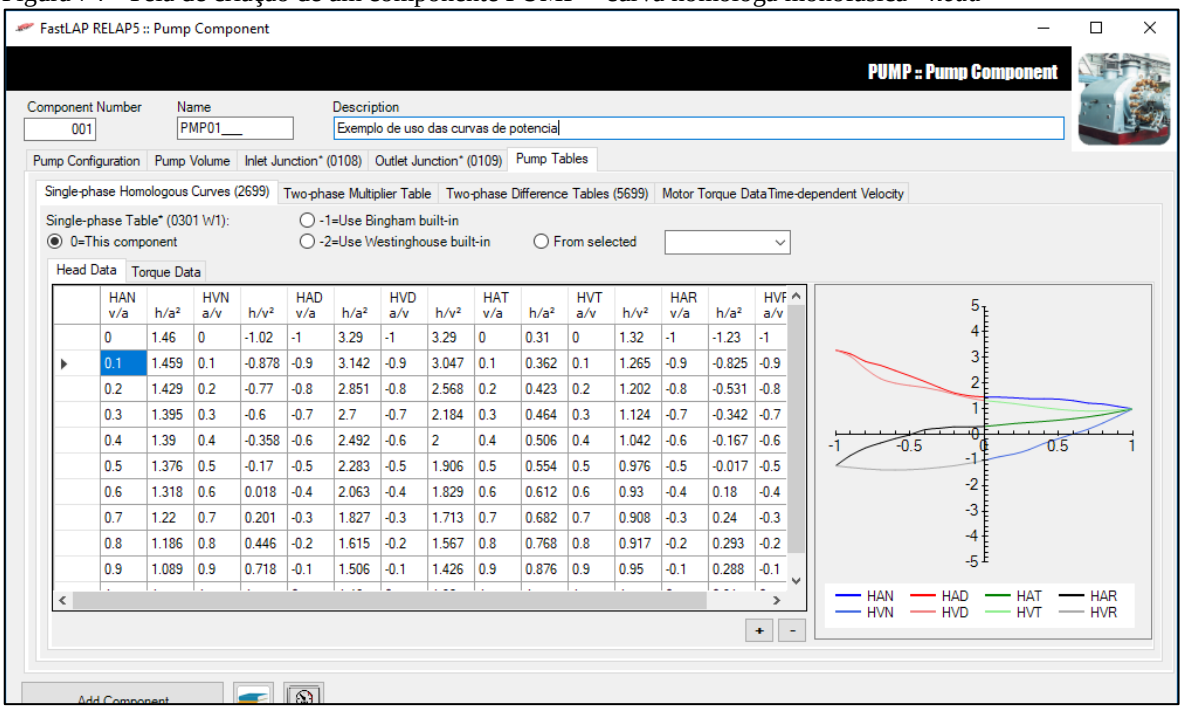
Figura 73 - Tela de configuração da junção de saída de um PUMP – dados opcionais



Fonte: autor da dissertação.

Além de possuir uma série de dados específicos de uma bomba, o RELAP5 oferece um conjunto de cartões de entrada do tipo tabela para informação das curvas homólogas da bomba. Entretanto, a entrada dessas informações é feita cada uma em seus cartões específicos, podendo um arquivo de entrada de dados chegar a mais de três mil cartões de entrada só para descrever adequadamente uma única bomba. Foi pensando em reduzir essa complexidade para o usuário que o pré-processador subdividiu as características das bombas em *grids* específicas de acordo com as curvas características, porém mantendo numa única *grid* as curvas que em geral são informadas juntas. Além disso, tão logo o usuário comece a digitar uma determinada curva homóloga, um pequeno gráfico vai sendo desenhado em paralelo, ajudando o usuário a encontrar mais rapidamente erros na digitação dos valores inseridos. O gráfico, conforme mostrado na Figura 74, está presente em quatro situações dentro desta tela de criação de bombas hidráulicas: para configuração das curvas homólogas monofásica de *head* e de torque, e para as curvas homólogas bifásica, também para *head* e torque.

Figura 74 - Tela de criação de um componente PUMP – curva homóloga monofásica - *head*



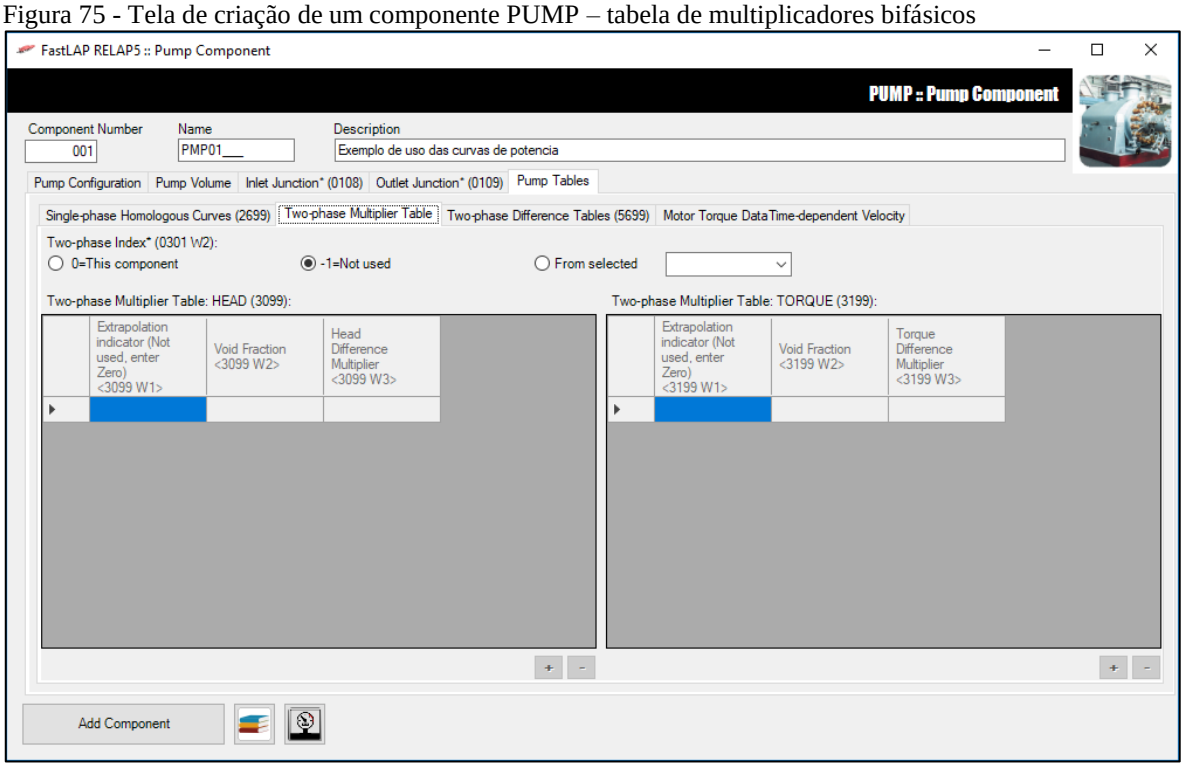
Fonte: autor da dissertação.

Mais informações sobre as curvas de potência e curvas homólogas podem ser vistas no APÊNDICE I.

As demais informações da bomba inseridas para o código RELAP5 no formato de cartões de tabelas são alimentadas pelos usuários usando controle de *grid*, com o qual o

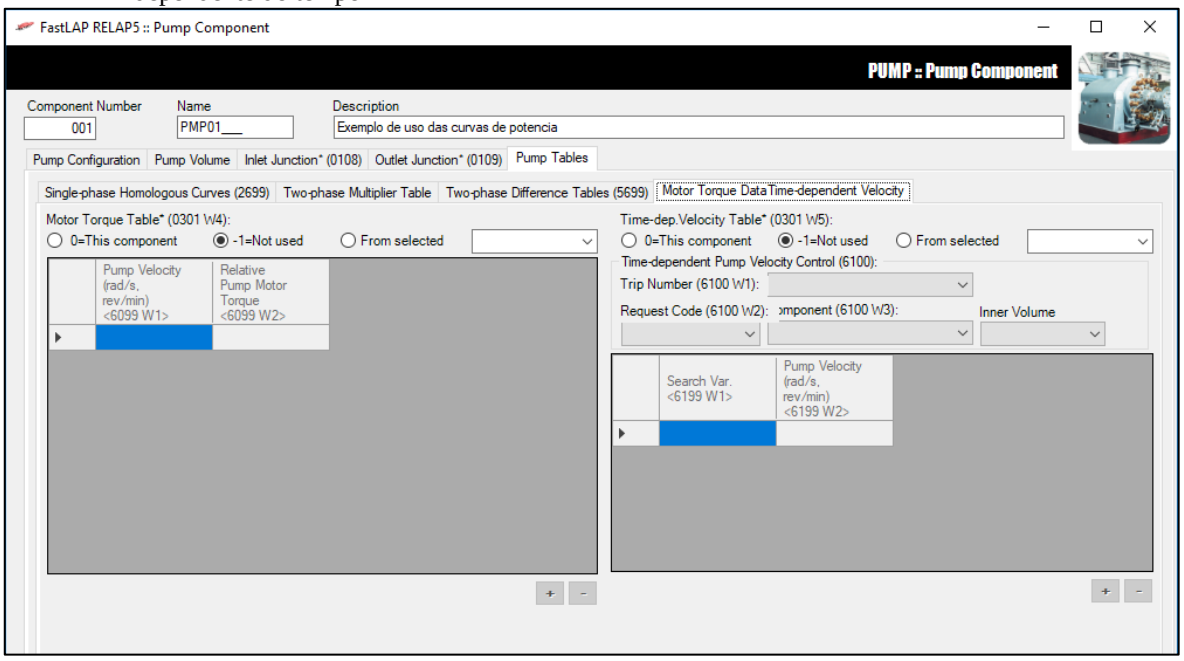
usuário a essa altura já deve ter se familiarizado. Abaixo, vemos a tela para informação das tabelas de multiplicadores para bombas em regime bifásico (Figura 75), e para alimentação dos dados referentes ao torque relativo do motor e do controle de velocidade dependente do tempo (Figura 76).

Figura 75 - Tela de criação de um componente PUMP – tabela de multiplicadores bifásicos



Fonte: autor da dissertação.

Figura 76 - Tela de criação de um componente PUMP – dados do torque do motor e controle de velocidade dependente do tempo



Fonte: autor da dissertação.

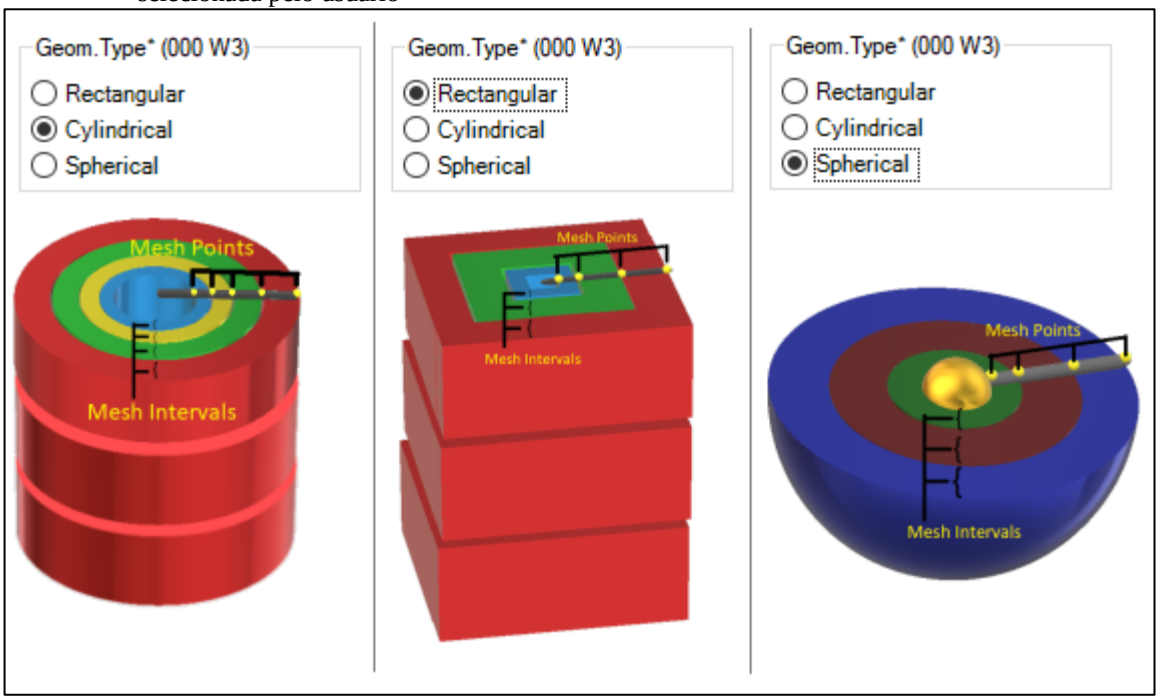
3.2.5.13 Estrutura de troca de calor (HEAT TRANSFER)

É por meio do componente de troca de calor, ou HEAT TRANSFER, que o RELAP5 permite ao usuário modelar fisicamente as estruturas que envolvem a todos os elementos hidrodinâmicos dos problemas modelados no RELAP5. É aqui que serão definidos volumes de controle para monitorar a troca de calor entre os diferentes elementos do problema, bem como com o meio ambiente.

Por padrão, o RELAP5 trabalha com três geometrias básicas para modelagem de elementos HEAT TRANSFER: geometria cilíndrica, retangular e esférica.

O RELAP5 subdivide os volumes de troca de calor em basicamente dois eixos: o eixo radial, cujos intervalos são chamados de *mesh intervals*, sendo cada um desses intervalos separados por *mesh points*; e o eixo axial, cujos volumes de controle são chamados simplesmente de *Heat Structures*. Para ajudar o usuário a lembrar-se dessa definição, o pré-processador mostra uma tela com um pequeno esquema contendo essas definições de acordo com a geometria escolhida, como pode ser visto na Figura 77.

Figura 77 - Diagramas ilustrativos dos volumes de controle das estruturas de calor, conforme a geometria selecionada pelo usuário



Fonte: autor da dissertação.

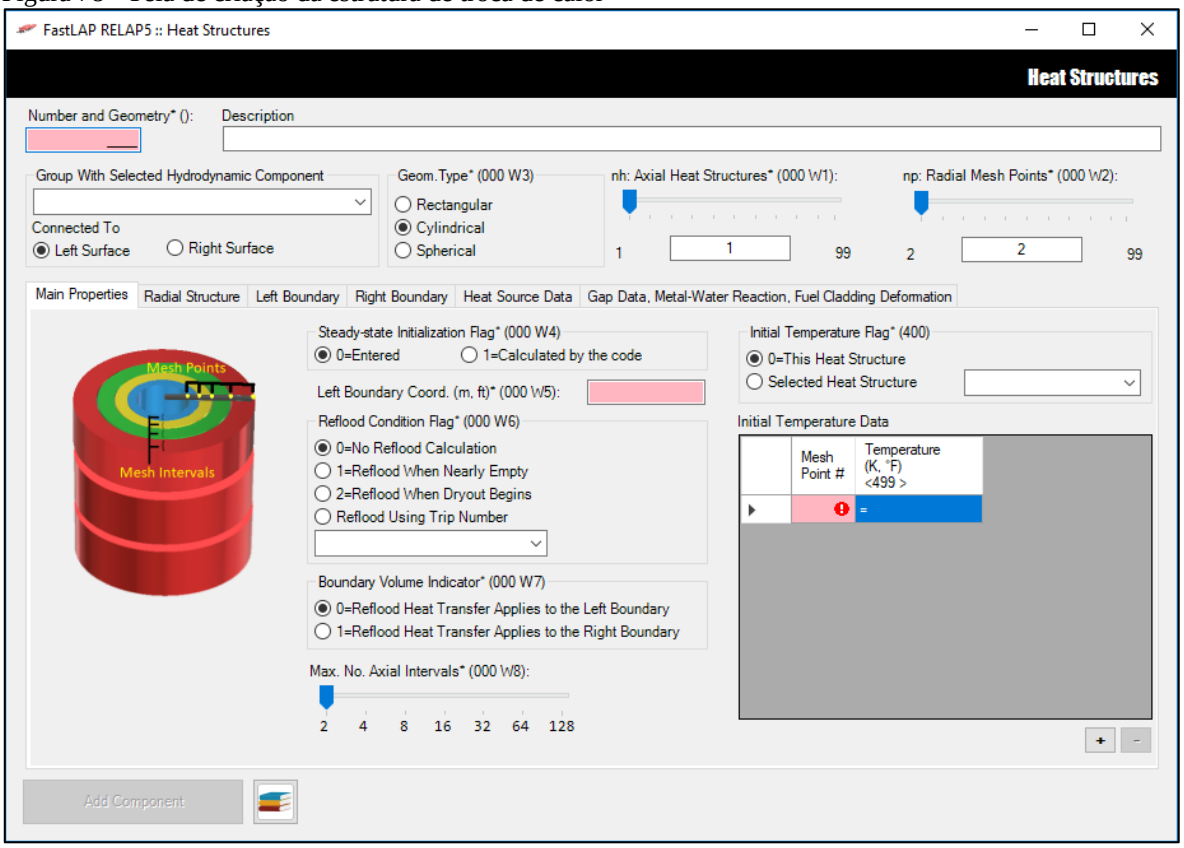
Devido a isso, a organização das informações da estrutura de troca de calor foi separada de forma a tentar manter próximas as estruturas axiais e radiais, evitando criar *grids* para qualquer dado. Nos casos em que o RELAP5 permite a expansão de cartões, como

explicado no item 3.2.5.5, a expansão dentro do pré-processador funcionará da mesma forma, utilizando-se do símbolo de igual (“=”).

A estrutura de troca de calor possui uma particularidade de estar sempre conectada a um ou dois elementos hidrodinâmicos. Em geral, essas estruturas são representadas nos diagramas de nodalização como uma caixa com linhas hachuradas encostada no elemento hidrodinâmico principal a ele ligado.

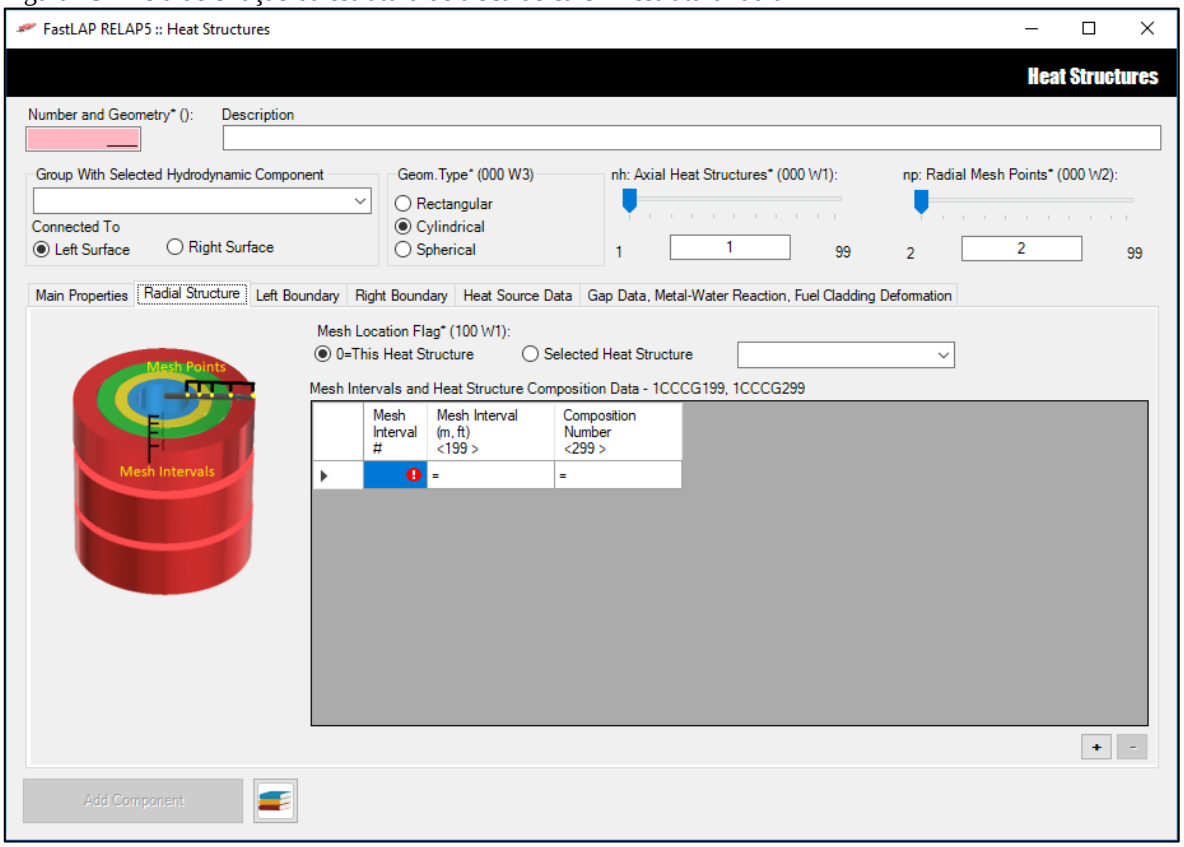
Uma visão geral da criação de uma estrutura de calor pode ser vista por meio das Figuras 78 a 82.

Figura 78 - Tela de criação da estrutura de troca de calor



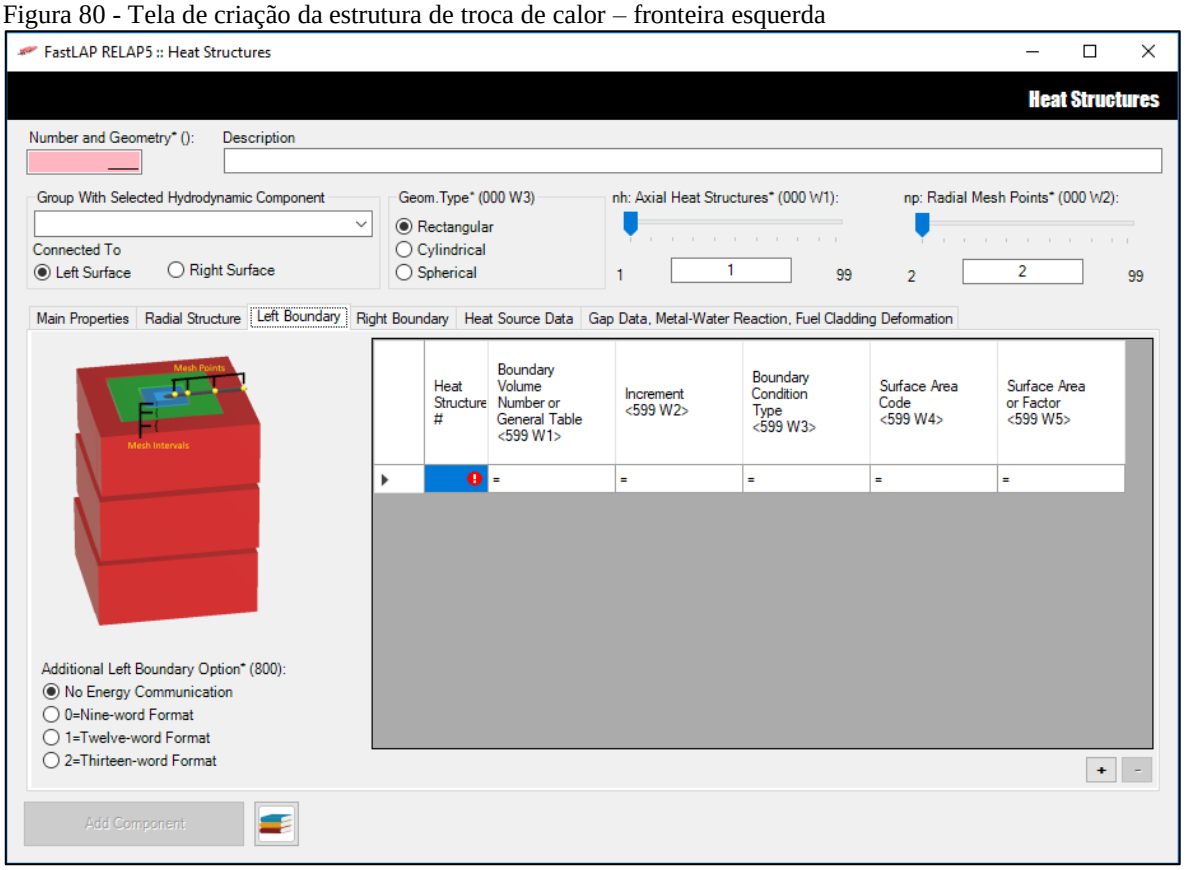
Fonte: autor da dissertação.

Figura 79 - Tela de criação da estrutura de troca de calor – estrutura radial



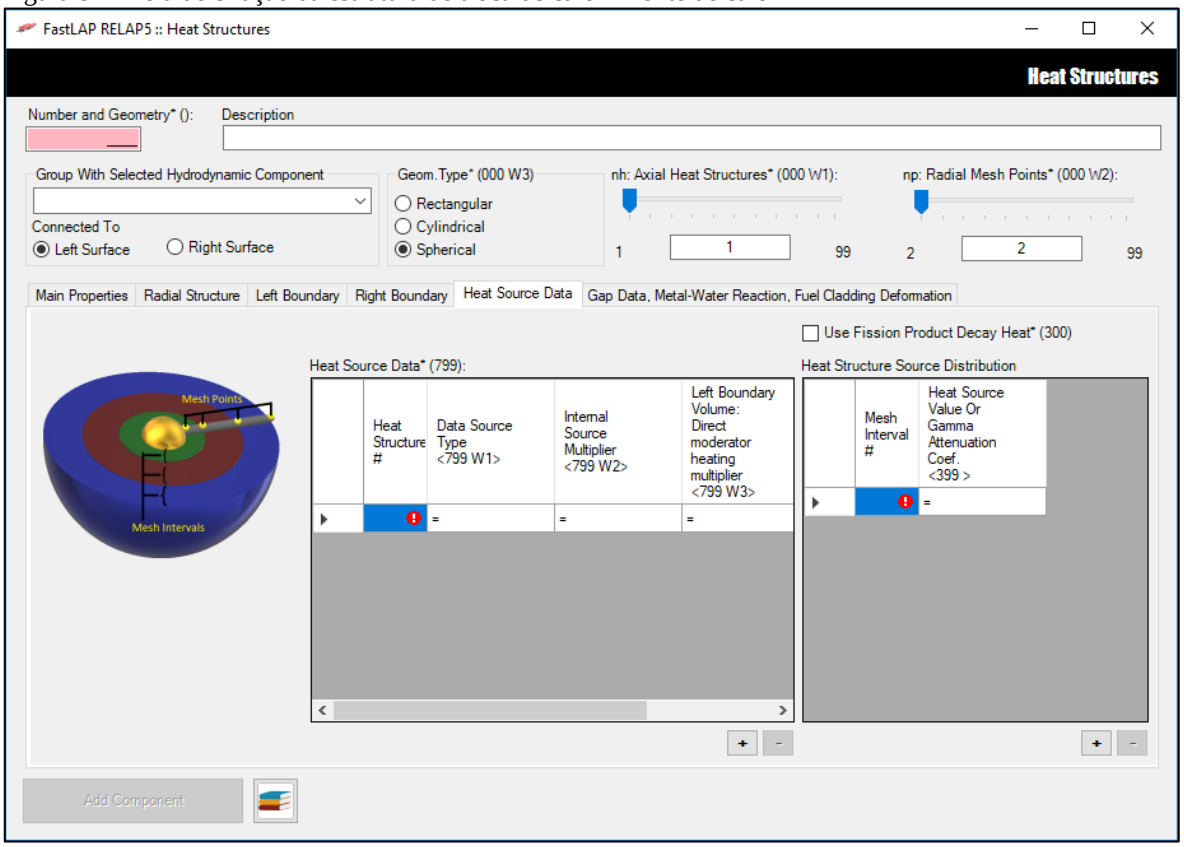
Fonte: autor da dissertação.

Figura 80 - Tela de criação da estrutura de troca de calor – fronteira esquerda



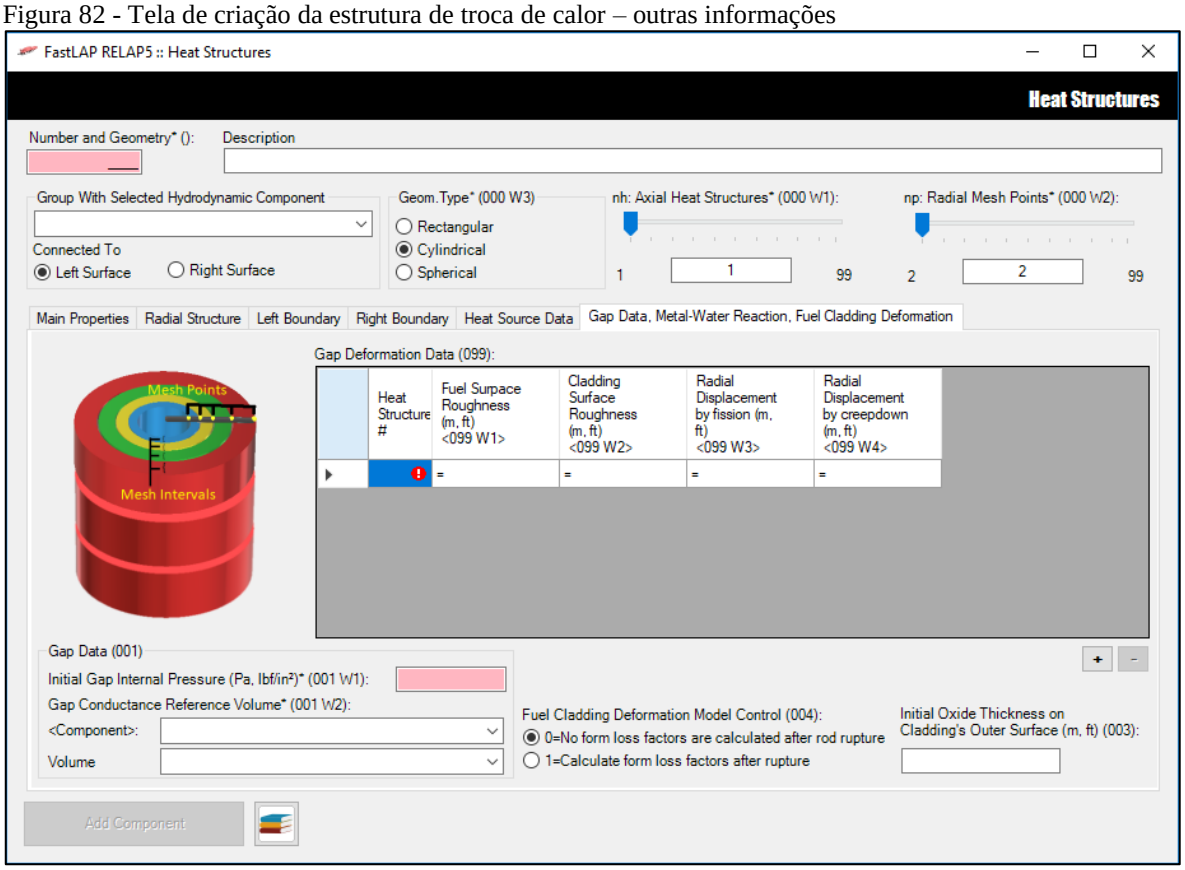
Fonte: autor da dissertação.

Figura 81 - Tela de criação da estrutura de troca de calor – fonte de calor



Fonte: autor da dissertação.

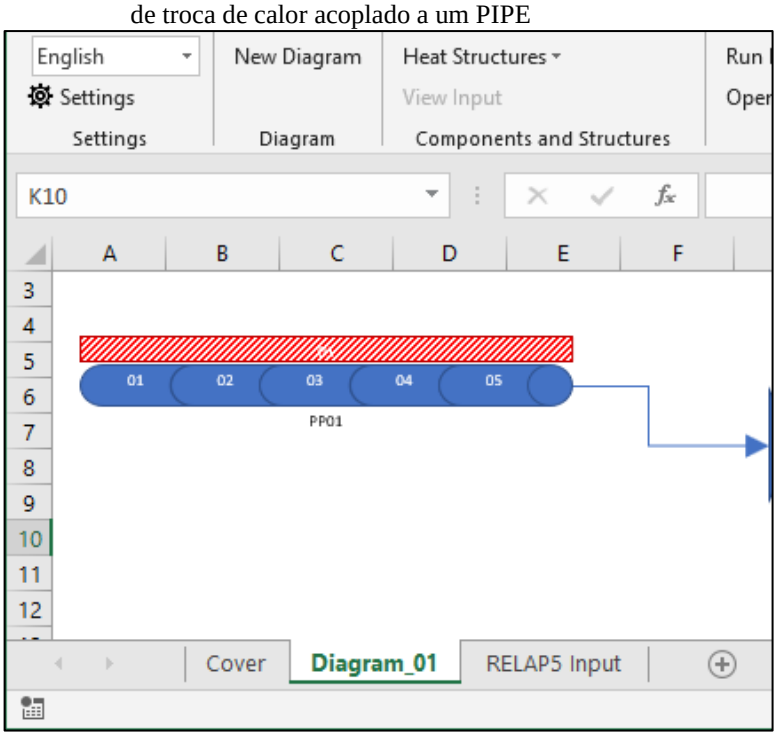
Figura 82 - Tela de criação da estrutura de troca de calor – outras informações



Fonte: autor da dissertação.

Uma vez que todos os campos obrigatórios são corrigidos, o botão para inserção do HEAT TRANSFER no problema é habilitado para uso e, uma vez clicado, a estrutura de troca de calor configurada é desenhada na tela de diagrama. Uma observação interessante é que a sua representação gráfica é inserida no diagrama como uma caixa vermelha hachurada acoplada ao componente hidrodinâmico principal ligado a ele. Dessa forma, ao mover o componente hidrodinâmico dentro do diagrama, o seu elemento de transferência de calor é movido juntamente com ele. Todavia, o elemento de troca de calor só pode ser representado graficamente acoplado a um único elemento hidrodinâmico por vez; ou seja, se o elemento troca calor entre dois PIPES, por exemplo (representando um casco interno), o usuário deverá escolher em qual dos dois PIPES o elemento de troca de calor deverá ser acoplado visualmente. Dependendo do problema, a subdivisão da estrutura de troca de calor seguirá o número de volumes de controle do componente conectado a ela. A Figura 83 mostra a representação visual de um elemento de troca de calor no diagrama, acoplado a um PIPE.

Figura 83 - Detalhe do diagrama de nodalização mostrando um elemento de troca de calor acoplado a um PIPE



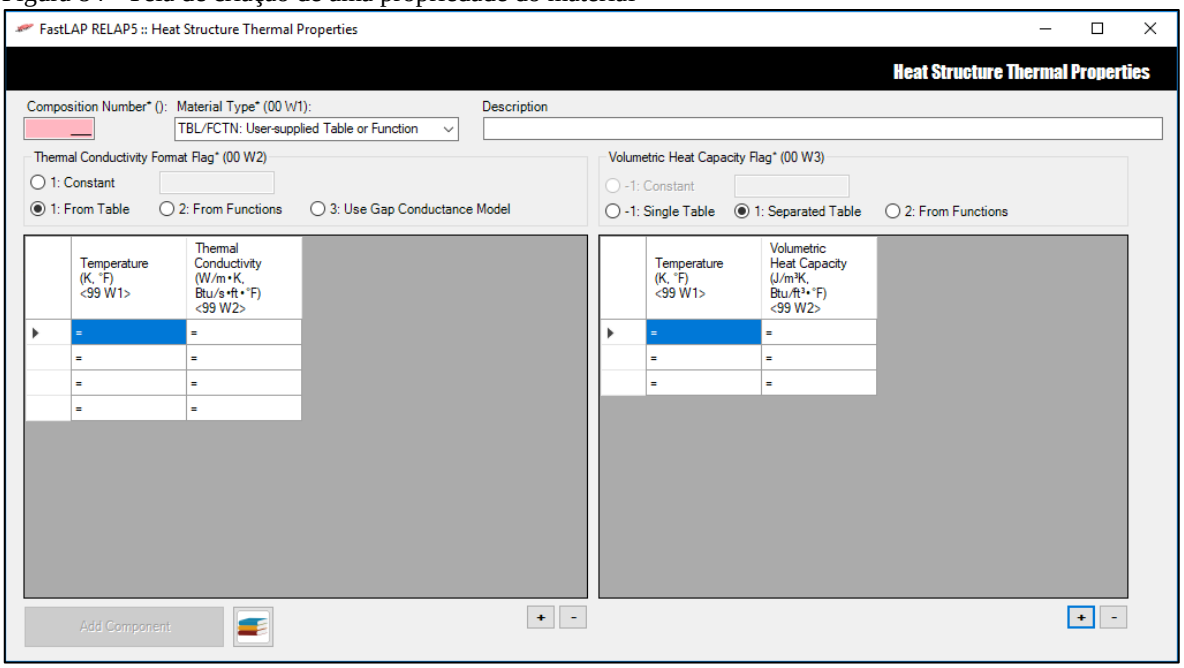
Fonte: autor da dissertação.

3.2.5.14 Propriedade dos materiais (HEAT STRUCTURE THERMAL PROPERTIES)

O componente HEAT TRANSFER, por si só, não é capaz de conter todas as informações necessárias para a correta configuração de um modelo de troca de calor. As propriedades físicas dos materiais usados nas estruturas, tais como tipo de material, condutividade térmica, capacidade de calor volumétrica, por exemplo, ficam configuradas

aqui neste componente, e quando se cria uma nova estrutura HEAT TRANSFER, faz-se a referência ao elemento de propriedade do material criado pela tela ilustrada na Figura 84.

Figura 84 - Tela de criação de uma propriedade do material



Fonte: autor da dissertação.

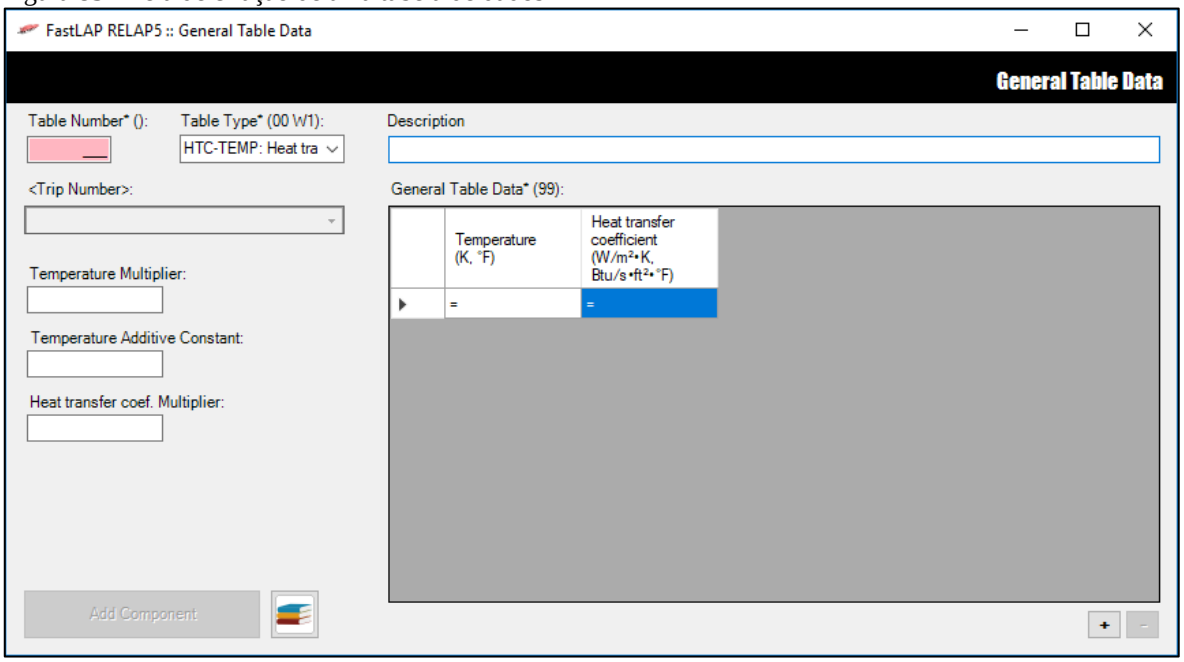
O código RELAP5 já possui alguns materiais definidos internamente: aço carbono (C-STEEL), aço inoxidável (S-STEEL), dióxido de urânio (UO_2) e zircônio (ZR). Caso o usuário opte por utilizar algum desses materiais, não é necessária a configuração das propriedades térmicas desses materiais. Caso o usuário opte por utilizar um material diferente dos enumerados acima, ele terá que informar sua condutividade térmica e sua capacidade de calor volumétrica. Ambas as propriedades podem ser informadas como uma constante, ou como uma função da temperatura. O pré-processador está apto a oferecer ao usuário a mesma flexibilidade do código RELAP5 na configuração dessas propriedades.

O elemento de propriedade dos materiais não possui representação visual no diagrama de nodalização.

3.2.5.15 Tabela de dados gerais (GENERAL TABLE DATA)

Para completar o conjunto de elementos de troca de calor, um dado bastante usado é a Tabela de dados gerais. É por meio dessa tabela que o usuário do RELAP5 consegue configurar fontes permanentes de calor, definir condições de contorno, ou criar funções de calor dependentes do tempo ou da temperatura. A tela de criação de uma tabela pode ser vista na Figura 85.

Figura 85 - Tela de criação de uma tabela de dados



Fonte: autor da dissertação.

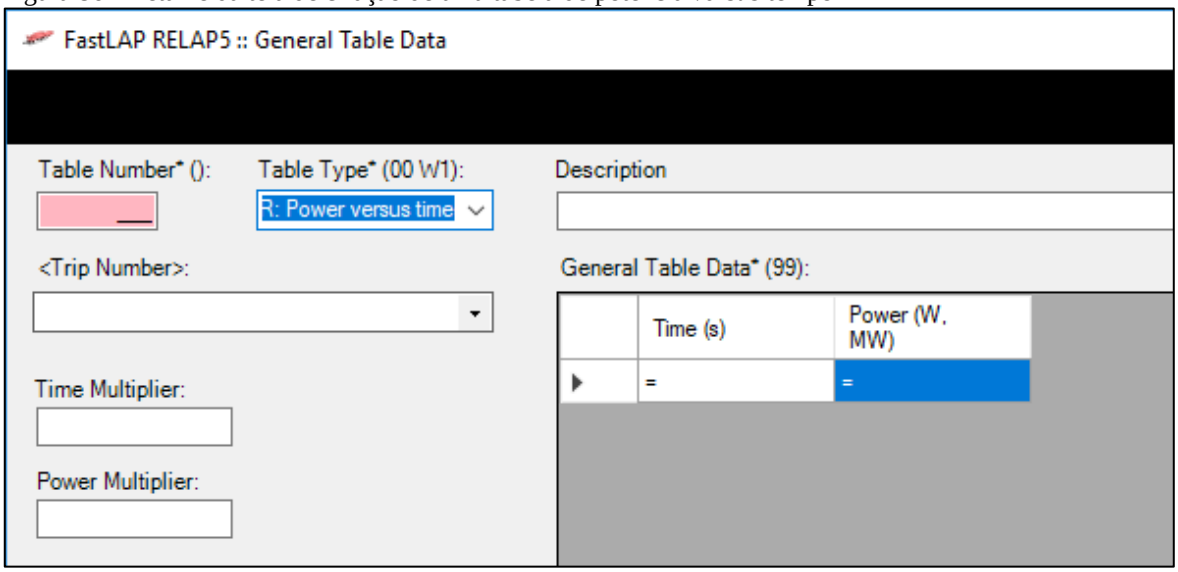
O código RELAP5 permite criar sete tipos diferentes de tabelas, a saber:

- Potência *versus* Tempo (POWER);
- Fluxo de calor *versus* Tempo (HTRNRATE);
- Coeficiente de transferência de calor *versus* Tempo (HTC-T);
- Coeficiente de transferência de calor *versus* Temperatura (HTC-TEMP);
- Temperatura *versus* Tempo (TEMP);
- Reatividade *versus* Tempo (REAC-T);
- Área normalizada *versus* Posição da vareta combustível (NORMAREA).

De acordo com o tipo de tabela escolhida, dois ou três parâmetros precisam ser informados. O pré-processador ajusta automaticamente o controle *grid* para que o usuário saiba exatamente que tipo de informação precisa ser inserida para cada tipo de tabela, como pode ser visto nas Figuras 86 e 87.

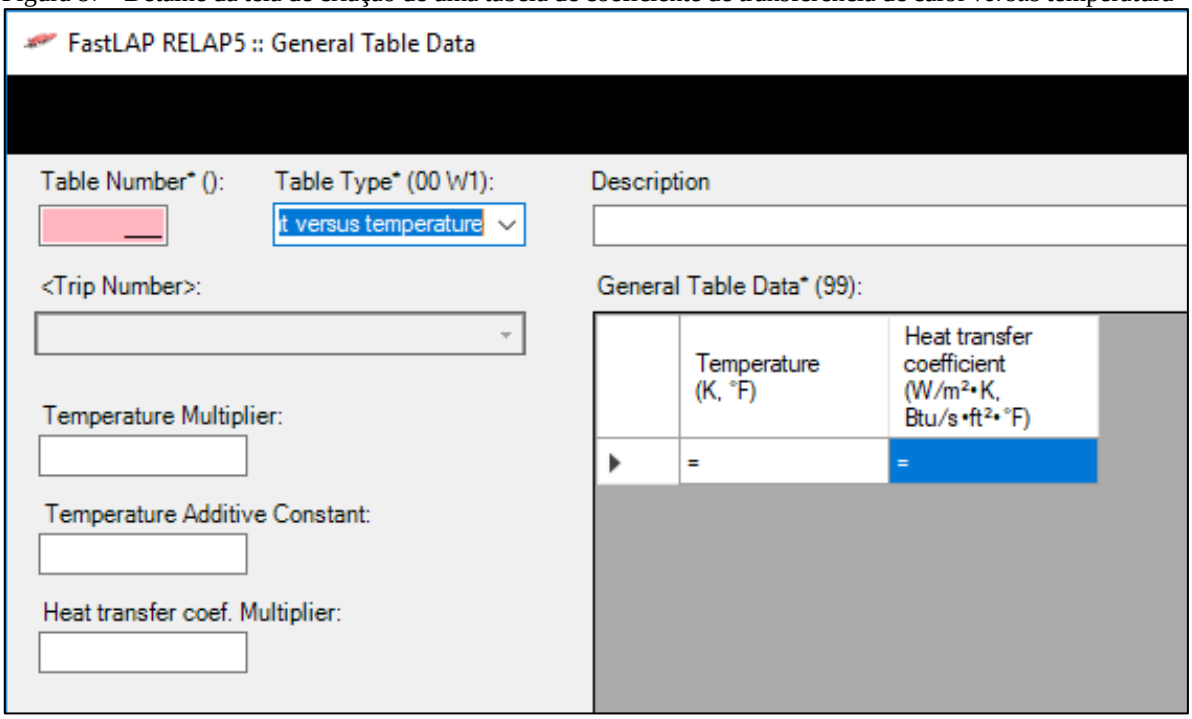
O elemento de tabela de dados gerais não possui representação visual no diagrama de nodalização.

Figura 86 - Detalhe da tela de criação de uma tabela de potência *versus* tempo



Fonte: autor da dissertação.

Figura 87 - Detalhe da tela de criação de uma tabela de coeficiente de transferência de calor *versus* temperatura



Fonte: autor da dissertação.

A seguir serão apresentadas as telas do pré-processador que foram utilizadas para gerar os dados de entrada da "Experiência SUPER CANON" para o código RELAP5 e os resultados da simulação de um dos cenários de acidente.

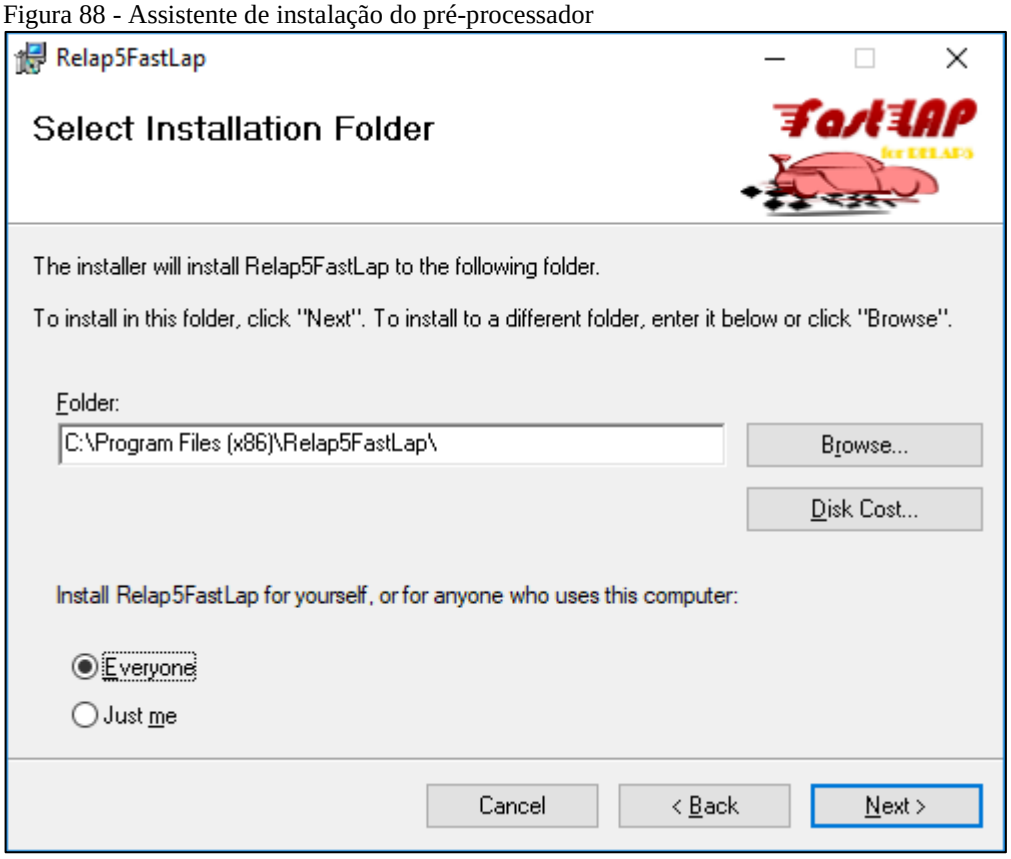
3.2.6 Instalação e distribuição do pré-processador

O pré-processador FastLAP consiste em um complemento para o *MS Excel*®, embutido dentro de um arquivo DLL do *Windows*®. A DLL que contém o pré-processador

pode ser facilmente copiada em um novo computador e registrada como complemento dentro do *MS Excel*®, por meio da tela de configurações nele disponível.

Para facilitar a instalação do pré-processador em novos computadores, um assistente de instalação padrão para o ambiente *Windows*® foi criado. Por meio dele, como mostrado na Figura 88, será feita a criação de uma pasta de instalação, o registro da DLL como complemento dentro do *MS Excel*® e a cópia dos demais arquivos necessários para uso do pré-processador (como o arquivo de traduções inicial, por exemplo). Além disso, o módulo de instalação também verifica se a versão do *.NET Framework*® necessária para uso está instalada no computador do usuário; caso não esteja, o usuário é direcionado para fazer o *download* e instalação da versão adequada para o correto funcionamento do pré-processador.

Figura 88 - Assistente de instalação do pré-processador



Fonte: autor da dissertação.

O assistente de instalação consiste em dois arquivos:

- FastLapSetup.msi: o pacote de instalação no formato do *Windows Installer*® referente ao pré-processador FastLAP;

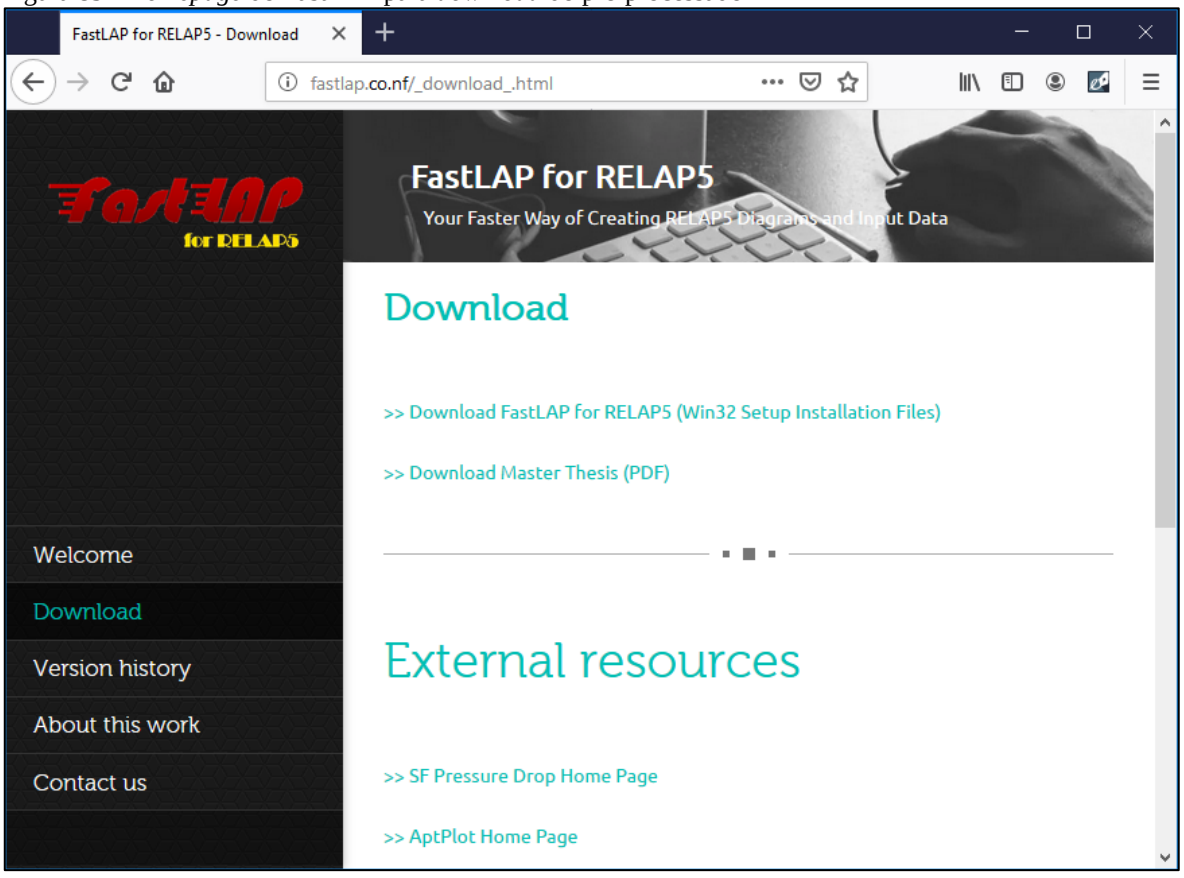
- Setup.exe: o arquivo inicial de instalação. Esse programa checa se os pré-requisitos necessários para que a instalação possa iniciar estão presentes; em caso positivo, o módulo MSI é iniciado automaticamente.

Ambos os arquivos estão disponíveis para *download* dentro de um único arquivo compactado, no formato ZIP, a partir do endereço de internet http://fastlap.co.nf/. Nesse endereço, também estão disponíveis:

- Informações gerais sobre o pré-processador;
- Endereço eletrônico de e-mail para dúvidas, erros e sugestões;
- Atalhos diversos para outros programas relacionados ao *FastLAP*;
- Atalho para *download* dessa dissertação.

A Figura 89 mostra a página criada para que os usuários possam baixar o instalador do pré-processador.

Figura 89 - *Homepage* do FastLAP para *download* do pré-processador



Fonte: autor da dissertação.

4 RESULTADOS E DISCUSSÕES

4.1 Programação dos protótipos

Nessa etapa foram desenvolvidos dois protótipos:

- o protótipo desenvolvido em VBA embutido dentro de documento *MS Excel*® e com moderna interface *Ribbon*®;
- o protótipo de *add-in* para *MS Excel*®, desenvolvido em C#, com moderna interface *Ribbon*®.

A prototipação de uma primeira versão em *MS Excel*® gerou alguns resultados positivos que confirmaram a viabilidade deste trabalho.

Por meio desse protótipo em *MS Excel*®, foi confirmado que os *AutoShapes* do *MS Excel*® poderiam ser manipulados via programação, de forma a facilitar o desenho de diagramas de nodalização para o RELAP5. O *MS Excel*® oferece um modelo de objetos *COM*® suficientemente capaz de manipular os desenhos e diagramas gerados de forma a acelerar a atividade do usuário, que pode focar mais esforço nas opções que envolvem modelos para solução do problema que no manuseio de ferramentas gráficas.

Também foi observado que os *AutoShapes* gerados podem ser agrupados e manipulados posteriormente pela própria interface oferecida pelo *MS Excel*®, diretamente pelo usuário, sem a necessidade de desenvolvimento adicional. Identificou-se também que os dados de entrada do RELAP5 podem ser completamente armazenados em planilhas *MS Excel*®, integrando-os totalmente com o diagrama (nodalização) criado.

O protótipo confirmou que a entrada de dados para o RELAP5 pode ser toda armazenada em planilhas *MS Excel*®, fazendo assim com que a ferramenta possa fazer uma total integração entre a nodalização criada e os dados de entrada gerados, permitindo que o usuário possa, por exemplo, clicar num componente hidráulico na nodalização e navegar para a parte do código RELAP5 que contém o referido componente.

Esse protótipo confirmou também a viabilidade de tradução em tempo real das telas, permitindo assim o desenvolvimento de uma única planilha pré-processadora que traduz automaticamente seus elementos de tela para as línguas desejadas. Isso evitaria a necessidade de manutenção de diferentes edições da ferramenta em diferentes planilhas, como ocorreu nas versões anteriores, onde toda atualização do pré-processador implicava na modificação e redistribuição de uma edição em português e outra em inglês, da mesma ferramenta (SILVESTRE, 2016). Além disso, verificou-se que é viável que os elementos a

serem traduzidos possam ser armazenados fora da programação VBA, ficando numa aba da planilha. Isso facilitaria a localização e tradução do complemento para outras línguas adicionais, além das originalmente desenvolvidas. A localização dessa aba pode ser feita por qualquer pessoa que tenha a planilha, sem necessidade de acesso ao programa-fonte.

Contudo, no desenvolvimento desse protótipo observou-se algumas ressalvas, tais como:

- atualização do programa: uma vez que toda a programação estivesse em VBA, toda e qualquer atualização realizada só se aplicaria aos novos problemas, uma vez que os arquivos do *MS Excel*® contendo problemas previamente existentes, também conteriam em si a versão antiga da programação. A menos que, a cada atualização, toda a programação em VBA dos problemas criados fossem substituídas com a nova programação, não seria possível a atualização da versão da ferramenta para problemas em andamento;
- a linguagem VBA fica devendo alguns recursos modernos que foram essenciais para reduzir a complexidade do *add-in* para o pré-processador do RELAP5, tais como herança de telas, criação de componentes reutilizáveis, criação de objetos estruturados, polimorfismo de classes e métodos e funções, e tratamento estruturado de erros.

Desta forma, com os resultados preliminares obtidos na fase do protótipo, optou-se pelo desenvolvimento do pré-processador no formato de complemento (*add-in*) para o *MS Excel*® programados na linguagem C#, que consome o modelo de objetos do *MS Office*® por meio do *Interop*.

4.2 Programação do pré-processador

A opção pelo desenvolvimento do pré-processador em um *add-in MS Excel*®, feita por meio da programação em C#, permitiu que o problema de atualização e distribuição de novas versões da solução possa ser aproveitado em problemas previamente criados com as versões anteriores do pré-processador, uma vez que a programação das regras deste pré-processador não fica mais copiada dentro de cada planilha usada para criar novos problemas, mas sim dentro do *add-in* criado.

A reutilização da programação obtida com a linguagem C# reduz em muito a prática do “recorta-e-cola” no desenvolvimento do *add-in* do pré-processador. Apesar de resolver rapidamente o problema, a prática do “recorta e cola” gera sérios entraves para a escalabilidade dos programas criados dessa forma, como ocorreu na programação em VBA.

Não são raros os casos em que a prática do “recorta e cola” geram defeitos nos programas, pelo fato de determinada manutenção ter sido feita somente num trecho da programação, mas não nos demais trechos em que aquelas linhas de programação foram copiadas e adaptado.

Com recursos de herança e polimorfismo, as classes funcionais criadas para abstração do código RELAP5 tornaram-se mais reaproveitáveis sem necessidade de replicar a programação desnecessariamente, fazendo o reaproveitamento da programação de forma adequada. Da mesma forma, a herança em telas do *Windows*®, bem como a criação de controles reutilizáveis, foram fatores primordiais para bom reaproveitamento de programação escrita tanto para validação dos parâmetros de entrada como para garantir a padronização das telas dentro do pré-processador.

A construção da interface *Ribbon*® para *MS Excel*® por meio do C# e *Interop* mostrou-se muito mais simples de ser feita do que a construção do *Ribbon*® diretamente no *MS Excel*®: a IDE do *MS Visual Studio*® permite a criação e desenho da interface *Ribbon*® por meio de uma ferramenta gráfica para desenho de telas de forma muito similar à que é feita para desenho de telas para a *Web* ou telas para o *Windows*®. Com isso, o controle dos estados dos botões de acordo com os objetos selecionados dentro do *MS Excel*® se mostrou mais robusto e mais confiável, evitando condições de erro, e permitindo uma interface amigável sem tantas mensagens de erro. A construção do *Ribbon*® diretamente no *MS Excel*® envolvia o desenho da interface *Ribbon*® diretamente em formato XML que era importado dentro da planilha *MS Excel*® alvo.

O recurso de gravação de macros em VBA do *MS Excel*® também pôde ser utilizado. Apesar da programação gerada pelo gravador de macros ser em VBA, não sendo, portanto, automaticamente convertida para C#, como o modelo de objetos do *MS Excel*® é o mesmo exposto tanto para o VBA quanto para o C#, por meio do *Interop*, as macros gravadas foram facilmente aproveitadas e convertidas em código C#.

A interface do usuário com o pré-processador foi melhorada. Os parâmetros exibidos nas telas, representando as propriedades dos componentes hidrodinâmicos, mostram não somente a que cartão e parâmetros inseridos se referem, mas também, um nome amigável, dando um significado físico para o valor que está na tela, reduzindo assim a necessidade de consulta a todo instante do manual dos dados de entrada do RELAP5.

Para a modelagem das interfaces visuais, optou-se por uma abordagem moderna de construção amigável para com o usuário. Essa abordagem tem como premissa uma interface inteligente, que evita que o usuário cometa erros na inserção dos dados de entrada.

A interface consegue oferecer ao usuário uma experiência amigável e silenciosa, trabalhando com as seguintes premissas:

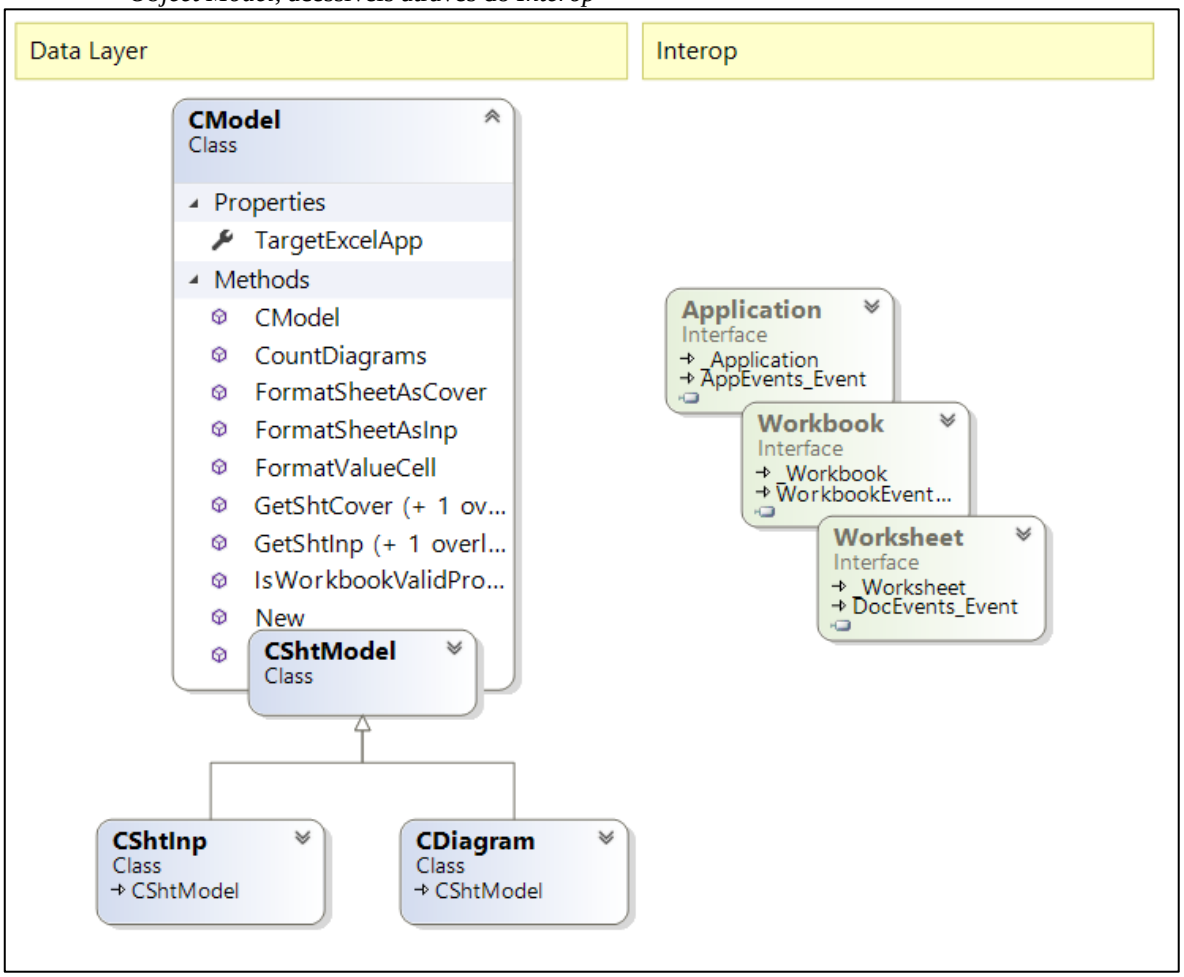
- os campos inválidos têm sua cor de fundo trocada, a fim de alertar o usuário de que algo está errado; caso o mesmo queira saber a razão do erro, basta mover o cursor sobre o campo em vermelho e uma etiqueta de mensagem mostrará o problema;
- as telas possuem definidos em si a ordem de tabulação dos campos, facilitando a navegação entre os campos, com teclas de atalho no teclado, e não somente via cliques do mouse;
- ao mesmo tempo em que o usuário pode preencher os campos na ordem em que desejar, a interface impede, de forma silenciosa, que um componente seja inserido no diagrama com dados inválidos. Ao invés de uma mensagem *popup* aparecer ao clicar no botão de criação informando ao usuário de que há campos inválidos na tela, o botão de criação é desabilitado, e permanece nessa condição até que os erros sejam corrigidos. Os erros podem ser corrigidos no momento em que o usuário decidir, mas enquanto não forem retificados, o programa garante que o usuário não prossiga usando inválidos: tudo isso com uma interface amigável, não-intrusiva e silenciosa;
- da mesma forma, situações de erro são evitadas ao habilitar/desabilitar os controles da tela. Sempre que o usuário desejar tentar entender o motivo de determinado controle estar desabilitado, basta passar o mouse sobre o mesmo que uma etiqueta será exibida mostrando o motivo;
- a interface jamais bloqueará a navegação do usuário de um campo para outro até que a informação válida seja inserida; afinal, a realidade multitarefa do ambiente *Windows*® não condiz com esse tipo de bloqueio. Antes de querer corrigir o conteúdo de um campo da tela, o usuário pode querer alterar o tamanho da tela, que não tem absolutamente nenhuma relação com a informação inválida; porém, o comportamento de alguns programas, escritos com esse princípio de bloqueio, acaba resultando numa experiência final desagradável para o usuário ao utilizar o programa, e consequentemente desestimulando o mesmo a continuar usando o programa. A interface foi desenvolvida de forma a evitar esse tipo de comportamento.

Para facilitar a manutenção futura de novas versões do *add-in* criado nesse trabalho, optou-se por uma abordagem de programação que, apesar de inicialmente mais trabalhosa, dá condições para que manutenções futuras possam ser feitas sem a necessidade de replicar essas modificações em diferentes pontos do programa fonte. Algumas das características dessa abordagem são:

- todas as constantes usadas no programa ficam centralizadas numa classe estática, em constantes definidas na linguagem C#, ao invés de ficarem *hardcoded* espalhadas no programa fonte. Com isso, caso haja necessidade de mudar algum determinado padrão de programação, ou de formatação, a mudança ocorre em um único local do programa fonte;
- todas as transformações de formatos de dados são feitas em métodos e classes especificamente criadas para tal, ao invés de serem feitas chamadas diretamente em qualquer trecho do programa, evitando-se assim erros por determinado formato de dados, por exemplo, ser alterado num trecho do programa e não em outro;
- todas as variáveis criadas no programa fonte possuem um padrão de nomenclatura indicando, por prefixos, seu tipo de dado e seu escopo, e facilitando assim a leitura da programação, permitindo saber a natureza da informação e o escopo com uma leitura simples do nome da variável. Dessa forma evita-se erros de programação devido ao programador estar esperando um formato de dado diferente do que está armazenado na variável. Apesar da IDE do *MS Visual Studio® Community* oferecer rico suporte a *IntelliSense*, permitindo rapidamente navegar para a definição de variáveis, ou inspecionar sua declaração, optou-se por manter essa padronização de nomenclatura por ser uma boa prática de programação;
- analogamente, todas as mensagens visíveis aos usuários, quer por meio de mensagens *popup*, ou por legendas e etiquetas de tela, legendas de botões, entre outros, foram centralizadas numa classe de traduções;
- abordagem de modelagem 3-camadas/n-camadas (*3-tier/n-tier modeling*): definição de classes específicas, divididas em separações lógicas de interface, regras e manipulação de dados. As classes criadas para regras de validação não podem interagir com os usuários via interfaces gráficas: somente por meio de interfaces programáticas. As classes de regras de validação também não

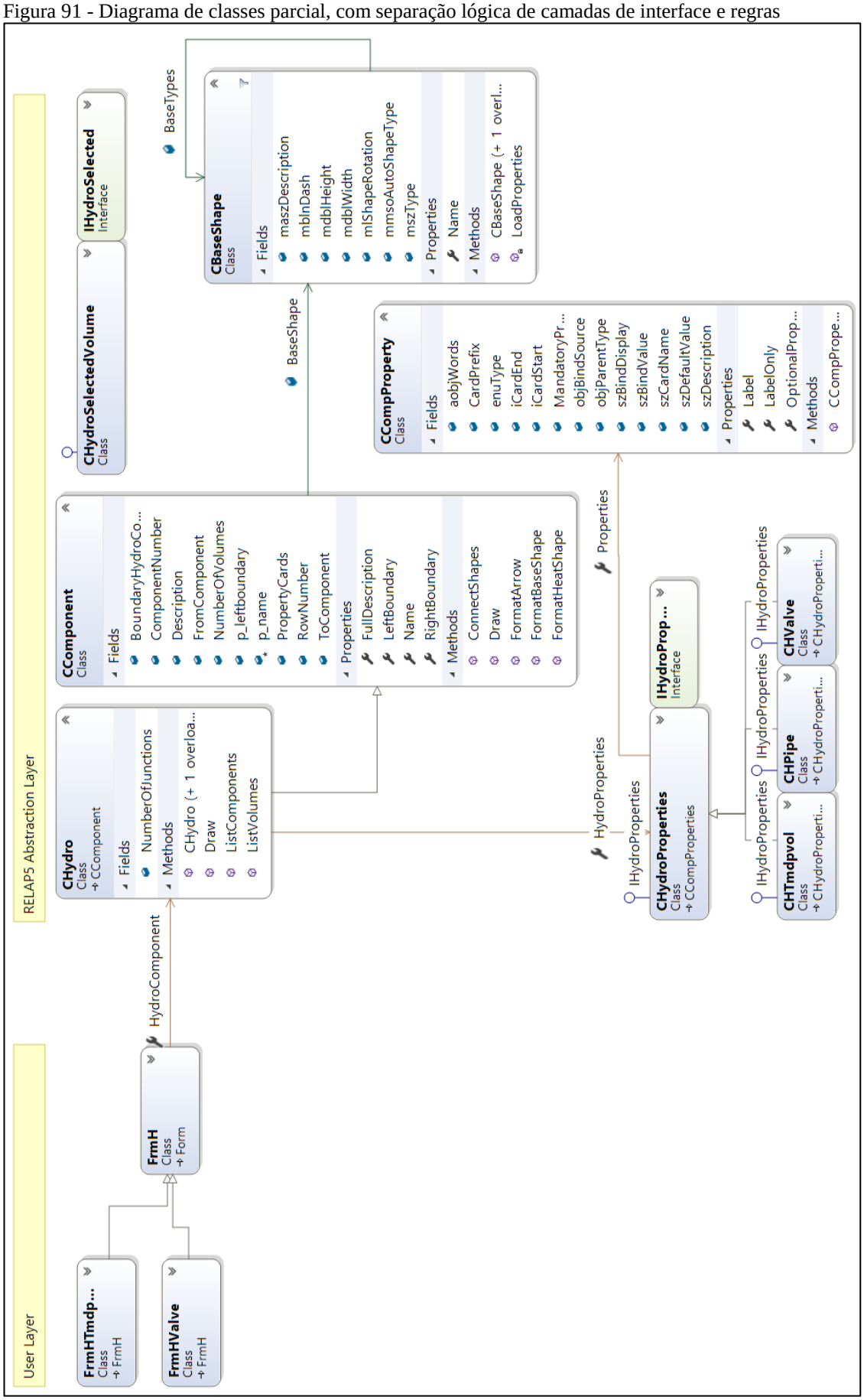
acessam diretamente as planilhas *MS Excel*® usadas como repositório de dados; esses acessos ficam centralizados em classes específicas para manipulação dos dados de entrada do RELAP5. Quando uma tela informa que precisa criar um componente no diagrama, a tela faz a chamada às classes apropriadas; um grupo de classes se encarrega do desenho dos componentes no diagrama, outro grupo é responsável por fazer a criação dos dados de entrada do RELAP5 na planilha apropriada. Tal representação lógica pode ser melhor observada nas Figuras 90 e 91.

Figura 90 - Diagrama de classes parcial, com separação lógica de camadas de dados e *MS Excel® Document Object Model*, acessíveis através do *Interop*



Fonte: autor da dissertação.

Figura 91 - Diagrama de classes parcial, com separação lógica de camadas de interface e regras



Fonte: autor da dissertação.

A Plataforma *.NET*® oferece ao C# serviços essenciais para a criação de um complemento robusto e flexível. Um exemplo disso é o oferecimento de interpretadores JSON integrados à Plataforma *.NET*®. Os elementos de tradução dos componentes e mensagens de tela são serializados em tempo de execução a partir de um arquivo externo, em formato JSON, que será distribuído junto ao complemento pré-processador. O arquivo tem formato amigável de texto, e não exige conhecimento de programação para ser alterado, e também não exige que seja compilado para ser utilizado, pois ele é lido e transformado em objeto durante a execução do complemento. Assim, o complemento poderá ser localizado para qualquer idioma que o usuário deseje, sem que para isso ele precise de acesso ao programa fonte ou tenha conhecimento de programação, evitando, também, a necessidade de manutenção de diferentes versões de planilha do mesmo programa simplesmente por causa do idioma.

Quanto à distribuição do pré-processador desenvolvido nesse trabalho, será simples: o *add-in* ficará compilado em um arquivo DLL, que é copiado para a máquina do usuário, devendo somente ser registrado dentro do *MS Excel*® por meio de interface gráfica disponível pelo próprio, não havendo necessidade de comandos complicados ou criação de instaladores para a sua distribuição.

4.3 Simulação da “Experiência SUPER CANON” com o RELAP5

A modelagem da “Experiência SUPER CANON” foi feita duas vezes. Na primeira vez foram gerados os dados de entrada do RELAP5 manualmente, e na segunda por meio do pré-processador desenvolvido nesse trabalho.

Antes de criar os dados de entrada para o RELAP5, foi necessário um estudo do arranjo experimental inicial para possibilitar a nodalização do experimento e forma adequada para o código RELAP5. Uma prática comum para solucionar problemas complexos de termo-hidráulica é a subdivisão do problema em questão em partes menores, chamadas de volumes de controle, permitindo que a solução das equações de conservação das partes leve à solução do todo. No RELAP5, isso não é diferente. A etapa de nodalização do problema revela-se a etapa mais importante, a fim de garantir que a mesma seja adequadamente transcrita para o código RELAP5 e que os cálculos realizados pelo código possam ser conferidos, ou até mesmo comparados a um experimento real.

O código RELAP5 utiliza o método de diferenças finitas para a solução das equações de conservação, caso os volumes de controle informados à ferramenta sejam muito grandes pode ocorrer que essas equações de conservação não convirjam; por isso os volumes

de controle precisam ser cuidadosamente modelados para permitir que os pontos experimentais definidos estejam no centro desses volumes e tenham tamanhos adequados para que a solução numérica seja possível.

A nodalização da experiência SUPER CANON para o código RELAP, feita manualmente, pode ser vista por meio da Figura 92. O arquivo de dados de entrada criado manualmente pode ser visto no APÊNDICE G.

Por meio da modelagem mostrada na Figura 92, percebe-se que são necessários alguns artifícios para que o RELAP5 possa compreender adequadamente a geometria do problema, bem como do meio ambiente, em que o experimento está inserido.

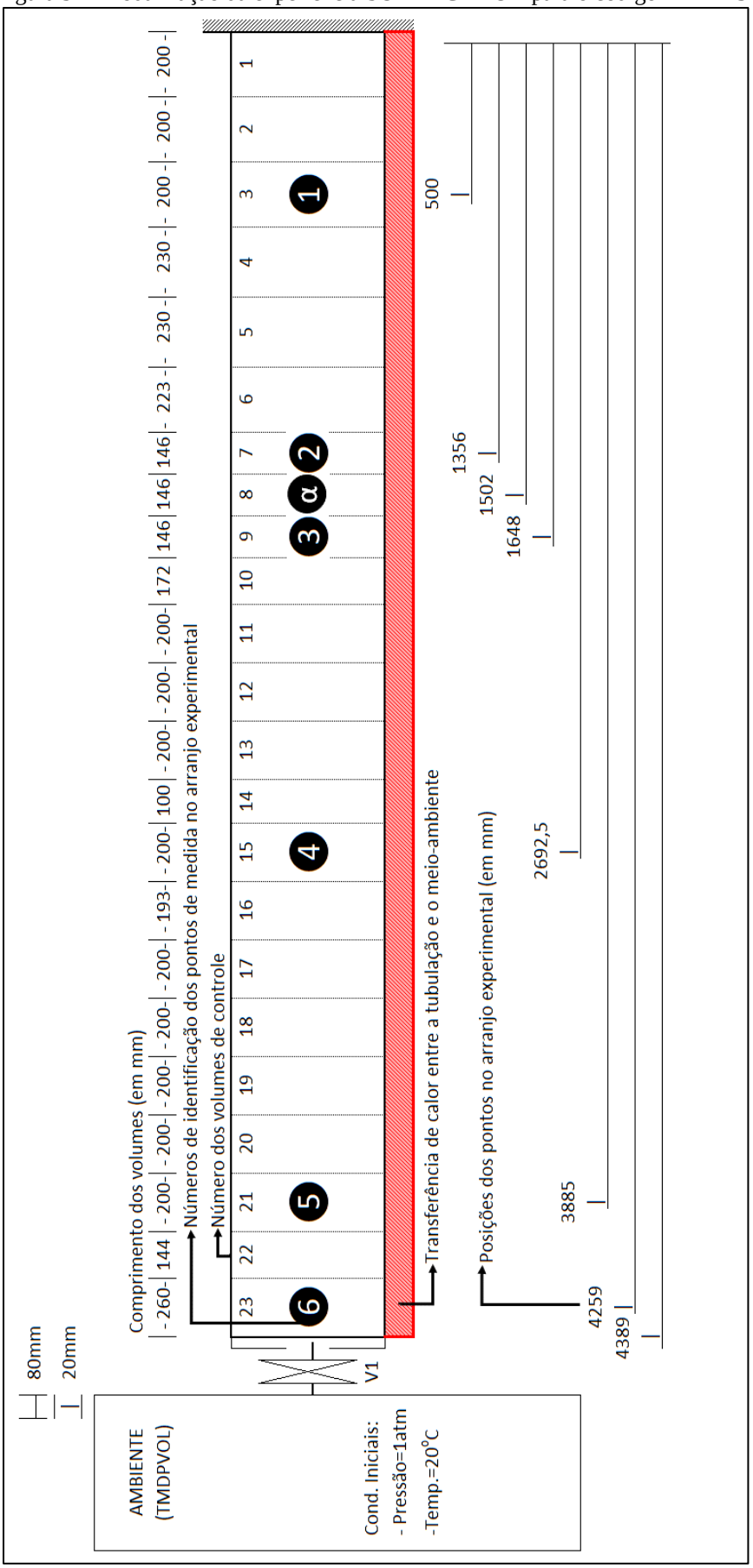
A tubulação principal do experimento foi modelada por meio de um único componente PIPE (tubulação); todavia esse único PIPE foi subdividido em 23 volumes de controle, para que o código possa convergir os resultados mais facilmente.

Para modelar a ruptura foi utilizado um componente do tipo válvula, TRIP VALVE. Essa válvula, para o RELAP5, possui um comportamento de abertura instantânea, ou seja, num intervalo de tempo instantâneo a válvula pula do estado completamente fechado para o estado completamente aberto. Apesar desse tipo de abertura instantânea de válvula ser fisicamente impossível, como a transição de estado nesse acidente é abrupta, esse tipo de válvula do RELAP5 comporta-se de forma muito similar à que ocorre em uma ruptura abrupta.

Por fim, para a modelagem do ambiente, lançou-se mão de um componente do RELAP5 chamado TMDPVOL (ou *Time-dependent Volume*). Esse componente geralmente é usado para representar no RELAP5 volumes com propriedades geométricas fixas, tais como altura, comprimento, área de junção, entre outros. Todavia, nesse tipo de componente, pode-se definir os dados operacionais termo-hidráulicos constantes - como pressão e temperatura, ao longo do tempo. Afinal, para o problema em questão, qualquer alteração da temperatura ou pressão no ambiente devido a toda a massa de água ou vapor que está sendo inserida no ambiente é considerada desprezível.

Além disso, a troca de calor entre os componentes é feita no RELAP5 por meio dos cartões *Heat Structures*. Nesse problema, a troca de calor entre a água dentro da tubulação com o material que a constitui e a transferência de calor da tubulação com o ambiente foi, também, considerada nesse caso. A transferência de calor foi modelada nesse experimento a fim de demonstrar a potencialidade do pré-processador desenvolvido.

Figura 92 - Nodalização da experiência SUPER CANON para o código RELAP5



Fonte: autor da dissertação.

Os pontos marcados em preto no diagrama anterior representam pontos onde os termopares foram instalados no arranjo experimental realizado em laboratório.

As regiões hachuradas representam as estruturas de troca de calor entre o fluido, a tubulação e o ambiente.

A "Experiência SUPER CANON" foi realizada em diferentes condições de pressão, temperatura e tamanhos da ruptura. Todavia, para este trabalho, foi escolhido somente um conjunto de condições iniciais para simulação com o código RELAP5, a saber:

- temperatura do fluido (água) em 300°C e a pressão em 150 bar;
- temperatura do ambiente em 20°C e pressão em 1 atm;
- ruptura circunferencial total, ou seja, área do esvaziamento equivalente à 100% da área de vazão da tubulação.

A partir da nodalização exibida anteriormente, foi possível fazer a criação manual dos dados de entrada para o código RELAP5 versão MOD3.3.

No item 4.5 são apresentados os resultados obtidos com os dados de entrada do RELAP5 obtidos manualmente, com os dados de entrada gerados com o pré-processador desenvolvido nesse trabalho para o RELAP5, e, também, são comparados os resultados de ambos aos obtidos experimentalmente, segundo SABUNDJIAN et. al. (1986).

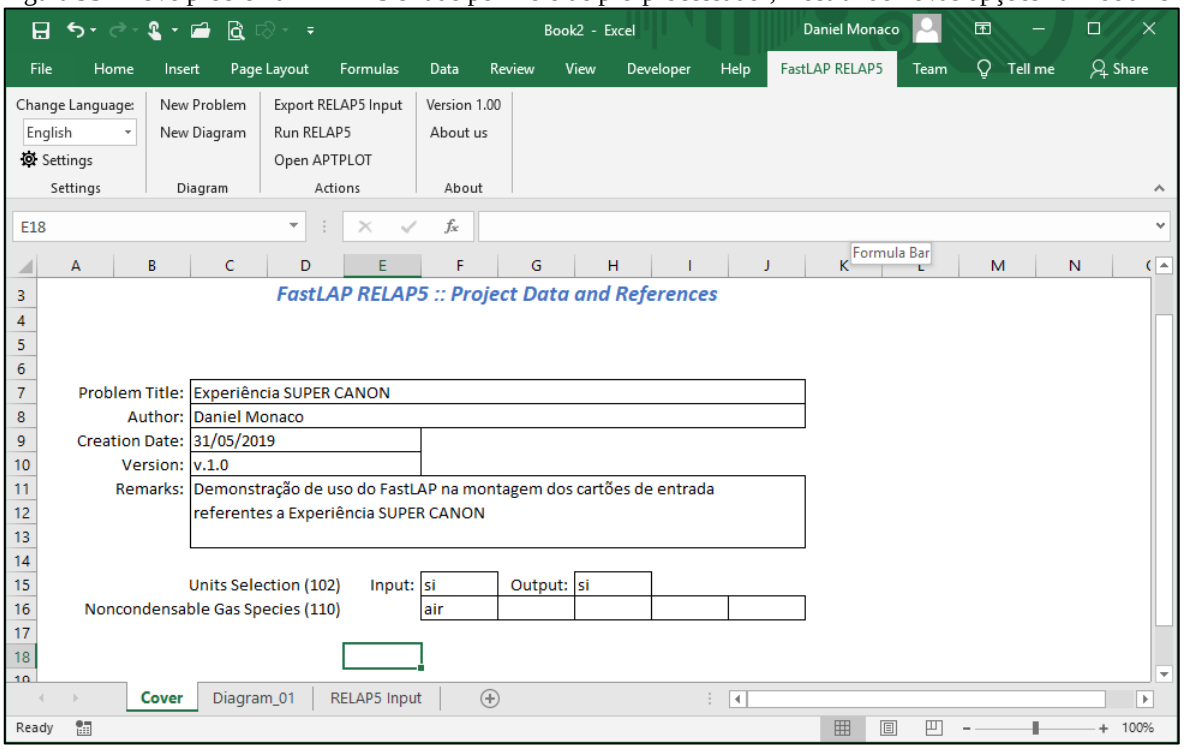
4.4 Utilização do pré-processador

Nesse item são apresentadas as telas criadas para o RELAP5, passo a passo, dos dados de entrada da "Experiência SUPER CANON" utilizando o pré-processador gráfico desenvolvido nesse trabalho.

Ao iniciar uma nova janela do *MS Excel*® onde o pré-processador está instalado, o primeiro passo é criar um novo problema RELAP5 com o *add-in*.

Após clicar no botão "novo problema", uma nova planilha *MS Excel*® é criada, com formato de abas e tratamentos específicos para ser gerenciada pelo pré-processador. Assim que um novo problema é criado, novos botões são disponibilizados na *Ribbon*® do *add-in*, conforme Figura 93.

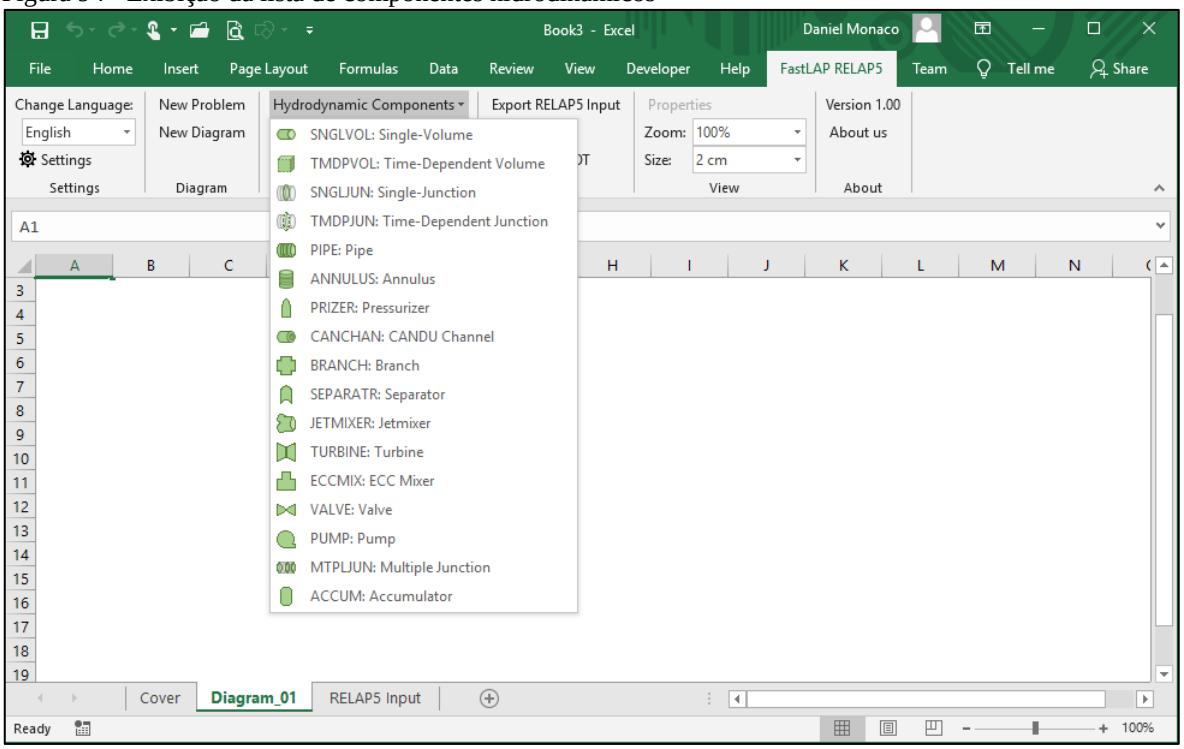
Figura 93 - Novo problema RELAP5 criado por meio do pré-processador, mostrando novas opções na *Ribbon*®



Fonte: autor da dissertação.

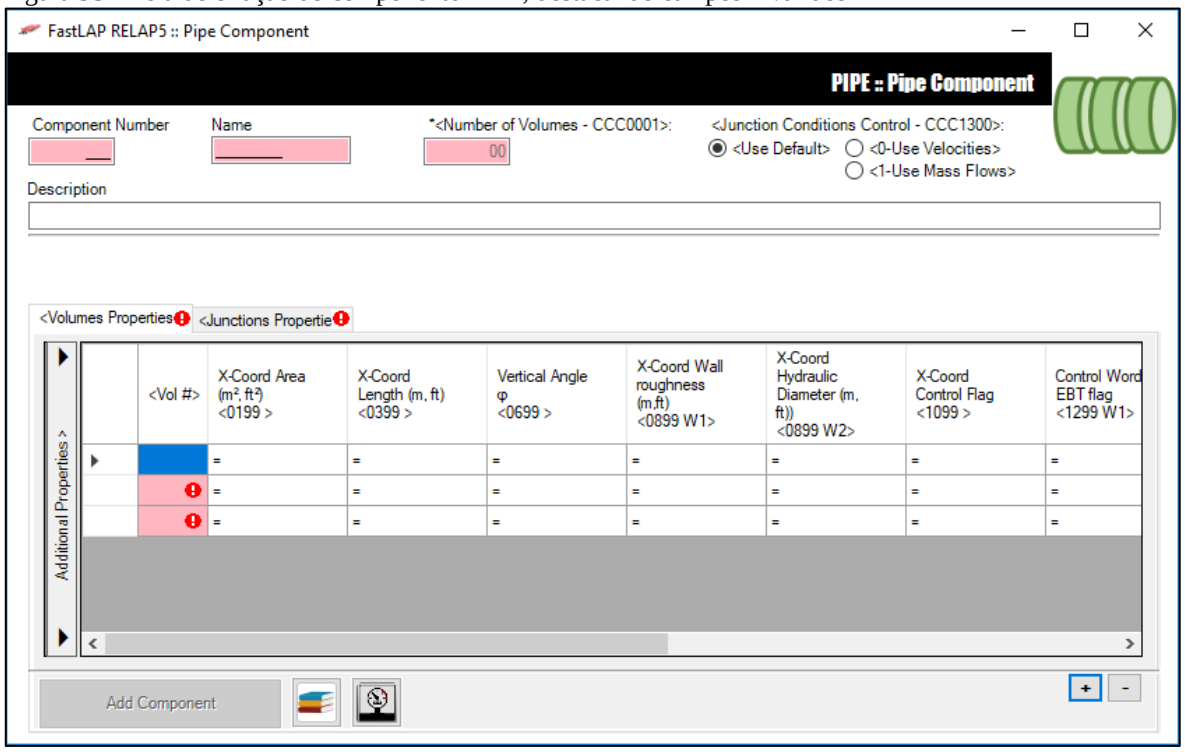
Os objetos são inseridos no diagrama por meio da seleção do componente desejado, conforme Figura 94, e posterior preenchimento da tela correspondente, de acordo com a Figura 95.

Figura 94 - Exibição da lista de componentes hidrodinâmicos



Fonte: autor da dissertação.

Figura 95 - Tela de criação do componente PIPE, destacando campos inválidos

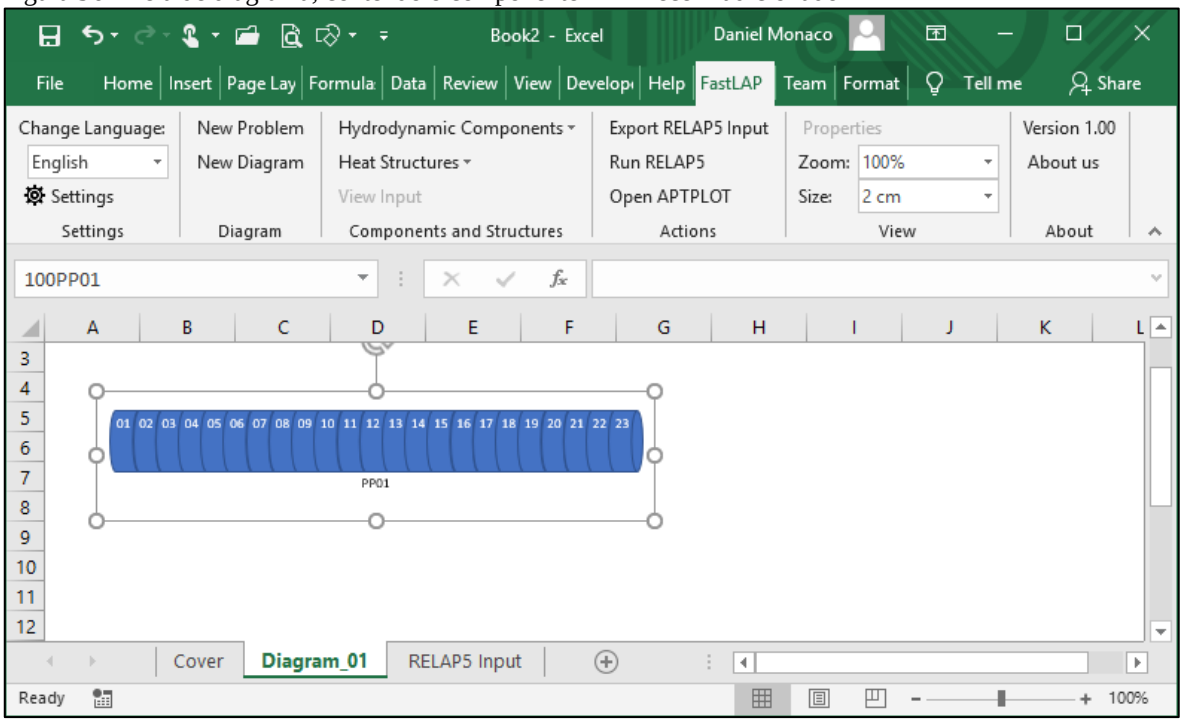


Fonte: autor da dissertação.

Na Figura 95, nota-se que a tela mostra os problemas sem gerar mensagens de *popup* intrusivas, permitindo aos usuários resolver os erros na ordem desejada. Observa-se, também, que o botão de adicionar componentes permanece desabilitado até que os erros sejam sanados.

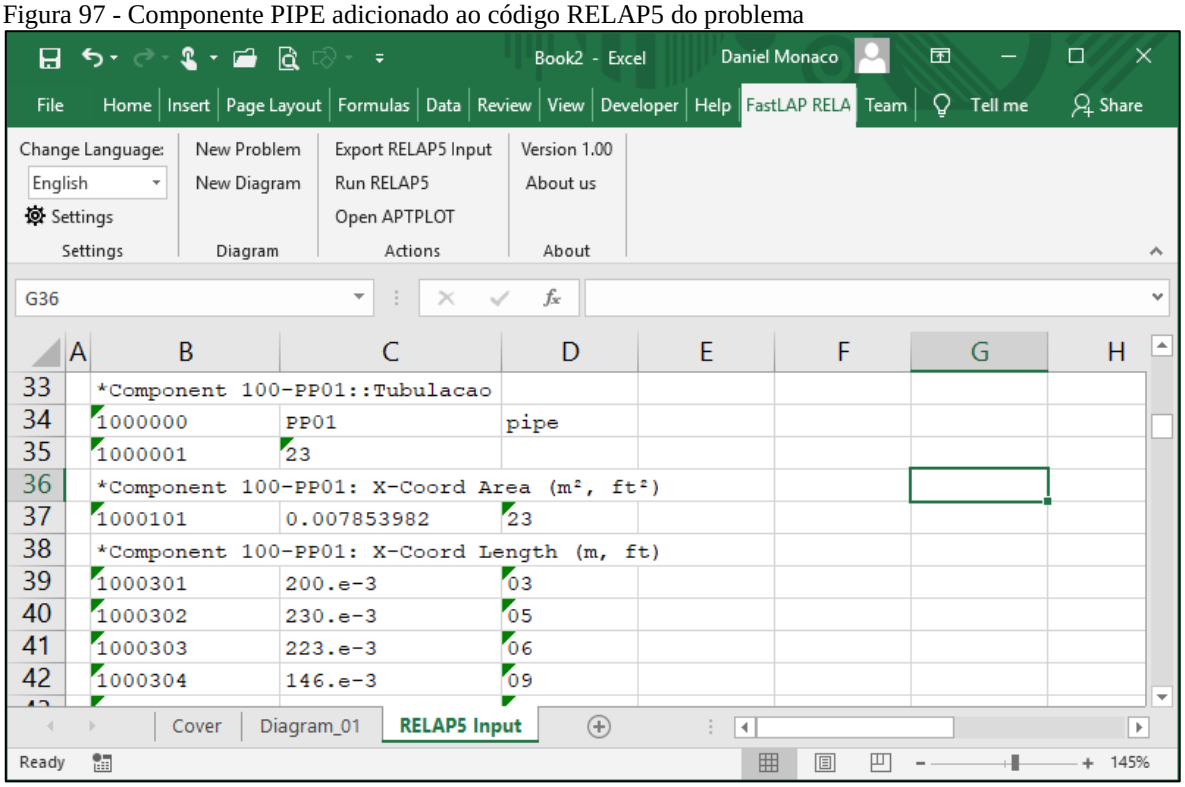
Assim que todos os erros da tela foram sanados, o botão de adição do componente ao diagrama foi habilitado. Após clicar no botão, o componente foi desenhado no diagrama, conforme mostrado na Figura 96, e pode ter sua formatação e cores alteradas pelo usuário conforme desejado. Além disso, os dados de entrada do RELAP5 deste componente já foram adicionados à aba correspondente, e ficam disponíveis para verificação a qualquer momento, conforme Figura 97.

Figura 96 - Tela de diagrama, contendo o componente PIPE recém-adicionado



Fonte: autor da dissertação.

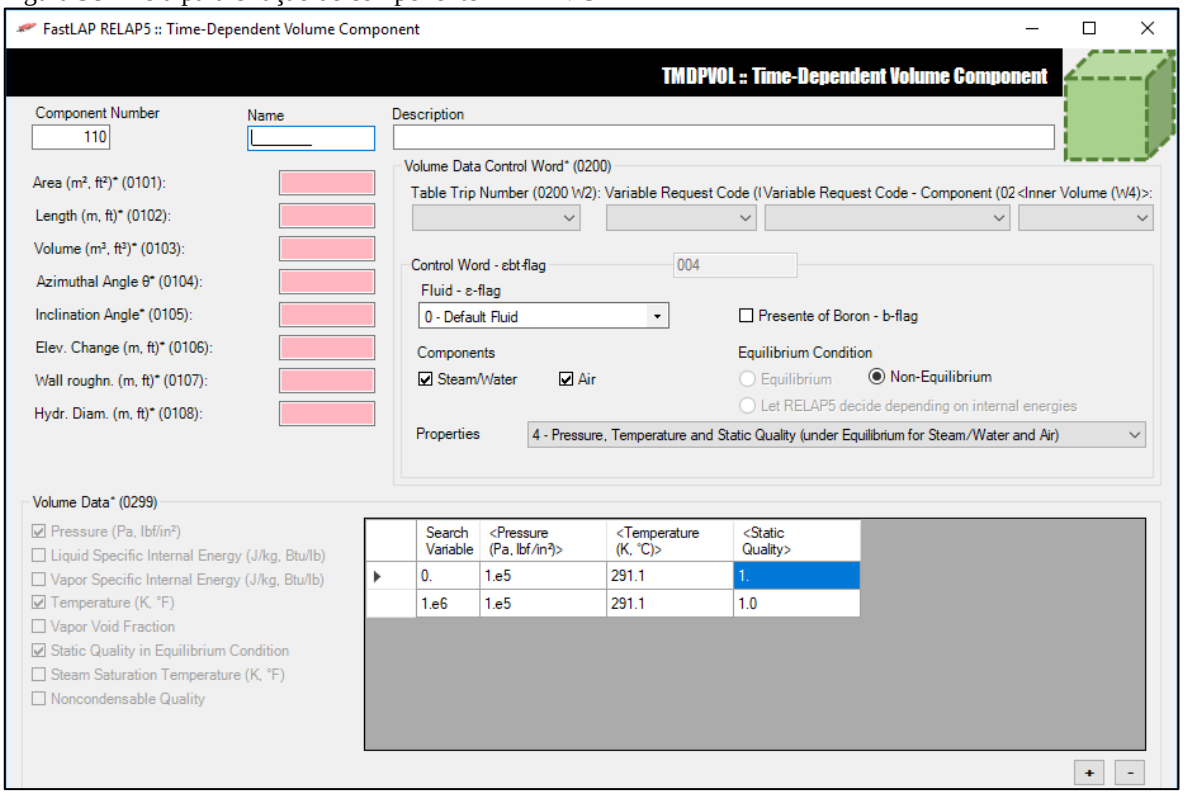
Figura 97 - Componente PIPE adicionado ao código RELAP5 do problema



Fonte: autor da dissertação.

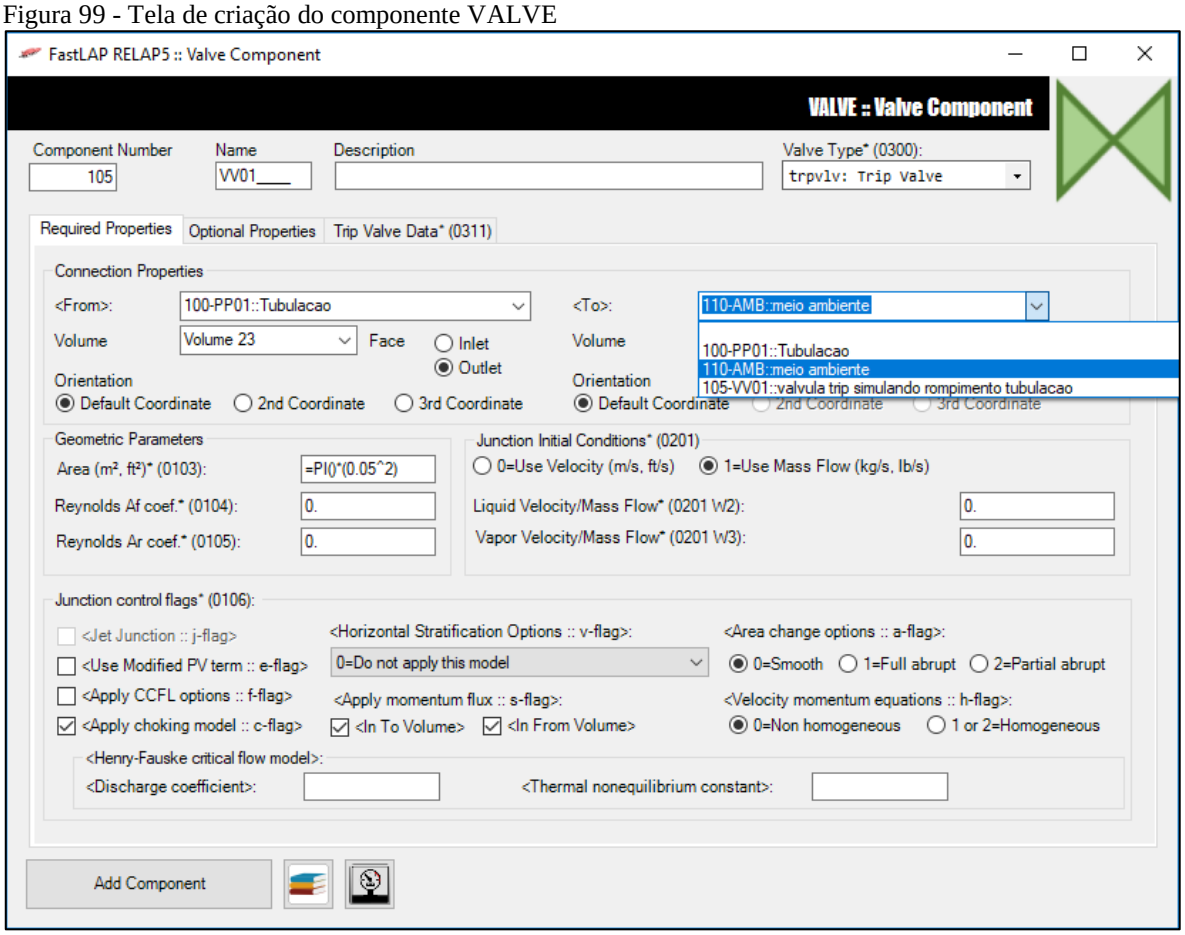
De forma análoga, os demais componentes VALVE e TMDPVOL serão adicionados ao diagrama para permitir a modelagem do problema para o RELAP5, conforme as Figuras 98 a 99.

Figura 98 - Tela para criação de componente TMDPVOL



Fonte: autor da dissertação.

Figura 99 - Tela de criação do componente VALVE

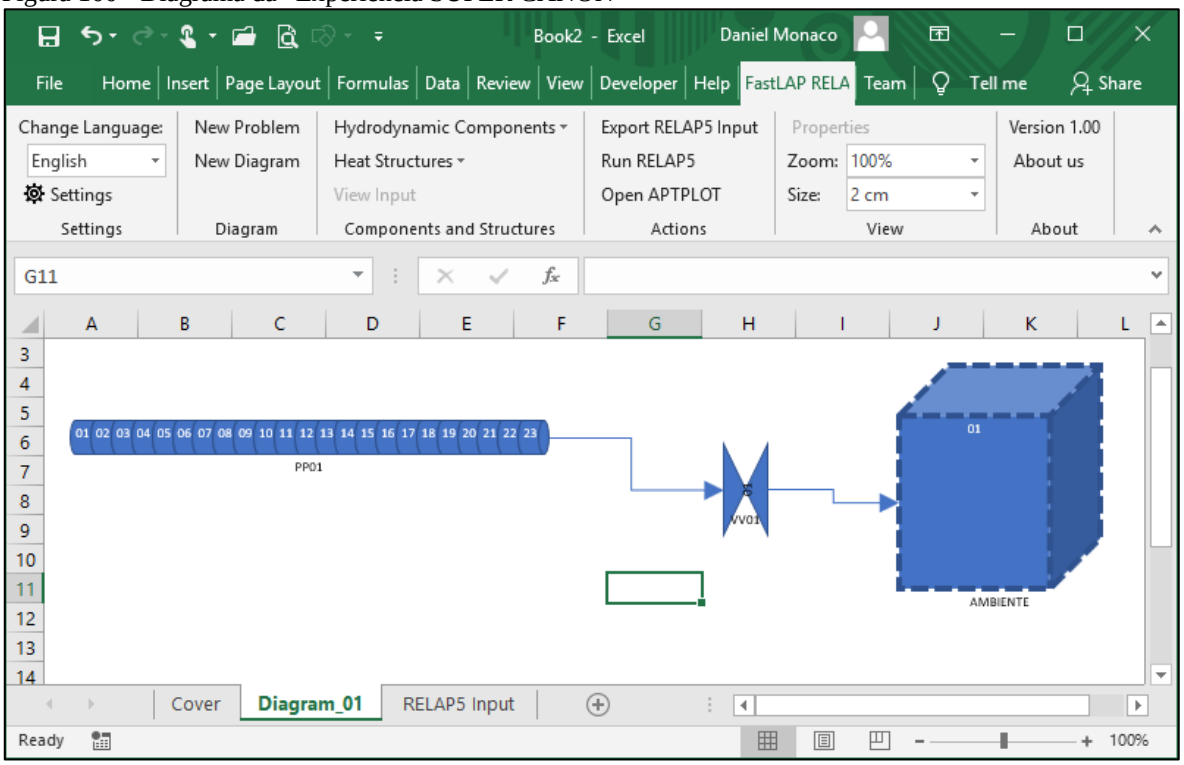


Fonte: autor da dissertação.

O componente VALVE é um componente do tipo junção, ou conexão. No RELAP5, a conexão de componentes se dá com um padrão numérico complexo, que precisa ser informado como parâmetro para a criação do componente VALVE. Por meio do pré-processador, uma tela amigável permite a busca e seleção de todos os componentes já criados e existentes no problema atual, incluindo não só o nome e o número do componente, mas também o metadado de descrição alimentado pelo usuário, quando da criação dos componentes, como pode ser visto na Figura 99.

A Figura 100 apresenta o diagrama da "Experiência SUPER CANON" até então.

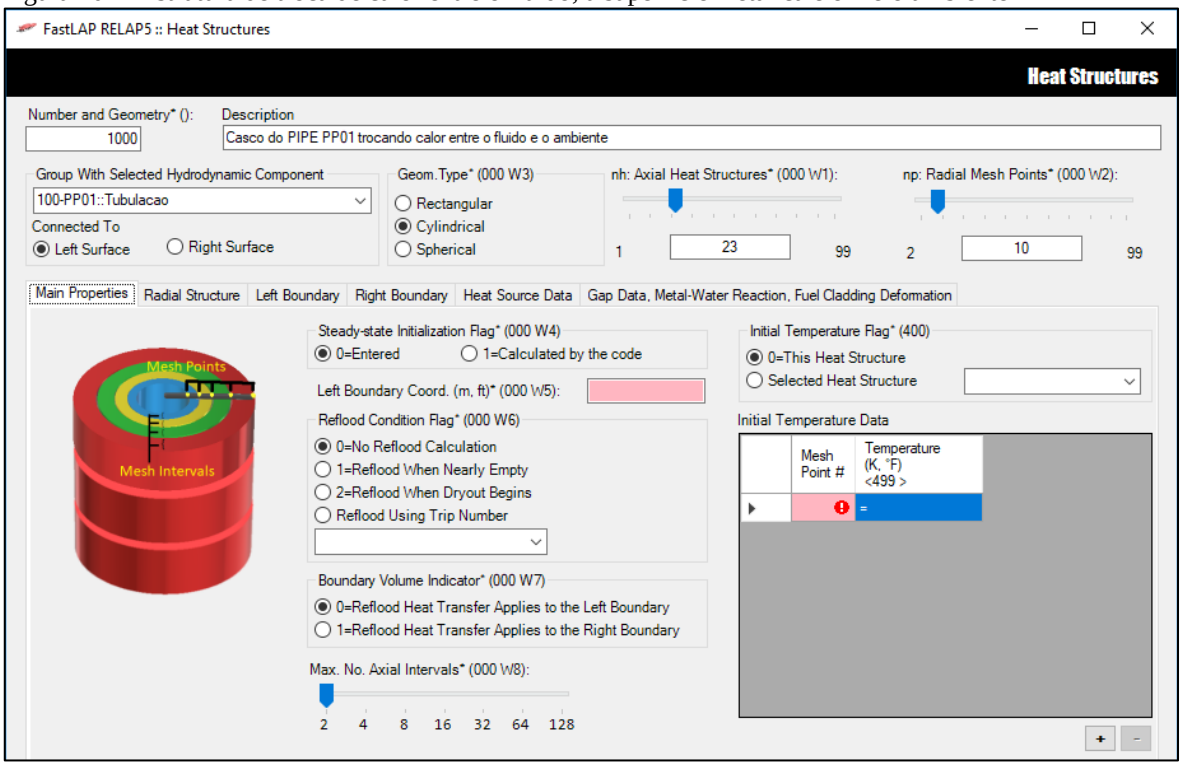
Figura 100 - Diagrama da "Experiência SUPER CANON"



Fonte: autor da dissertação.

As estruturas de troca de calor (HEAT STRUCTURE) são inseridas no componente PIPE, sendo que a troca de calor ocorre entre o fluido no interior da tubulação e a superfície metálica e entre a superfície e o meio ambiente. As telas correspondentes a esse componente são apresentadas na Figura 101.

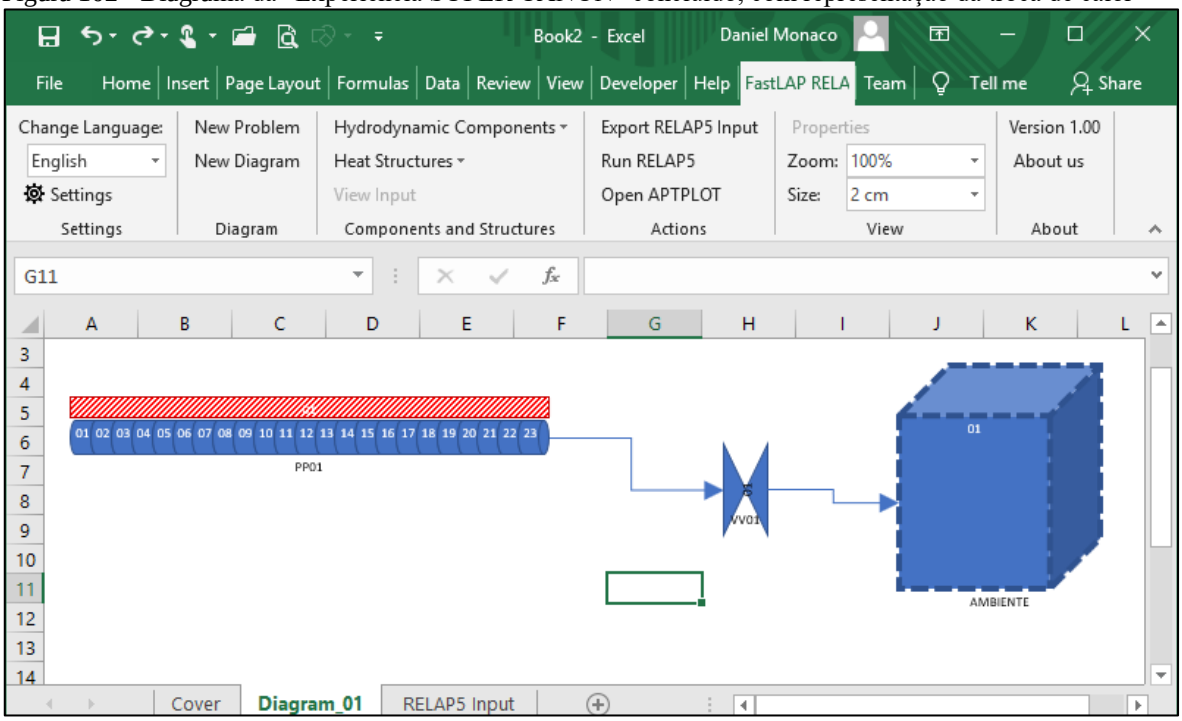
Figura 101 - Estrutura de troca de calor entre o fluído, a superfície metálica e o meio ambiente



Fonte: autor da dissertação.

A Figura 102 é mostrada a modelagem final da “Experiência SUPER CANON”.

Figura 102 - Diagrama da “Experiência SUPER CANON” concluído, com representação da troca de calor



Fonte: autor da dissertação.

Uma vez finalizada a modelagem do problema, o usuário está pronto para exportar os dados de entrada para o RELAP5, e em seguida executar o problema diretamente

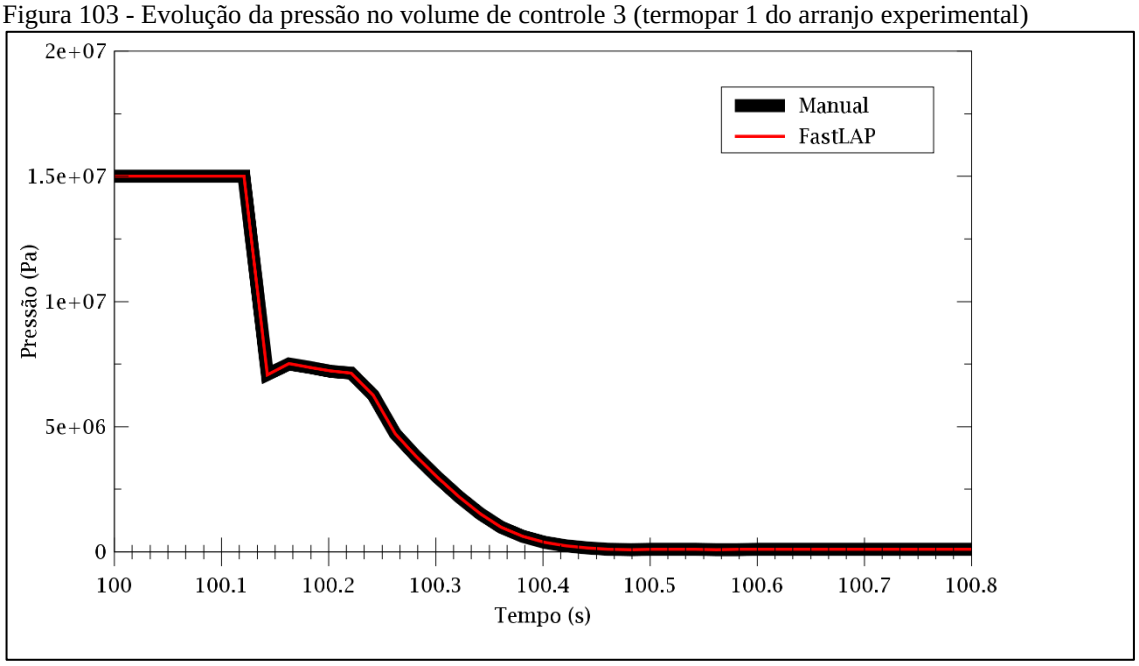
da ferramenta, por meio das opções disponíveis na *Ribbon*® do pré-processador. O arquivo de dados de entrada criados pelo pré-processador para este trabalho pode ser visto no APÊNDICE H.

4.5 Resultados obtidos com o pré-processador

Após execução dos dados de entrada gerados pelo pré-processador com o código RELAP5, comparou-se, então, os resultados obtidos desses dados de entrada com os obtidos por meio dos gerados manualmente, ambos representando a mesma "Experiência SUPER CANON" e nas mesmas condições iniciais de simulação. Os resultados podem ser vistos nas Figuras 103 a 108.

A Figura 103 mostra a evolução temporal da pressão do terceiro volume de controle do componente PIPE durante o transiente proposto, onde se encontra o primeiro termopar no arranjo experimental.

Figura 103 - Evolução da pressão no volume de controle 3 (termopar 1 do arranjo experimental)

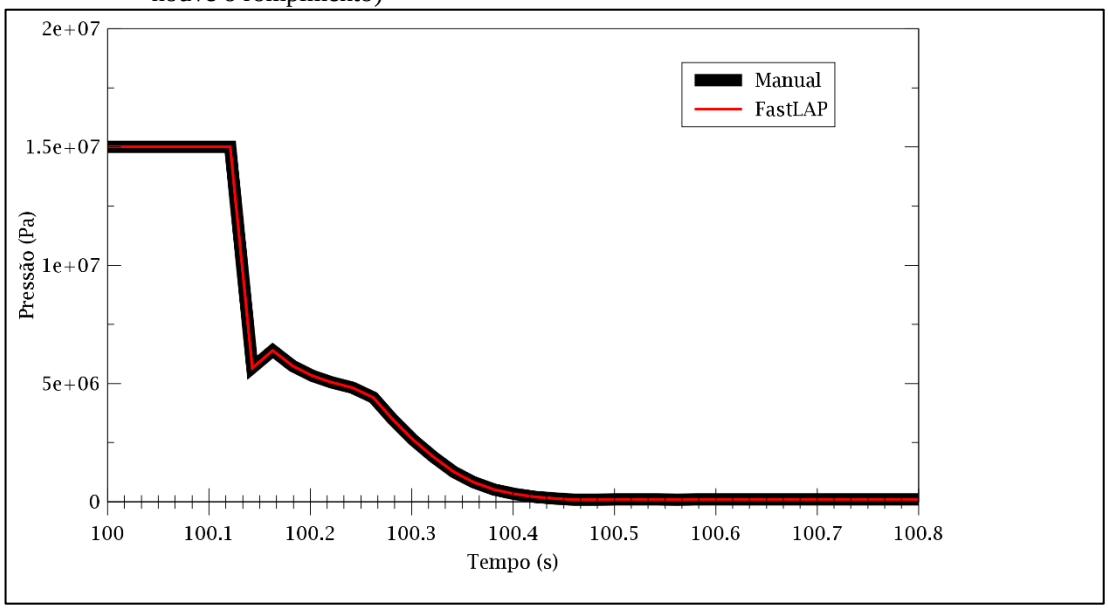


Fonte: autor da dissertação.

Observa-se o regime permanente nos 100 primeiros segundos e após a abertura da válvula aos 100,1s ocorre uma rápida despressurização nesse volume. Em torno dos 100,2s há uma pequena elevação da pressão devido a onda de pressão que ocorre no tubo durante a sua despressurização, até equalizar a pressão com a pressão ambiente, que se estabiliza em 10^5 Pa ou 1 bar a partir dos 100,4s. Observa-se, também, que houve plena concordância entre os dois resultados obtidos, tanto os calculados manualmente como os

obtidos por meio do pré-processador. O mesmo comportamento da pressão é observado no vigésimo terceiro volume de controle do PIPE ao longo do tempo, conforme Figura 104.

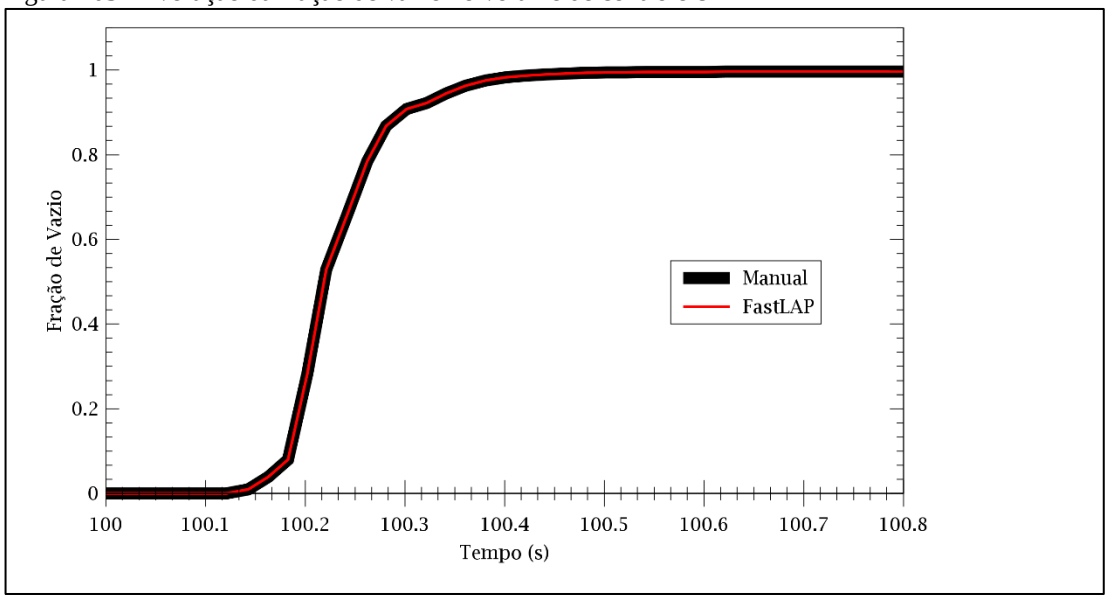
Figura 104 - Evolução da pressão no volume de controle número 23 (extremidade da tubulação onde houve o rompimento)



Fonte: autor da dissertação.

Da mesma forma que nos casos anteriores, o comportamento da fração de vazio ao longo da simulação, tanto com dados obtidos com o pré-processador como os manualmente, foi idêntico, como pode ser visto pela Figura 105.

Figura 105 - Evolução da fração de vazio no volume de controle 8

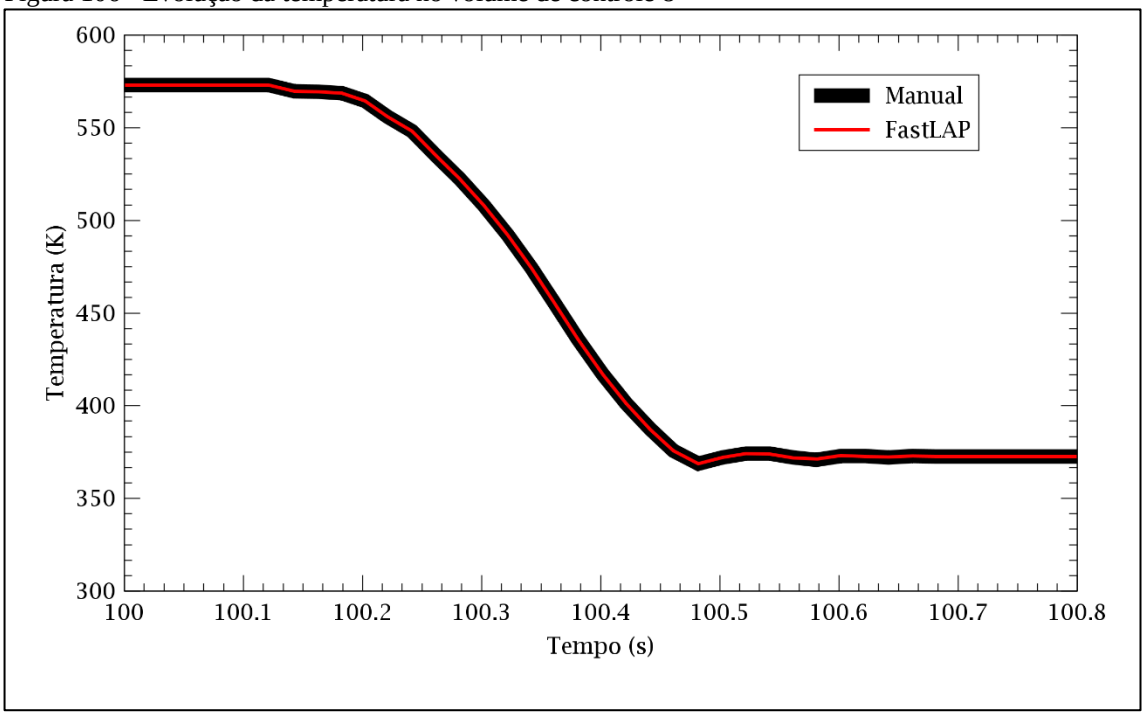


Fonte: autor da dissertação.

Inicialmente a frações de vazio do vapor era zero, pois o tubo estava preenchido de água, e no final da despressurização ficou estável em 1,0, mostrando assim, que toda a água foi vaporizada.

Na Figura 106 é mostrada a evolução temporal da temperatura do oitavo volume de controle do componente PIPE, onde se encontra o medidor de fração de vazio no experimento, durante o acidente proposto. Após a abertura da válvula aos 100,1s ocorre uma rápida diminuição da temperatura, estabilizando-se em torno de 375K (≈102°C) devido ao fim da despressurização, quando ocorre a equalização da temperatura com a ambiente.

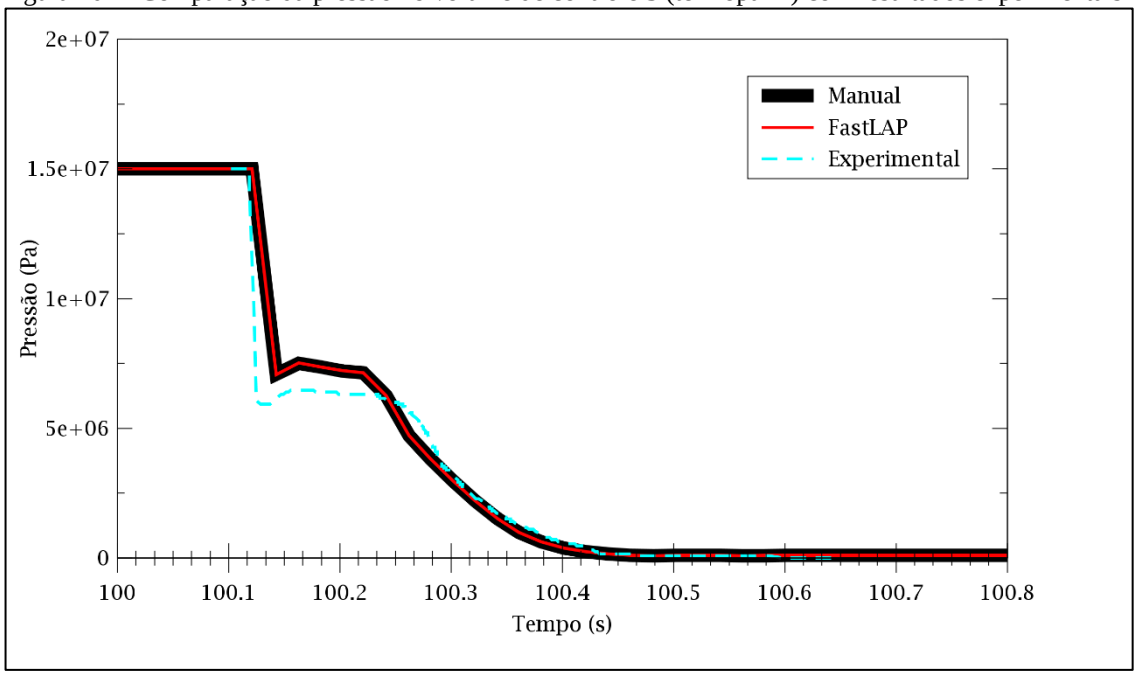
Figura 106 - Evolução da temperatura no volume de controle 8



Fonte: autor da dissertação.

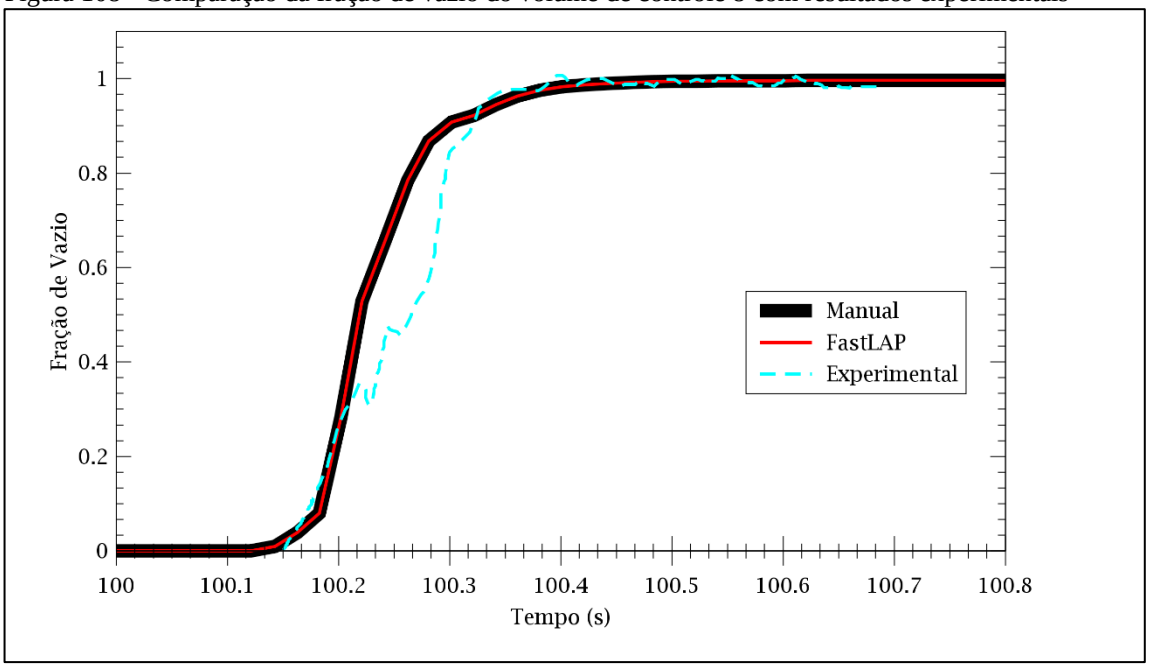
As Figuras 107 e 108 mostram que os resultados obtidos nesse trabalho apresentam o comportamento das variáveis pressão e fração de vazio, respectivamente, semelhantes aos obtidos experimentalmente. Os resultados experimentais foram obtidos a partir do trabalho de SABUNDJIAN et. al. (1986).

Figura 107 - Comparação da pressão no volume de controle 3 (termopar 1) com resultados experimentais



Fonte: autor da dissertação.

Figura 108 - Comparação da fração de vazio do volume de controle 8 com resultados experimentais



Fonte: autor da dissertação.

Os resultados obtidos nesse trabalho demonstram a eficiência do pré-processador criado quanto a sua capacidade de cálculo, de gerar os dados de entrada para o código RELAP5 e de executar o programa.

5 CONCLUSÕES

O pré-processador apresentado neste trabalho, o *FastLAP*, mostrou-se uma poderosa ferramenta gráfica que permitirá aos atuais usuários do código RELAP5 reduzirem tempo e esforço na criação e depuração de dados de entrada para o mesmo. Com a ajuda deste *add-in* para *MS Excel*®, os usuários do RELAP5 poderão concentrar seus esforços na análise dos resultados e no estudo da fenomenologia envolvida no problema, e não em questões de formatação e sintaxe de dados de entrada para o código.

Como o *FastLAP* foi desenvolvido na forma de um *add-in* para o *MS Excel*®, ele agrega as vantagens dessa plataforma, como cálculos em paralelo e ferramentas de desenho. Isso fornecerá aos usuários do RELAP5 uma experiência mais amigável e com o tempo significativamente reduzido.

O diagrama de nodalização, que para os usuários do RELAP5 era uma ferramenta acessória na criação de seus dados de entrada, passa a ter papel principal na criação de dados de entrada para o código RELAP5, uma vez que o desenhar dos componentes no diagrama automaticamente gera os dados de entrada correspondentes. Além disso, as funcionalidades do *MS Excel*® para desenho e formatação dos gráficos oferecem aos usuários do RELAP5 uma experiência de manutenção e análise de problemas termo-hidráulicos que vai muito além das oferecidas até então.

Por meio do pré-processador, os dados de entrada da “Experiência SUPER CANON” foram criados em alguns instantes, por meio de cliques do mouse de forma amigável e inédita. Os resultados do experimento em questão obtidos com os dados de entrada criados manualmente e por meio do *FastLAP* para o RELAP5 foram idênticos, mostrando assim a confiabilidade oferecida pelo pré-processador desenvolvido neste trabalho.

O presente trabalho oferece uma nova abordagem para a solução de problemas termo-hidráulicos com o RELAP5, não somente ajudando os usuários em uma melhor organização de seus dados de entrada, mas proporcionando melhor produtividade, uma vez que as alterações na nodalização são automaticamente transportadas para os dados de entrada do código em tempo real, de forma confiável e segura.

5.1 Trabalhos futuros

As próximas etapas sugeridas a partir deste trabalho são:

- implementação dos componentes hidrodinâmicos que não foram incluídos nesta versão: SEPARATR, TURBINE, ECCMIX, MTPLJUN e ACCUM;
- implementar a programação para permitir que os componentes criados possam ser alterados por meio das telas de criação de componentes;
- implementar programação para excluir os cartões de entrada quando o componente é excluído do diagrama de nodalização;
- implementar programação para permitir a recuperação da representação gráfica de componentes excluídos acidentalmente do diagrama de nodalização;
- implementar telas para *variable trip* e *logical trip*;
- implementar telas para os blocos de controle;
- implementar tela para controle do tempo de execução do problema;
- implementar tela da cinética de reatores;
- implementar tela para adição de variáveis de saída que não estão na lista *default* do código RELAP5.

REFERÊNCIAS BIBLIOGRÁFICAS

APTPLOT version 6.6.1, 2015. Disponível em: <https://ramp.nrc-gateway.gov/content/aptplot-installation-screenspng> Acesso em: 25 jun. 2019.

ARAUJO FILHO, F.; BARROSO, A. C. O.; BELCHIOR JR., A.; GEBRIM, A. Uma Interface Amigável para Códigos de Simulação Termo-hidráulica de Instalações Nucleares. In: V CONGRESSO GERAL DE ENERGIA NUCLEAR, 1994, Rio de Janeiro, ***Proceedings*** ...Rio de Janeiro, 1994.

BATTISTI, J. ***Macros e programação VBA no Excel 2007***. Santa Cruz, Rio Grande do Sul: Instituto Alpha Editora, 2013.

ENCICLOPÉDIA BRITANNICA. Microsoft Excel. Disponível em: <https://www.britannica.com/technology/Microsoft-Excel> Acesso em: 25 jun. 2019

CHERNOBYL'S. *Legacy: health, environmental and socio - economic impacts and recommendations to the governments of Belarus*, THE RUSSIAN FEDERATION AND UKRAINE. IAEA. VIENNA, 2003.

COREY, G. R. A brief review of the accident at Three Mile Island: **IAEA Bulletin**, v. 21, n. 5, p. 54-59, 1979. Disponível em: <https://www.iaea.org/sites/default/files/publications/magazines/bulletin/bull21-5/21502795459.pdf> Acesso em: 25 jun. 2019.

D'AURIA, F. et al. The Best Estimate plus Uncertainty (BEPU) Approach in Licensing of Current Nuclear Reactors. Nuclear **Engineering and Design**, v. 248, p.317-328, 2012.

DOWNAR, T., XU; SEKER, V. ***PARCS v3.0 - U.S. NRC Core Neutronics Simulator - USER MANUAL***, Department of Nuclear Engineering and Radiological Sciences, University of Michigan, 2010.

ELECTRIC POWER RESEARCH INSTITUTE. **RETRAN-02 - A Program for Transient Thermal Hydraulic Analysis of Complex Fluid Flow Systems**. NP-1850-CCMA, rev. 6.1, June 2007.

GAUNTT, R. O., COLE R.K. ***MELCOR Computer Code Manuals****, v. 1.2: Reference Manuals, Version 1.8.5, Nuclear Regulatory Commission. Prepared by Sandia National Laboratories (SNL) for the US Nuclear Regulatory Commission, Office of Nuclear Regulatory Research, NUREG/CR-6119, Rev.2, 2000.*

GAUNTT, R.; KALINICH, D.; CARDONI, J. ***Fukushima Daiichi accident study*** *(status as of April 2012), Tech. Rep. SAND2012-6173, Sandia National Laboratories, Albuquerque, NM, USA, 2012.*

GESELLSCHAFT FÜR ANGALEN-UND REAKTORSICHERHEIT, ATHLET Mod. 2.1 *Cycle A, Models and Methods*. Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) mbH, 2006.

IDAHO NATIONAL LABORATORY. The RELAP5 Development Team. ***RELAP5/MOD3 Code Manual, Code Structure, System Models, and Solution Methods***, Idaho National Engineering Laboratory, NUREG/CR-5535, INEL-95/0174, Volume 1, 1995

IDAHO NATIONAL LABORATORY. ***Relap5/Mod3.3 Code Manual – Volume I: Code Structure, System Models, and Solution Methods***, Information Systems Laboratories, Inc., Idaho, October 2010.

IDAHO NATIONAL LABORATORY. ***Relap5/Mod3D Code Manual – Volume I: Code Structure, System Models, and Solution Methods***, Information Systems Laboratories, Inc., Idaho, October 2019.

IGAMI, M. P. Z. (Org); VIEIRA, M. M. F. (Org.) Guia para a elaboração de teses e dissertações: programa de Pós-graduação Tecnologia Nuclear – IPEN/USP. 3 ed. São Paulo: Instituto de Pesquisas Energéticas e Nucleares, 2017. Disponível em: <https://intranet.ipen.br/portal_por/conteudo/biblioteca/arquivos/NOVO_GUIA_TESES_E_DISSERTACOES.pdf> Acesso em: 25 jun. 2019.

INTERNATIONAL ATOMIC ENERGY AGENCY. Chernobyl's Legacy: Health, Environmental and Socio-Economic Impacts and Recommendations to the Governments of Belarus, the Russian Federation and Ukraine. The Chernobyl Forum: 2003–2005 Second revised version.

INTERNATIONAL NUCLEAR AND RADIOLOGICAL EVENT SCALE (INES), 1990. Disponível em: <https://www.iaea.org/topics/emergency-preparedness-and-response-epr/international-nuclear-radiological-event-scale-ines>. Acesso em: 01 de mar. 2019.

KLOOS, M. ***Software for Uncertainty and Sensitivity Analyses - SUSA, version 4.0***. GRS, January 2015

KOVACS, J. *C#/.NET History Lesson*, 2007. Disponível em: <http://jameskovacs.com/2007/09/07/cnet-history-lesson/>. Acesso em: 25 jun. 2019.

MICAELLY, J.; BESTUIB, D. CATHARE, the New Improved French Thermal Hydraulic Code for Safety Reactor Studies. In: INTERNATIONAL ENS/ANS CONFERENCE ON THERMAL REACTOR SAFETY, October 2-7, 1988, Avignon, France, ***Proceedings***... Avignon.

MICROSOFT. *Excel performance: Improving calculation performance*, 2017. Disponível em: <https://docs.microsoft.com/en-us/office/vba/excel/concepts/excel-performance/excel-improving-calcuation-performance/>. Acesso em: 21 mai. 2019.

MICROSOFT .NET CORE. *.NET Core Guide*. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/core/>. Acesso em: 27 mai. 2019.

MICROSOFT .NET FRAMEWORK. *Get started with the .NET Framework*, 2019. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/framework/get-started/>. Acesso em: 25 jun. 2019.

MICROSOFT EXCEL. *Spreadsheet software – Microsoft Excel*. Disponível em: <https://products.office.com/en-us/excel>. Acesso em: 25 jun. 2019.

MICROSOFT INTEROP. *Introduction to COM Interop*, 2015. Disponível em: <https://docs.microsoft.com/en-us/dotnet/visual-basic/programming-guide/com-interop/introduction-to-com-interop>. Acesso em: 25 jun. 2019.

MICROSOFT VBA. *Excel VBA reference*, 2018. Disponível em: <https://msdn.microsoft.com/en-us/library/office/ee861528.aspx>. Acesso em: 25 jun. 2019.

MURATA, K.K.; WILLIAMS, D.C.; GRIFFITH, R.O.; GIDO, R.G.; TADIOS, E.L.; DAVIS, F.J.; MARTINEZ, G.M.; WASHINGTON, K.E.; TILLS, J. ***Code Manual for Contain 2.0: A computer code for nuclear reactor containment analysis***. Sandia National Laboratories, Albuquerque, New Mexico, USA, 1997.

NELSON, M. L., ***An introduction to Object-Oriented Programming***, Naval Postgraduate School, Monterey-CA, 1990.

PALADINO, P. A. ***Pré-processador matemático para o código RELAP5 utilizando o Microsoft Excel.*** 2006. Dissertação (Mestrado) - Instituto de Pesquisa Energéticas Nucleares, São Paulo.

ROWE, D. S. ***Crossflow Mixing Between Parallel Flow Channel During Boiling: Part 1. COBRA – Computer Program for Coolant Boiling in Rod Arrays.*** BNWL-371. Richland, WA: Pacific Northwest Laboratories, 1967.

SABUNDJIAN, G., FREITAS, R. L., CONTI, T. N. ***Comparação entre os códigos RELAP5/MOD1 e TRAC-PD2 na simulação da experiência CANON***, IPEN/CNEN-SP, São Paulo, SP, 1986.

SABUNDJIAN, G.; ANDRADE, D. A.; BAPTISTA FILHO, B. D.; PALMIERI, E. T. ***Preliminary Transient Analysis for the IRIS Reactor Pressurizer with RELAP5/MOD3.3 Code***. In: IMAAC 2004 - Integrated Modeling and Analysis and Applied Control and Automation, 2004, Genova. PRELIMINARY TRANSIENT ANALYSIS FOR THE IRIS REACTOR PRESSURIZER WITH RELAP5/Mod3.3 CODE, 2004.

SOFTWARE-FACTORY NORBERT SCHMITZ. ***SF Pressure Drop 6.2***. Germany, 2010. Disponível em: <http://www.pressure-drop.com/>. Acesso em: 25 jun. 2019.

SILVESTRE, LARISSA. J. B. ***PCRELAP5 – Programa de Cálculo para os Dados de Entrada do Código RELAP5***, Dissertação (mestrado), IPEN/CNEN-SP, São Paulo, SP, 2016.

SINGER, J. *JVM versus CLR: A Comparative Study*, University of Cambridge Computer Laboratory, Cambridge, UK, 2003. Disponível em: <https://pdfs.semanticscholar.org/b57a/af581e043fb63c56ebd662720190e3121220.pdf>. Acesso em: 25 jun. 2019.

SNAP version 2.4.2, 2015. Disponível em: <http://www.nrc.gov/about-nrc/regulatory/research/safetycodes.html>. Acesso em: 25 jun. 2019.

SPORE, J.W. et al. TRAC-PF1/M0D2 Code Manual, Volume 1, Theory Manual, NUREG/CR-5673, LA-12031-M, 1993.

TRACE V5.435 ***User's Manual. Volume 2: Modeling Guidelines***, Division of Safety Analysis Office of Nuclear Regulatory Research, U. S. Nuclear Regulatory Commission. Washington, DC, 2007.

TROPIC. Informação sobre o Programa TROPIC, 2004. Disponível em: <http://www.algo.be/cl/TEE-lisp/31837203264557830/>. Acesso em: 8 abr. 2019.

VELOSO, M.A., MATTOS, J.R.L. ***Parametric representation of centrifugal pump homologous curves***, Centro de Desenvolvimento da Tecnologia Nuclear (CDTN/CNEN-MG), Belo Horizonte, MG, 2015.

WESTINGHOUSE. *Non-LOCA Technology Transfer*, 2009. Disponível em: <http://www.westinghousenuclear.com/Portals/0/Operating%20Plant%20Services/Engineering/Safety%20Analysis/NS-ES-0023%20Online%20RTS%20ESFAs.pdf>. Acesso em: 25 jun. 2019.

GLOSSÁRIO

Add-in — *Add-ins* são complementos para o *MS Excel*® que fornecem comandos e características adicionais ao *MS Excel*®. Os *add-ins* podem ser fornecidos tanto pela Microsoft quanto por terceiros, e devem ser previamente instalados para seu uso.

AutoShape — *AutoShape*, ou simplesmente *shape*, é o nome dado pelo pacote *MS Office* a uma série de Figuras geométricas que podem ser desenhadas diretamente dentro de qualquer documento criado pelo pacote *MS Office*. Dentre os principais *AutoShapes* destacam-se os quadrados, círculos, linhas, setas, caixas de texto e balões de texto. Todos os *AutoShapes* são facilmente formatados em sua forma, tamanho e cores, o que torna muito fácil seu uso.

Cartão de entrada — Termo usado pelo RELAP5 para referir-se a cada uma das linhas de um arquivo de entrada de dados para o código RELAP5. Um cartão de entrada é composto de um número de identificação e uma lista de palavras (ver *words*, abaixo) separadas por espaços. O termo "cartão" remete ao tempo em que as informações para os programas dos computadores de grande porte antigos eram inseridas através de cartões perfurados. Apesar de cartões perfurados serem atualmente completamente obsoletos, o RELAP5 ainda guarda esse termo para referir-se aos seus registros de entrada.

Código — Código, nesta dissertação, irá sempre se referir a programas computacionais de simulação numérica de análise termo-hidráulica. Neste trabalho especificamente, o termo "código" refere-se na maioria das vezes ao programa computacional RELAP5, que também é chamado de código RELAP5. Apesar do termo "código" ter significado abrangente, neste trabalho esse termo estará restrito a definição mencionada. A título de exemplo: expressões como "linhas de código", portanto, foram substituídos por "linhas de programação". O próprio arquivo de entrada do RELAP5, que possui um padrão especial de formatação, também não será chamado de código.

Controle — O termo "controle" é largamente usado pela indústria de desenvolvimento de programas para referir-se a objetos de tela usados para apresentação ou inserção de dados pelo usuário. Exemplos de controles são caixas de texto, caixas de seleção múltiplas, listas e botões.

Curva de aprendizado — Do inglês *learning curve*, a curva de aprendizado, ou curva de aprendizagem, representa o efeito do esforço empregado no aprendizado de determinado processo na produtividade do mesmo. A redução da curva de aprendizado significa que determinado grupo de pessoas pode atingir uma maior

proficiência em determinado processo empregando para tal menos esforço nesse aprendizado.

Grid — *Datagrid*, ou simplesmente *grid*, é o nome dado a um controle de tela com formato de tabela. Esse tipo de controle, formado por um conjunto de linhas e colunas, lembra uma planilha de cálculo do *MS Excel*®, mas sem os recursos de fórmulas. Ao cruzamento de linhas e colunas dá-se o nome de "células" onde os dados são informados.

Hardcoded — Termo usado na programação para descrever casos onde dados de entrada de determinado programa ficam escritos diretamente dentro da programação do mesmo, ao invés de se deixar esses dados em fontes externas, tais como arquivos de parâmetros ou configuração. Em geral, são usados para dados que visam facilitar a programação em si, não sendo necessários obter esses dados de um usuário ou do ambiente onde o programa está sendo executado.

IntelliSense — *IntelliSense* é o termo geral para vários recursos embutidos nas IDEs atualmente: Listar Membros, Informações do Parâmetro, Informação Rápida e Completar Palavra. Esses recursos ajudam o programador a aprender mais sobre o programa que está usando, a manter o acompanhamento dos parâmetros que está digitando e a adicionar chamadas a métodos e propriedades pressionando apenas algumas teclas.

Interop — Interoperabilidade COM, ou *COM Interop* ou simplesmente *Interop*, é uma tecnologia incluída na CLR da plataforma .NET que permite que os objetos da plataforma .NET interajam com objetos do tipo COM, e vice-versa. A *Interop* visa fornecer acessos aos componentes COM existentes sem exigir que os componentes COM atuais sejam recompilados ou modificados. Como todo o modelo de objetos do MS Office® é disponibilizado para as linguagens de programação através de objetos COM o *Interop* permite que todos os programas que compõem o MS Office® possam ser automatizados através de linguagens da plataforma .NET da mesma forma com que o são através do VBA.

Mensagem *popup* — Mensagens *popup*, ou simplesmente *popup*, é o termo usado para referir-se a pequenas caixas de diálogo que aparecem ao longo de um programa para chamar a atenção do usuário para determinada mensagem, ou que exigem a interação do usuário para continuarem determinada rotina de processamento. Em geral, são pequenas janelas contendo alguma mensagem ao usuário, seguidas de alguns botões exigindo interações simples, tais como: sim, não, cancelar, *OK*. As mensagens *popup* bloqueiam a tela do usuário e o impedem de continuar interagindo com o *software* em uso até que uma resposta seja dada pelo usuário a essas janelas.

***Ribbon*®**	*Ribbon*® é um formato de apresentação de interface gráfica introduzido pelo MS Office 2007, onde o tradicional menu de ferramentas é mostrado através de uma barra horizontal mais larga, com ícones maiores, permitindo o acesso às funcionalidades desse menu não só através do mouse, mas também por dispositivos *touch screen*.
Superclasse	Também referida como classe mãe, superclasse é um termo usado na orientação a objeto para referir-se a um tipo de classe usado como base na herança para outras classes. As classes criadas tendo como base a superclasse, herdando assim seu comportamento e propriedades, são chamadas de subclasses, ou classes filhas.
Tooltip	*Tooltip*, ou rótulo, é o termo usado para informações que aparecem na forma de uma etiqueta suspensa quando se passa o cursor do *mouse* sobre determinado objeto na tela e, quando o cursor é movido para fora do mesmo, a etiqueta volta a ficar invisível. Recurso muito usado nos programas do *MS Office*® e também em *sites* da internet.
Word	Word, ou palavra, é o nome usado pelo RELAP5 para identificar cada um dos parâmetros solicitados por determinado cartão de entrada do RELAP5 para a configuração apropriada dos dados de entrada

APÊNDICE A - Programação do componente PIPE

```
1)   using System;
2)   using System.Collections.Generic;
3)   using System.Linq;
4)   using System.Text;
5)   using System.Threading.Tasks;
6)   using Office = Microsoft.Office.Core;
7)   using Excel = Microsoft.Office.Interop.Excel;
8)   using Microsoft.Office.Tools.Excel;
9)
10)  namespace FastLapAddin
11)  {
12)      public class CHydro : CComponent
13)      {
14)          //public new CHydroProperties PropertyCards;
15)          public int NumberOfJunctions = 0;
16)
17)          public CHydro()
18)          {
19)
20)          }
21)          public CHydro(string v_szCompNumberOrHydroType)
22)          {
23)              if (v_szCompNumberOrHydroType.Length == 3 && int.TryParse(v_szCompNumb
     erOrHydroType, out _))
24)              {//This case, this method will load existing component with this compo
     nent number (if exists).
25)                  CShtInp ocshtInp = CModel.GetShtInp();
26)
27)                  int iCardRow = ocshtInp.FindCardRow(v_szCompNumberOrHydroType, CHy
     droProperties.SZ_IDENTIFICATION_0000_CARDNAME);
28)                  if (iCardRow > -1)
29)                  {
30)                      string[] aszWords = ocshtInp.ReadCardWords(v_szCompNumberOrHyd
     roType, CHydroProperties.SZ_IDENTIFICATION_0000_CARDNAME, CConstants.SZ_INP_PREFIX
     _HYDRO, iCardRow);
31)                      string szDescr = ocshtInp.TargetSheet.Cells[iCardRow - 1, CCon
     stants.I_REF_SHT_INP_COL_CARD].Value;
32)                      string szHydroType = aszWords[2];
33)                      this.RowNumber = iCardRow;
34)                      this.ComponentNumber = aszWords[0].Left(3);
35)                      this.Name = aszWords[1];
36)                      this.Description = szDescr.Substring(szDescr.IndexOf(CConstant
     s.SZ_TXT_DESCR_SEP) + CConstants.SZ_TXT_DESCR_SEP.Length);
37)                      this.BaseShape = new CBaseShape(szHydroType);
38)                  }
39)              }
40)              else
41)              {//This case, this method will load a new CHydro, but with correspondi
     ng hydro shape.
42)                  this.BaseShape = new CBaseShape(v_szCompNumberOrHydroType);
43)                  this.PropertyCards = CHydroProperties.GetHydroProperties(v_szCompN
     umberOrHydroType);
44)              }
45)          }
46)
47)          public CHydroProperties HydroProperties
48)          {
49)              get
50)              {
51)                  CHydroProperties objRet = null;
52)                  try
53)                  {
54)                      objRet = (CHydroProperties)this.PropertyCards;
55)                  }
56)                  catch { }
57)                      return objRet;
58)              }
59)          }
60)
61)          public void Draw(string v_szCompNo, string v_szCompName, string v_szDescri
     ption = "", int v_iNumberOfVolumes = 1)
62)          {
```

```
63)                 this.ComponentNumber = v_szCompNo;
64)                 this.Name = v_szCompName;
65)                 this.Description = v_szDescription;
66)                 this.NumberOfVolumes = v_iNumberOfVolumes;
67)                 this.Draw();
68)             }
69)
70)             public Dictionary<int, string> ListVolumes(bool v_blnFirstItemAsEmpty = tr
    ue)
71)             {
72)                 Dictionary<int, string> dicRet = new Dictionary<int, string>();
73)                 if (v_blnFirstItemAsEmpty)
74)                     dicRet.Add(-1, "");
75)                 for (int i = 1; i <= this.NumberOfVolumes; i++)
76)                     dicRet.Add(i, "Volume " + i.ToString());
77)                 return dicRet;
78)             }
79)
80)             public static IList<CHydro> ListComponents(bool v_blnFirstItemAsEmpty = tr
    ue)
81)             {
82)                 IList<CHydro> laszRet = new List<CHydro>();
83)                 CShtInp oshtInp = CModel.GetShtInp();
84)                 Excel.Worksheet shtInp = oshtInp.TargetSheet;
85)                 int iCurRow = CConstants.I_REF_SHT_INP_ROW_1STROW + 1;
86)                 Excel.Range rngCur = shtInp.Cells[iCurRow, CConstants.I_REF_SHT_INP_CO
    L_CARD];
87)                 CHydro objHydro;
88)
89)                 string szVal;
90)                 string szCompNo;
91)                 string szCommentIdentif;
92)                 string szCompName;
93)                 string szCompType;
94)                 string szCompDescr;
95)                 string szQtVolJun;
96)                 int iIxDash, iIxColon;
97)
98)                 if (v_blnFirstItemAsEmpty)
99)                 {
100)                    objHydro = new CHydro();
101)                    objHydro.ComponentNumber = "";
102)                    objHydro.Name = "";
103)                    objHydro.Description = "";
104)                    objHydro.RowNumber = -1;
105)                    laszRet.Add(objHydro);
106)                }
107)
108)                while (rngCur.Text != "")
109)                {
110)                    szCompNo = "";
111)                    szCommentIdentif = "";
112)                    szCompName = "";
113)                    szCompDescr = "";
114)                    iIxDash = -1;
115)                    iIxColon = -1;
116)                    szVal = rngCur.Text;
117)                    if (szVal.Substring(3).Left(CConstants.SZ_INP_SEARCHMASK_HYDROID.L
    ength) == CConstants.SZ_INP_SEARCHMASK_HYDROID)
118)                    {//Found a Hydro Identification Card
119)                        szCommentIdentif = shtInp.Cells[iCurRow - 1, CConstants.I_REF_
    SHT_INP_COL_CARD].Text;
120)                        iIxDash = szCommentIdentif.IndexOf(CConstants.SZ_INP_DASH);
121)                        iIxColon = szCommentIdentif.IndexOf(CConstants.SZ_TXT_DESCR_SE
    P, iIxDash);
122)
123)                        szCompNo = szVal.Left(3);
124)                        szCompName = shtInp.Cells[iCurRow, CConstants.I_REF_SHT_INP_CO
    L_WORD1].Text;
125)                        szCompType = shtInp.Cells[iCurRow, CConstants.I_REF_SHT_INP_CO
    L_WORD2].Text;
126)                        szCompDescr = szCommentIdentif.Substring(iIxColon + CConstants
    .SZ_TXT_DESCR_SEP.Length);
127)
128)                        objHydro = new CHydro(szCompType);
129)                        if (objHydro.HydroProperties.NumberOfVolumes == -1)
130)                        {//Qtt of volumes is specified on card CCC0001.W1
```

```
131)                            szQtVolJun = shtInp.Cells[iCurRow + 1, CConstants.I_REF_SH
    T_INP_COL_WORD1].Text;
132)                            objHydro.NumberOfVolumes = int.Parse(szQtVolJun);
133)                            objHydro.NumberOfJunctions = objHydro.NumberOfVolumes - 1;
134)                        }
135)                        else
136)                        {
137)                            objHydro.NumberOfVolumes = objHydro.HydroProperties.Number
    OfVolumes;
138)                            objHydro.NumberOfJunctions = objHydro.HydroProperties.Numb
    erOfJunctions;
139)                        }
140)                        if (objHydro.NumberOfJunctions == -1)
141)                        {//Qtt of junctions is specified on card CCC0001.W1: volumes a
    re constant and were already filled.
142)                            szQtVolJun = shtInp.Cells[iCurRow + 2, CConstants.I_REF_SH
    T_INP_COL_WORD1].Text;
143)                            objHydro.NumberOfJunctions = int.Parse(szQtVolJun);
144)                        }
145)
146)                        objHydro.ComponentNumber = szCompNo;
147)                        objHydro.Name = szCompName;
148)                        objHydro.Description = szCompDescr;
149)                        objHydro.RowNumber = iCurRow;
150)                        laszRet.Add(objHydro);
151)                    }
152)
153)                    iCurRow++;
154)                    rngCur = shtInp.Cells[iCurRow, CConstants.I_REF_SHT_INP_COL_CARD];
155)                }
156)
157)                return laszRet;
158)            }
159)        }
160)
161)        public interface IHydroProperties
162)        {
163)            CBaseShape ShapeType { get; }
164)
165)            string Type { get; }
166)            int NumberOfVolumes { get; }
167)            int NumberOfJunctions { get; }
168)
169)            CCompProperty[] Properties { get; }
170)            CCompProperty[] RequiredProperties { get; }
171)            CCompProperty[] OptionalProperties { get; }
172)            CCompProperty[] VolumesProperties { get; }
173)            CCompProperty[] VolumesMainProperties { get; }
174)            CCompProperty[] VolumesOptionalProperties { get; }
175)            CCompProperty[] JunctionsProperties { get; }
176)            CCompProperty[] JunctionsMainProperties { get; }
177)            CCompProperty[] JunctionsOptionalProperties { get; }
178)        }
179)
180)        public class CHydroProperties : CCompProperties, IHydroProperties
181)        {
182)            public static string SZ_IDENTIFICATION_0000_CARDNAME = "0000 ";
183)            public CCompProperty   hydpHydroNumber              { get { return new CCo
    mpProperty(null, " "      , true, null, CGlobal.Translation.TXT_HYDRO_PROP_COMPNO.
    Translate()    , EnumHydroPropertyType.WordCompNumber);
                     } }
184)            public CCompProperty   hydpHydroName_0000W1         { get { return new CCo
    mpProperty(null, SZ_IDENTIFICATION_0000_CARDNAME+"W1", true, null, CGlobal.Transla
    tion.TXT_HYDRO_PROP_CCC0000_W1.Translate(), EnumHydroPropertyType.WordCompName);
                                                 } }
185)            public CCompProperty   hydpHydroType_0000W2         { get { return new CCo
    mpProperty(null, SZ_IDENTIFICATION_0000_CARDNAME+"W2", true, null, CGlobal.Transla
    tion.TXT_HYDRO_PROP_CCC0000_W2.Translate(), EnumHydroPropertyType.WordCompType);
                                                 } }
186)            public CCompProperty   hydpHydroIdentification_0000 { get { return new CCo
    mpProperty(null, SZ_IDENTIFICATION_0000_CARDNAME     , true, null, CGlobal.Transla
    tion.TXT_HYDRO_PROP_CCC0000.Translate()   , new CCompProperty[] { this.hydpHydroNa
    me_0000W1, this.hydpHydroType_0000W2 });  } }
187)            public static CHydroProperties GetHydroProperties(string v_szType)
188)            {
189)                CHydroProperties objRet;
190)                switch (v_szType)
```

```
191)                {
192)                    case CConstants.SZ_HYDRO_SNGLVOL:
193)                        objRet = new CHSnglvol();
194)                        break;
195)                    case CConstants.SZ_HYDRO_TMDPVOL:
196)                        objRet = new CHTmdpvol();
197)                        break;
198)                    case CConstants.SZ_HYDRO_SNGLJUN:
199)                        objRet = new CHSngljun();
200)                        break;
201)                    case CConstants.SZ_HYDRO_TMDPJUN:
202)                        objRet = new CHTmdpjun();
203)                        break;
204)                    case CConstants.SZ_HYDRO_PIPE:
205)                        objRet = new CHPipe();
206)                        break;
207)                    case CConstants.SZ_HYDRO_ANNULUS:
208)                        objRet = new CHAnnulus();
209)                        break;
210)                    case CConstants.SZ_HYDRO_PRIZER:
211)                        objRet = new CHPrizer();
212)                        break;
213)                    case CConstants.SZ_HYDRO_CANCHAN:
214)                        objRet = new CHCanchan();
215)                        break;
216)                    case CConstants.SZ_HYDRO_BRANCH:
217)                        objRet = new CHBranch();
218)                        break;
219)                    case CConstants.SZ_HYDRO_JETMIXER:
220)                        objRet = new CHJetmixer();
221)                        break;
222)                    case CConstants.SZ_HYDRO_ECCMIX:
223)                        objRet = new CHEccmix();
224)                        break;
225)                    case CConstants.SZ_HYDRO_VALVE:
226)                        objRet = new CHValve();
227)                        break;
228)                    case CConstants.SZ_HYDRO_PUMP:
229)                        objRet = new CHPump();
230)                        break;
231)                    default:
232)                        objRet = new CHydroProperties();
233)                        break;
234)                }
235)
236)                return objRet;
237)            }
238)
239)
240)            public virtual string Type
241)            {
242)                get
243)                {
244)                    return "";
245)                }
246)            }
247)            public virtual int NumberOfVolumes
248)            {
249)                get
250)                {
251)                    return -1;
252)                }
253)            }
254)            public virtual int NumberOfJunctions
255)            {
256)                get
257)                {
258)                    return -1;
259)                }
260)            }
261)
262)            public CHydroProperties()
263)            {
264)
265)            }
266)            public CCompProperty[] Identification
267)            {
```

```
268)                get
269)                {
270)                    CCompProperty[] aobjRet = {
271)                          hydpHydroNumber
272)                        , hydpHydroIdentification_0000
273)                    };
274)                    return aobjRet;
275)                }
276)            }
277)            public override CCompProperty[] Properties
278)            {
279)                get
280)                {
281)                    CCompProperty[] aobjRet = CHydroProperties.PackProperties( this.Id
    entification
282)                                                                                , this.R
    equiredProperties
283)                                                                                , this.O
    ptionalProperties
284)                                                                                , this.V
    olumesMainProperties
285)                                                                                , this.V
    olumesOptionalProperties
286)                                                                                , this.J
    unctionsMainProperties
287)                                                                                , this.J
    unctionsOptionalProperties
288)                                                                                , this.T
    imeDataMainProperties
289)                                                                                , this.T
    imeDataOptionalProperties);
290)                    return aobjRet;
291)                }
292)            }
293)            public CCompProperty[] VolumesProperties
294)            {
295)                get
296)                {
297)                    CCompProperty[] aobjMain = this.VolumesMainProperties;
298)                    CCompProperty[] aobjOpt = this.VolumesOptionalProperties;
299)                    CCompProperty[] aobjRet = CHydroProperties.PackProperties(aobjMain
    , aobjOpt);
300)                    return aobjRet;
301)                }
302)            }
303)            public CCompProperty[] JunctionsProperties
304)            {
305)                get
306)                {
307)                    CCompProperty[] aobjMain = this.JunctionsMainProperties;
308)                    CCompProperty[] aobjOpt = this.JunctionsOptionalProperties;
309)                    CCompProperty[] aobjRet = CHydroProperties.PackProperties(aobjMain
    , aobjOpt);
310)                    return aobjRet;
311)                }
312)            }
313)            public CCompProperty[] TimeDataProperties
314)            {
315)                get
316)                {
317)                    CCompProperty[] aobjMain = this.TimeDataMainProperties;
318)                    CCompProperty[] aobjOpt = this.TimeDataOptionalProperties;
319)                    CCompProperty[] aobjRet = CHydroProperties.PackProperties(aobjMain
    , aobjOpt);
320)                    return aobjRet;
321)                }
322)            }
323)            public virtual CCompProperty[] VolumesMainProperties
324)            {
325)                get
326)                {
327)                    return null;
328)                }
329)            }
330)            public virtual CCompProperty[] VolumesOptionalProperties
331)            {
332)                get
```

```
333)                {
334)                    return null;
335)                }
336)            }
337)            public virtual CCompProperty[] JunctionsMainProperties
338)            {
339)                get
340)                {
341)                    return null;
342)                }
343)            }
344)            public virtual CCompProperty[] JunctionsOptionalProperties
345)            {
346)                get
347)                {
348)                    return null;
349)                }
350)            }
351)            public virtual CCompProperty[] TimeDataMainProperties
352)            {
353)                get
354)                {
355)                    return null;
356)                }
357)            }
358)            public virtual CCompProperty[] TimeDataOptionalProperties
359)            {
360)                get
361)                {
362)                    return null;
363)                }
364)            }
365)
366)        }
367)
368)        public class CHSnglvol : CHydroProperties, IHydroProperties
369)        { ... }
370)        public class CHSngljun : CHydroProperties, IHydroProperties
371)        { ... }
372)        public class CHTmdpjun : CHydroProperties, IHydroProperties
373)        { ... }
374)        public class CHTmdpvol : CHydroProperties, IHydroProperties
375)        { ... }
376)        public class CHPipe : CHydroProperties, IHydroProperties
377)        {
378)            public override CBaseShape ShapeType { get { return CBaseShape.ShapeHydroP
        ipe; } }
379)            public virtual CCompProperty hydpPipe_0001_Info          { get { return th
        is.CreateCompProperty("0001 "   , true, null, CGlobal.Translation.TXT_HPIPE_PROP_CC
        C0001.Translate()        , EnumHydroPropertyType.WordInteger1to99);} }
380)            public CCompProperty hydpPipe_0199_VolAreaX              { get { return th
        is.CreateCompProperty("0199 "   , true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
        L_CCC0199.Translate()    , EnumHydroPropertyType.WordReal);         } }
381)            public CCompProperty hydpPipe_0299_OptJunArea            { get { return th
        is.CreateCompProperty("0299 "   , true, null, CGlobal.Translation.TXT_HPIPE_PROP_JU
        N_CCC0299.Translate()    , EnumHydroPropertyType.WordReal);         } }
382)            public CCompProperty hydpPipe_0399_VolLengthX            { get { return th
        is.CreateCompProperty("0399 "   , true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
        L_CCC0399.Translate()    , EnumHydroPropertyType.WordReal);         } }
383)            public CCompProperty hydpPipe_0499_OptVolVolume          { get { return th
        is.CreateCompProperty("0499 "   , true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
        L_CCC0499.Translate()    , EnumHydroPropertyType.WordReal);         } }
384)            public CCompProperty hydpPipe_0599_OptVolAzimAngle       { get { return th
        is.CreateCompProperty("0599 "   , true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
        L_CCC0599.Translate()    , EnumHydroPropertyType.WordRealAngle360);} }
385)            public CCompProperty hydpPipe_0699_VolVertAngle          { get { return th
        is.CreateCompProperty("0699 "   , true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
        L_CCC0699.Translate()    , EnumHydroPropertyType.WordRealAngle90); } }
386)            public CCompProperty hydpPipe_0799W1_OptVolElevXX        { get { return th
        is.CreateCompProperty("0799 W1", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
        L_CCC0799_W1.Translate() , EnumHydroPropertyType.WordReal);         } }
387)            public CCompProperty hydpPipe_0799W2_OptVolElevXY        { get { return th
        is.CreateCompProperty("0799 W2",false, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
        L_CCC0799_W2.Translate() , EnumHydroPropertyType.WordReal);         } }
388)            public CCompProperty hydpPipe_0799W3_OptVolElevXZ        { get { return th
        is.CreateCompProperty("0799 W3",false, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
        L_CCC0799_W3.Translate() , EnumHydroPropertyType.WordReal);         } }
```

```
389)          public CCompProperty hydpPipe_0799_OptVolElevX           { get { return th
      is.CreateCompProperty("0799 "  ,false, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC0799.Translate()    , new CCompProperty[] { this.hydpPipe_0799W1_OptVolElevXX
      , this.hydpPipe_0799W2_OptVolElevXY, this.hydpPipe_0799W3_OptVolElevXZ} ); } }
390)          public CCompProperty hydpPipe_0899W1_VolWallRoughX       { get { return th
      is.CreateCompProperty("0899 W1", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC0899_W1.Translate() , EnumHydroPropertyType.WordReal); } }
391)          public CCompProperty hydpPipe_0899W2_VolHydrDiamX        { get { return th
      is.CreateCompProperty("0899 W2", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC0899_W2.Translate() , EnumHydroPropertyType.WordReal); } }
392)          public CCompProperty hydpPipe_0899_VolFrictionX          { get { return th
      is.CreateCompProperty("0899 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC0899.Translate()    , new CCompProperty[] { this.hydpPipe_0899W1_VolWallRough
      X, this.hydpPipe_0899W2_VolHydrDiamX }); } }
393)          public CCompProperty hydpPipe_0999W1_OptJunLossCoefAf    { get { return th
      is.CreateCompProperty("0999 W1", true, null, CGlobal.Translation.TXT_HPIPE_PROP_JU
      N_CCC0999_W1.Translate() , EnumHydroPropertyType.WordReal); } }
394)          public CCompProperty hydpPipe_0999W2_OptJunLossCoefAr    { get { return th
      is.CreateCompProperty("0999 W2", true, null, CGlobal.Translation.TXT_HPIPE_PROP_JU
      N_CCC0999_W2.Translate() , EnumHydroPropertyType.WordReal); } }
395)          public CCompProperty hydpPipe_0999_OptJunLossCoef        { get { return th
      is.CreateCompProperty("0999 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_JU
      N_CCC0999.Translate()    , new CCompProperty[] { this.hydpPipe_0999W1_OptJunLossCo
      efAf, this.hydpPipe_0999W2_OptJunLossCoefAr }); } }
396)          public CCompProperty hydpPipe_1099_VolCtrlFlgX           { get { return th
      is.CreateCompProperty("1099 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC1099.Translate()    , EnumHydroPropertyType.WordTLPVBFE); } }
397)          public CCompProperty hydpPipe_1199_JunCtrlFlg            { get { return th
      is.CreateCompProperty("1199 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_JU
      N_CCC1199.Translate()    , EnumHydroPropertyType.Word0EF0CAHS);} }
398)          public CCompProperty hydpPipe_1299W1_VolIni_EbtFlg       { get { return th
      is.CreateCompProperty("1299 W1", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC1299_W1.Translate() , EnumHydroPropertyType.WordEBT); } }
399)          public CCompProperty hydpPipe_1299W2_EbtParam1           { get { return th
      is.CreateCompProperty("1299 W2", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC1299_W2.Translate() , EnumHydroPropertyType.WordReal); } }
400)          public CCompProperty hydpPipe_1299W3_EbtParam2           { get { return th
      is.CreateCompProperty("1299 W3", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC1299_W3.Translate() , EnumHydroPropertyType.WordReal); } }
401)          public CCompProperty hydpPipe_1299W4_EbtParam3           { get { return th
      is.CreateCompProperty("1299 W4", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC1299_W4.Translate() , EnumHydroPropertyType.WordReal); } }
402)          public CCompProperty hydpPipe_1299W5_EbtParam4           { get { return th
      is.CreateCompProperty("1299 W5", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC1299_W5.Translate() , EnumHydroPropertyType.WordReal); } }
403)          public CCompProperty hydpPipe_1299W6_EbtParam5           { get { return th
      is.CreateCompProperty("1299 W6", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC1299_W6.Translate() , EnumHydroPropertyType.WordReal); } }
404)          public CCompProperty hydpPipe_1299_VolIniCond            { get { return th
      is.CreateCompProperty("1299 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC1299.Translate()    , new CCompProperty[] { this.hydpPipe_1299W1_VolIni_EbtFl
      g, this.hydpPipe_1299W2_EbtParam1, this.hydpPipe_1299W3_EbtParam2, this.hydpPipe_1
      299W4_EbtParam3, this.hydpPipe_1299W5_EbtParam4, this.hydpPipe_1299W6_EbtParam5 })
      ; } }
405)          public CCompProperty hydpPipe_1300_OptJunctionCond       { get { return th
      is.CreateCompProperty("1300 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_JU
      N_CCC1300.Translate()    , EnumHydroPropertyType.WordInteger); } }
406)          public CCompProperty hydpPipe_1399W1_JunIniVeloc         { get { return th
      is.CreateCompProperty("1399 W1", true, null, CGlobal.Translation.TXT_HPIPE_PROP_JU
      N_CCC1399_W1.Translate() , EnumHydroPropertyType.WordReal); } }
407)          public CCompProperty hydpPipe_1399W2_JunIniVaporVeloc    { get { return th
      is.CreateCompProperty("1399 W2", true, null, CGlobal.Translation.TXT_HPIPE_PROP_JU
      N_CCC1399_W2.Translate() , EnumHydroPropertyType.WordReal); } }
408)          public CCompProperty hydpPipe_1399W3_JunInterfVeloc      { get { return th
      is.CreateCompProperty("1399 W3", true, null, CGlobal.Translation.TXT_HPIPE_PROP_JU
      N_CCC1399_W3.Translate() , EnumHydroPropertyType.WordRealNotImplementedZero); } }
409)          public CCompProperty hydpPipe_1399_JunIniCond            { get { return th
      is.CreateCompProperty("1399 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_JU
      N_CCC1399.Translate()    , new CCompProperty[] { this.hydpPipe_1399W1_JunIniVeloc,
       this.hydpPipe_1399W2_JunIniVaporVeloc, this.hydpPipe_1399W3_JunInterfVeloc }); }
      }
410)          public CCompProperty hydpPipe_1499W1_OptJunHydrDiam      { get { return th
      is.CreateCompProperty("1499 W1", true, null, CGlobal.Translation.TXT_HPIPE_PROP_JU
      N_CCC1499_W1.Translate() , EnumHydroPropertyType.WordReal); } }
411)          public CCompProperty hydpPipe_1499W2_OptJunCcflFloodForm { get { return th
      is.CreateCompProperty("1499 W2", true, null, CGlobal.Translation.TXT_HPIPE_PROP_JU
      N_CCC1499_W2.Translate() , EnumHydroPropertyType.WordReal0to1); } }
```

```
412)          public CCompProperty hydpPipe_1499W3_OptJunCcflGasIntcpt { get { return th
      is.CreateCompProperty("1499 W3", true, null, CGlobal.Translation.TXT_HPIPE_PROP_JU
      N_CCC1499_W3.Translate() , EnumHydroPropertyType.WordRealHigherThanZero); } }
413)          public CCompProperty hydpPipe_1499W4_OptJunCcflSlope     { get { return th
      is.CreateCompProperty("1499 W4", true, null, CGlobal.Translation.TXT_HPIPE_PROP_JU
      N_CCC1499_W4.Translate() , EnumHydroPropertyType.WordReal); } }
414)          public CCompProperty hydpPipe_1499_OptJunDiamCcfl        { get { return th
      is.CreateCompProperty("1499 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_JU
      N_CCC1499.Translate()    , new CCompProperty[] { this.hydpPipe_1499W1_OptJunHydrDi
      am , this.hydpPipe_1499W2_OptJunCcflFloodForm , this.hydpPipe_1499W3_OptJunCcflGas
      Intcpt , this.hydpPipe_1499W4_OptJunCcflSlope}); } }
415)          public CCompProperty hydpPipe_1699_OptVolAreaY           { get { return th
      is.CreateCompProperty("1699 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC1699.Translate()    , EnumHydroPropertyType.WordReal); } }
416)          public CCompProperty hydpPipe_1799_OptVolAreaZ           { get { return th
      is.CreateCompProperty("1799 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC1799.Translate()    , EnumHydroPropertyType.WordReal); } }
417)          public CCompProperty hydpPipe_1899_OptVolLengthY         { get { return th
      is.CreateCompProperty("1899 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC1899.Translate()    , EnumHydroPropertyType.WordReal); } }
418)          public CCompProperty hydpPipe_1999_OptVolLenghZ          { get { return th
      is.CreateCompProperty("1999 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC1999.Translate()    , EnumHydroPropertyType.WordReal); } }
419)          public CCompProperty hydpPipe_2099_VolBoroConc           { get { return th
      is.CreateCompProperty("2099 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2099.Translate()    , EnumHydroPropertyType.WordReal); } }
420)          public CCompProperty hydpPipe_2199W1_OptVolElevYX        { get { return th
      is.CreateCompProperty("2199 W1", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2199_W1.Translate() , EnumHydroPropertyType.WordReal); } }
421)          public CCompProperty hydpPipe_2199W2_OptVolElevYY        { get { return th
      is.CreateCompProperty("2199 W2", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2199_W2.Translate() , EnumHydroPropertyType.WordReal); } }
422)          public CCompProperty hydpPipe_2199W3_OptVolElevYZ        { get { return th
      is.CreateCompProperty("2199 W3", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2199_W3.Translate() , EnumHydroPropertyType.WordReal); } }
423)          public CCompProperty hydpPipe_2199_OptVolElevY           { get { return th
      is.CreateCompProperty("2199 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2199.Translate()    , new CCompProperty[] { this.hydpPipe_2199W1_OptVolElevYX
       , this.hydpPipe_2199W2_OptVolElevYY , this.hydpPipe_2199W3_OptVolElevYZ }); } }
424)          public CCompProperty hydpPipe_2299W1_OptVolElevZX        { get { return th
      is.CreateCompProperty("2299 W1", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2299_W1.Translate() , EnumHydroPropertyType.WordReal); } }
425)          public CCompProperty hydpPipe_2299W2_OptVolElevZY        { get { return th
      is.CreateCompProperty("2299 W2", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2299_W2.Translate() , EnumHydroPropertyType.WordReal); } }
426)          public CCompProperty hydpPipe_2299W3_OptVolElevZZ        { get { return th
      is.CreateCompProperty("2299 W3", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2299_W3.Translate() , EnumHydroPropertyType.WordReal); } }
427)          public CCompProperty hydpPipe_2299_OptVolElevZ           { get { return th
      is.CreateCompProperty("2299 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2299.Translate()    , new CCompProperty[] { this.hydpPipe_2299W1_OptVolElevZX
       , this.hydpPipe_2299W2_OptVolElevZY, this.hydpPipe_2299W3_OptVolElevZZ }); } }
428)          public CCompProperty hydpPipe_2399W1_OptVolWallRoughY    { get { return th
      is.CreateCompProperty("2399 W1", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2399_W1.Translate() , EnumHydroPropertyType.WordReal); } }
429)          public CCompProperty hydpPipe_2399W2_OptVolHydrDiamY     { get { return th
      is.CreateCompProperty("2399 W2", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2399_W2.Translate() , EnumHydroPropertyType.WordReal); } }
430)          public CCompProperty hydpPipe_2399_OptVolFrictionY       { get { return th
      is.CreateCompProperty("2399 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2399.Translate()    , new CCompProperty[] { this.hydpPipe_2399W1_OptVolWallRo
      ughY, this.hydpPipe_2399W2_OptVolHydrDiamY}); } }
431)          public CCompProperty hydpPipe_2499W1_OptVolWallRoughZ    { get { return th
      is.CreateCompProperty("2499 W1", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2499_W1.Translate() , EnumHydroPropertyType.WordReal); } }
432)          public CCompProperty hydpPipe_2499W2_OptVolHydrDiamZ     { get { return th
      is.CreateCompProperty("2499 W2", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2499_W2.Translate() , EnumHydroPropertyType.WordReal); } }
433)          public CCompProperty hydpPipe_2499_OptVolFrictionZ       { get { return th
      is.CreateCompProperty("2499 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2499.Translate()    , new CCompProperty[] { this.hydpPipe_2499W1_OptVolWallRo
      ughZ, this.hydpPipe_2499W2_OptVolHydrDiamZ}); } }
434)          public CCompProperty hydpPipe_2599W1_OptVolShpFactorX    { get { return th
      is.CreateCompProperty("2599 W1", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2599_W1.Translate() , EnumHydroPropertyType.WordReal); } }
```

```
435)          public CCompProperty hydpPipe_2599W2_OptVolViscosityExpX { get { return th
      is.CreateCompProperty("2599 W2", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2599_W2.Translate() , EnumHydroPropertyType.WordReal); } }
436)          public CCompProperty hydpPipe_2599W3_OptVolShpFactorY    { get { return th
      is.CreateCompProperty("2599 W3", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2599_W3.Translate() , EnumHydroPropertyType.WordReal); } }
437)          public CCompProperty hydpPipe_2599W4_OptVolViscosityExpY { get { return th
      is.CreateCompProperty("2599 W4", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2599_W4.Translate() , EnumHydroPropertyType.WordReal); } }
438)          public CCompProperty hydpPipe_2599W5_OptVolShpFactorZ    { get { return th
      is.CreateCompProperty("2599 W5", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2599_W5.Translate() , EnumHydroPropertyType.WordReal); } }
439)          public CCompProperty hydpPipe_2599W6_OptVolViscosityExpZ { get { return th
      is.CreateCompProperty("2599 W6", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2599_W6.Translate() , EnumHydroPropertyType.WordReal); } }
440)          public CCompProperty hydpPipe_2599_OptVolAdWallFrict     { get { return th
      is.CreateCompProperty("2599 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2599.Translate()    , new CCompProperty[] { this.hydpPipe_2599W1_OptVolShpFac
      torX, this.hydpPipe_2599W2_OptVolViscosityExpX, this.hydpPipe_2599W3_OptVolShpFact
      orY, this.hydpPipe_2599W4_OptVolViscosityExpY , this.hydpPipe_2599W5_OptVolShpFact
      orZ, this.hydpPipe_2599W6_OptVolViscosityExpZ}); } }
441)          public CCompProperty hydpPipe_2799_OptVolCtrlFlgY        { get { return th
      is.CreateCompProperty("2799 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2799.Translate()    , EnumHydroPropertyType.Word00000F0); } }
442)          public CCompProperty hydpPipe_2899_OptVolCtrlFlgZ        { get { return th
      is.CreateCompProperty("2899 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC2899.Translate()    , EnumHydroPropertyType.Word00000F0); } }
443)          public CCompProperty hydpPipe_3099W1_OptJunFormLossBf    { get { return th
      is.CreateCompProperty("3099 W1", true, null, CGlobal.Translation.TXT_HPIPE_PROP_JU
      N_CCC3099_W1.Translate() , EnumHydroPropertyType.WordReal); } }
444)          public CCompProperty hydpPipe_3099W2_OptJunFormLossCf    { get { return th
      is.CreateCompProperty("3099 W2", true, null, CGlobal.Translation.TXT_HPIPE_PROP_JU
      N_CCC3099_W2.Translate() , EnumHydroPropertyType.WordReal); } }
445)          public CCompProperty hydpPipe_3099W3_OptJunFormLossBr    { get { return th
      is.CreateCompProperty("3099 W3", true, null, CGlobal.Translation.TXT_HPIPE_PROP_JU
      N_CCC3099_W3.Translate() , EnumHydroPropertyType.WordReal); } }
446)          public CCompProperty hydpPipe_3099W4_OptJunFormLossCr    { get { return th
      is.CreateCompProperty("3099 W4", true, null, CGlobal.Translation.TXT_HPIPE_PROP_JU
      N_CCC3099_W4.Translate() , EnumHydroPropertyType.WordReal); } }
447)          public CCompProperty hydpPipe_3099_OptJunFormLoss        { get { return th
      is.CreateCompProperty("3099 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_JU
      N_CCC3099.Translate()    , new CCompProperty[] { this.hydpPipe_3099W1_OptJunFormLo
      ssBf, this.hydpPipe_3099W2_OptJunFormLossCf, this.hydpPipe_3099W3_OptJunFormLossBr
      , this.hydpPipe_3099W4_OptJunFormLossCr }); } }
448)          public CCompProperty hydpPipe_3199W1_VolOrlnGap          { get { return th
      is.CreateCompProperty("3199 W1", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC3199_W1.Translate() , EnumHydroPropertyType.WordReal); } }
449)          public CCompProperty hydpPipe_3199W2_VolOrlnSpan         { get { return th
      is.CreateCompProperty("3199 W2", true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC3199_W2.Translate() , EnumHydroPropertyType.WordReal); } }
450)          public CCompProperty hydpPipe_3199_VolOrlnAns            { get { return th
      is.CreateCompProperty("3199 "  , true, null, CGlobal.Translation.TXT_HPIPE_PROP_VO
      L_CCC3199.Translate()    , new CCompProperty[] { this.hydpPipe_3199W1_VolOrlnGap,
      this.hydpPipe_3199W2_VolOrlnSpan }); } }
451)
452)          public CHPipe() : base()
453)          { }
454)          public override string Type { get { return CConstants.SZ_HYDRO_PIPE; } }
455)          public override int NumberOfVolumes { get { return -1; } }
456)          public override int NumberOfJunctions { get { return -1; } }
457)
458)          public override CCompProperty[] RequiredProperties
459)          {
460)              get
461)              {
462)                  CCompProperty[] aobjRet = {
463)                    hydpPipe_0001_Info
464)                  };
465)                  return aobjRet;
466)              }
467)          }
468)          public override CCompProperty[] OptionalProperties
469)          {
470)              get
471)              {
472)                  CCompProperty[] aobjRet = {
473)                    hydpPipe_1300_OptJunctionCond
```

```
474)                    };
475)                    return aobjRet;
476)                }
477)            }
478)            public override CCompProperty[] VolumesMainProperties
479)            {
480)                get
481)                {
482)                    CCompProperty[] aobjRet = {
483)                      hydpPipe_0199_VolAreaX
484)                    , hydpPipe_0399_VolLengthX
485)                    , hydpPipe_0699_VolVertAngle
486)                    , hydpPipe_0899_VolFrictionX
487)                    , hydpPipe_1099_VolCtrlFlgX
488)                    , hydpPipe_1299_VolIniCond
489)                    };
490)                    return aobjRet;
491)                }
492)            }
493)            public override CCompProperty[] VolumesOptionalProperties
494)            {
495)                get
496)                {
497)                    CCompProperty[] aobjRet = {
498)                      hydpPipe_0499_OptVolVolume
499)                    , hydpPipe_0599_OptVolAzimAngle
500)                    , hydpPipe_0799_OptVolElevX
501)                    , hydpPipe_2199_OptVolElevY
502)                    , hydpPipe_2299_OptVolElevZ
503)                    , hydpPipe_1699_OptVolAreaY
504)                    , hydpPipe_1799_OptVolAreaZ
505)                    , hydpPipe_1899_OptVolLengthY
506)                    , hydpPipe_1999_OptVolLenghZ
507)                    , hydpPipe_2099_VolBoroConc
508)                    , hydpPipe_2399_OptVolFrictionY
509)                    , hydpPipe_2499_OptVolFrictionZ
510)                    , hydpPipe_2599_OptVolAdWallFrict
511)                    , hydpPipe_2799_OptVolCtrlFlgY
512)                    , hydpPipe_2899_OptVolCtrlFlgZ
513)                    , hydpPipe_3199_VolOrlnAns
514)                    };
515)                    return aobjRet;
516)                }
517)            }
518)            public override CCompProperty[] JunctionsMainProperties
519)            {
520)                get
521)                {
522)                    CCompProperty[] aobjRet = {
523)                      hydpPipe_1199_JunCtrlFlg
524)                    , hydpPipe_1399_JunIniCond
525)                    };
526)                    return aobjRet;
527)                }
528)            }
529)            public override CCompProperty[] JunctionsOptionalProperties
530)            {
531)                get
532)                {
533)                    CCompProperty[] aobjRet = {
534)                      hydpPipe_0299_OptJunArea
535)                    , hydpPipe_0999_OptJunLossCoef
536)                    , hydpPipe_1499_OptJunDiamCcfl
537)                    , hydpPipe_3099_OptJunFormLoss
538)                    };
539)                    return aobjRet;
540)                }
541)            }
542)        }
543)
544)        public class CHAnnulus : CHPipe
545)        {
546)            public override CBaseShape ShapeType { get { return CBaseShape.ShapeHydroA
    nnulus; } }
547)            public CHAnnulus() : base()
548)            { }
549)        }
```

```
550)
551)        public class CHPrizer : CHPipe
552)        {
553)            public override CBaseShape ShapeType { get { return CBaseShape.ShapeHydroP
        rizer; } }
554)            public CCompProperty hydpPrizer_0001W1_QtVols              { get { return
        this.CreateCompProperty("0001 W1", true, null, CGlobal.Translation.TXT_HPRIZER_PRO
        P_CCC0001_W1.Translate() , EnumHydroPropertyType.WordInteger1to99); } }
555)            public CCompProperty hydpPrizer_0001W2_OptVolElevXX        { get { return
        this.CreateCompProperty("0001 W2", true, null, CGlobal.Translation.TXT_HPRIZER_PRO
        P_CCC0001_W2.Translate() , EnumHydroPropertyType.WordInteger0orHigher); } }
556)            public CCompProperty hydpPrizer_0001W3_OptVolElevXY        { get { return
        this.CreateCompProperty("0001 W3",false, null, CGlobal.Translation.TXT_HPRIZER_PRO
        P_CCC0001_W3.Translate() , EnumHydroPropertyType.WordReal); } }
557)            public CCompProperty hydpPrizer_0001W4_OptVolElevXZ        { get { return
        this.CreateCompProperty("0001 W4",false, null, CGlobal.Translation.TXT_HPRIZER_PRO
        P_CCC0001_W4.Translate() , EnumHydroPropertyType.WordReal); } }
558)            public CCompProperty hydpPrizer_0001W5_OptVolElevXZ        { get { return
        this.CreateCompProperty("0001 W5",false, null, CGlobal.Translation.TXT_HPRIZER_PRO
        P_CCC0001_W5.Translate() , EnumHydroPropertyType.WordReal); } }
559)            public override CCompProperty hydpPipe_0001_Info           { get { return
        this.CreateCompProperty("0001 "  , true, null, CGlobal.Translation.TXT_HPRIZER_PRO
        P_CCC0001.Translate()    , new CCompProperty[] { this.hydpPrizer_0001W1_QtVols, th
        is.hydpPrizer_0001W2_OptVolElevXX, this.hydpPrizer_0001W3_OptVolElevXY, this.hydpP
        rizer_0001W4_OptVolElevXZ, this.hydpPrizer_0001W5_OptVolElevXZ} ); } }
560)
561)            public CHPrizer() : base()
562)            { }
563)        }
564)
565)        public class CHCanchan : CHPrizer
566)        {
567)            public override CBaseShape ShapeType { get { return CBaseShape.ShapeHydroC
        anchan; } }
568)            public override CCompProperty hydpPipe_0001_Info           { get { return
        new CCompProperty(this.ShapeType, "0001 "  , true, null, CGlobal.Translation.TXT_H
        CANCHAN_PROP_CCC0001.Translate()    , new CCompProperty[] { this.hydpPrizer_0001W1
        _QtVols, this.hydpPrizer_0001W2_OptVolElevXX, this.hydpPrizer_0001W3_OptVolElevXY,
         this.hydpPrizer_0001W4_OptVolElevXZ} );  } }
569)
570)            public CHCanchan() : base()
571)            { }
572)        }
573)        public class CHBranch : CHydroProperties, IHydroProperties
574)        { ... }
575)        public class CHJetmixer : CHydroProperties, IHydroProperties
576)        { ... }
577)        public class CHValve : CHydroProperties, IHydroProperties
578)        { ... }
579)        public class CHPump : CHydroProperties, IHydroProperties
580)        { ... }
581)        public class CEbtTflag
582)        { ... }
583)        public interface IHydroSelected
584)        {
585)            string ComponentNumber { get; }
586)            string InnerElement { get; }
587)            int VolumeFace { get; }
588)        }
589)
590)        public class CHydroSelectedVolume : IHydroSelected
591)        {
592)            public string ComponentNumber { get; set; }
593)            public string ComponentName { get; set; }
594)            public string InnerElement { get { return this.InnerVolume; } }
595)            public string InnerVolume { get; set; }
596)            public int VolumeFace { get; }
597)            public string ShapeVolumeName
598)            {
599)                get
600)                {
601)                    string szRet = "";
602)                    if(this.InnerVolume != "")
603)                        szRet = this.ComponentNumber + this.ComponentName + CConstants
        .SZ_HYDRO_VOLUME_PREFIX + this.InnerVolume;
604)                    else
605)                        szRet = this.ComponentNumber + this.ComponentName;
```

```
606)                    return szRet;
607)                }
608)            }
609)            public int ShapeVolumeFace
610)            {
611)                get
612)                {
613)                    int iRet=2;
614)                    if (this.VolumeFace == 1 || this.VolumeFace == 5) //Inlet on 1st o
        r 3rd axis
615)                        iRet = 2; //Left face
616)                    else if (this.VolumeFace == 2 || this.VolumeFace == 6) //Outlet on
         1st or 3rd axis
617)                        iRet = 4; //Right face
618)                    else if (this.VolumeFace == 3) //Inlet on 2nd axis
619)                        iRet = 3; //Right face
620)                    else if (this.VolumeFace == 4) //Outlet on 2nd axis
621)                        iRet = 1; //Right face
622)
623)                    return iRet;
624)                }
625)            }
626)
627)            public CHydroSelectedVolume()
628)            {
629)
630)            }
631)            public CHydroSelectedVolume(string v_iComponentNumber, string v_szComponen
        tName, string v_szInnerVolume, int v_iVolumeFace)
632)            {
633)                this.ComponentNumber = v_iComponentNumber;
634)                this.ComponentName = v_szComponentName;
635)                this.InnerVolume = v_szInnerVolume;
636)                this.VolumeFace = v_iVolumeFace;
637)            }
638)            public string GetHydroRefVolume()
639)            {
640)                string szRet = "";
641)                if(this.ComponentNumber != "")
642)                    szRet = this.ComponentNumber.ToString().PadLeft(3, '0') + this.Inn
        erVolume.ToString().PadLeft(2, '0') + "0000";
643)                return szRet;
644)            }
645)            public string GetHydroRefVolumeAndFace()
646)            {
647)                return   this.ComponentNumber.ToString().PadLeft(3, '0')
648)                       + this.InnerVolume.ToString().PadLeft(2, '0')
649)                       + "000"
650)                       + this.VolumeFace.ToString();
651)            }
652)        }
653)    }
```

APÊNDICE B - Programação da tela do componente PIPE

```
1)   using System;
2)   using System.Collections.Generic;
3)   using System.ComponentModel;
4)   using System.Data;
5)   using System.Drawing;
6)   using System.Linq;
7)   using System.Text;
8)   using System.Threading.Tasks;
9)   using System.Windows.Forms;
10)  using System.Text.RegularExpressions;
11)  using Excel = Microsoft.Office.Interop.Excel;
12)  using Office = Microsoft.Office.Core;
13)  using Microsoft.Office.Tools.Excel;
14)  using Microsoft.VisualBasic;
15)  using Microsoft.VisualBasic.CompilerServices;
16)  using MsForms = Microsoft.Vbe.Interop.Forms;
17)
18)  namespace FastLapAddin
19)  {
20)      public partial class FrmHPipe : FrmH
21)      {
22)          protected bool m_blnJuncIdBeingValidated = false;
23)          protected bool m_blnVolIdBeingValidated = false;
24)
25)          public override int NumberOfVolumes
26)          {
27)              get
28)              {
29)                  int iRet = 1;
30)                  try
31)                  {
32)                      iRet = int.Parse(this.mskQtVol.Text);
33)                  }
34)                  catch { }
35)                  return iRet;
36)              }
37)          }
38)
39)          public FrmHPipe()
40)          {
41)              InitializeComponent();
42)              HydroComponent = new CHydro(CConstants.SZ_HYDRO_PIPE);
43)          }
44)          public FrmHPipe(string v_szCompType)
45)          {
46)              InitializeComponent();
47)              if(v_szCompType == CConstants.SZ_HYDRO_ANNULUS || v_szCompType == CCon
     stants.SZ_HYDRO_PRIZER ||  v_szCompType == CConstants.SZ_HYDRO_CANCHAN)
48)                  HydroComponent = new CHydro(v_szCompType);
49)              else
50)                  HydroComponent = new CHydro(CConstants.SZ_HYDRO_PIPE);
51)          }
52)
53)          private void FrmHPipe_Load(object sender, EventArgs e)
54)          {
55)              if(!this.DesignMode)
56)              {
57)                  //Check which type of Hydro component has been initialized using t
     his form, to change Icon and other fields
58)                  switch (this.HydroComponent.BaseShape.mszType)
59)                  {
60)                      case CConstants.SZ_HYDRO_CANCHAN:
61)                          pnlInfoCanchan.Visible = true;
62)                          pnlInfoPrizer.Visible = false;
63)                          this.PictureIcon = Properties.Resources.resIcoHCanchan;
64)                          this.PdfNamedReference = CGlobal.Translation.SZ_PDF_HOOK_H
     YDRO_CANCHAN;
65)                          break;
66)                      case CConstants.SZ_HYDRO_PRIZER:
67)                          pnlInfoCanchan.Visible = true;
68)                          pnlInfoPrizer.Visible = true;
69)                          this.PictureIcon = Properties.Resources.resIcoHPrizer;
```

```
70)                         this.PdfNamedReference = CGlobal.Translation.SZ_PDF_HOOK_H
    YDRO_PRIZER;
71)                         break;
72)                     case CConstants.SZ_HYDRO_ANNULUS:
73)                         this.PictureIcon = Properties.Resources.resIcoHAnnulus;
74)                         this.PdfNamedReference = CGlobal.Translation.SZ_PDF_HOOK_H
    YDRO_ANNULUS;
75)                         break;
76)                     default:
77)                         this.PictureIcon = Properties.Resources.resIcoHPipe;
78)                         this.PdfNamedReference = CGlobal.Translation.SZ_PDF_HOOK_H
    YDRO_PIPE;
79)                         break;
80)                 }
81)
82)                 CUdfGrid.AttachSelectionOnDataGrids(dgrVolPropSel, dgrVolPropVal,
    pnlShowHideVolProp
83)                     , this.HydroComponent.HydroProperties.VolumesMainProperties
84)                     , this.HydroComponent.HydroProperties.VolumesOptionalPropertie
    s);
85)                 CUdfGrid.AttachSelectionOnDataGrids(dgrJunPropSel, dgrJunPropVal,
    pnlShowHideJunProp
86)                     , this.HydroComponent.HydroProperties.JunctionsMainProperties
87)                     , this.HydroComponent.HydroProperties.JunctionsOptionalPropert
    ies);
88)
89)
90)                 //Set fields initial values
91)
92)
93)                 this.Base_Load(sender, e);
94)             }
95)         }
96)         public new CHPipe HydroProperties
97)         {
98)             get
99)             {
100)                return (CHPipe)this.PipeProperties;
101)            }
102)        }
103)        public CHPipe PipeProperties
104)        {
105)            get
106)            {
107)                return (CHPipe)this.HydroComponent.HydroProperties;
108)            }
109)        }
110)        public CHPipe AnnulusProperties
111)        {
112)            get
113)            {
114)                return (CHAnnulus)this.HydroComponent.HydroProperties;
115)            }
116)        }
117)        public CHPipe CanchanProperties
118)        {
119)            get
120)            {
121)                return (CHCanchan)this.HydroComponent.HydroProperties;
122)            }
123)        }
124)        public CHPipe PrizerProperties
125)        {
126)            get
127)            {
128)                return (CHPrizer)this.HydroComponent.HydroProperties;
129)            }
130)        }
131)        protected override void AssociateTags()
132)        {//Code to attach right Cards onto form controls
133)            mskQtVol.Tag     = this.HydroProperties.hydpPipe_0001_Info;
134)        }
135)        protected override void TranslateThis()
136)        {//Code to attach the right Translated messages to form labels.
137)
138)        }
139)
```

```
140)          protected void ValidateControl(object sender, CancelEventArgs e)
141)          {
142)              Control objSender = (Control)sender;
143)              bool blnRet = false;
144)              string szMsg = "";
145)
146)
147)              if (objSender == mskCompNo)
148)              {
149)                  szMsg = CGlobal.Translation.TXT_ERR_COMP_NUMBER.Translate();
150)                  int iNo = 0;
151)                  if (mskCompNo.Text != "" && int.TryParse(mskCompNo.Text, out iNo))
152)                  {
153)                      blnRet = (iNo > 0 && iNo < 1000);
154)                      if (blnRet)
155)                      {
156)                          mskCompNo.Text = iNo.ToString("000");
157)                      }
158)                  }
159)              }
160)              else if (objSender == mskCompName)
161)              {
162)                  szMsg = CGlobal.Translation.TXT_ERR_COMP_NAME.Translate();
163)                  blnRet = (mskCompName.Text != "");
164)              }
165)              else
166)                  blnRet = true;
167)              //Visually Reflecting Form Validation on controls
168)              CUdf.ReflectValidationOnControl(blnRet, objSender, szMsg, tipMsg, ref
    p_szInvalidFields, this.btnAdd);
169)
170)              e.Cancel = !blnRet;
171)          }
172)
173)          public override void UpdateInp()
174)          {
175)              string szCompNo = this.HydroComponent.ComponentNumber;
176)              string szCompName = this.HydroComponent.Name;
177)              CShtInp ocshtInp = CModel.GetShtInp();
178)
179)              ocshtInp.WriteCard("", szCompNo, szCompName, this.HydroProperties.hydp
    Pipe_0001_Info, this.mskQtVol.Text
180)                  , this.txtInfoSurgeline.Text, txtInfoTrCoefLiquid.Text, txtInfoTrC
    oefVapor.Text //These are used only in Pressurizer or Canchan; otherwise they are
    blank.
181)                  , txtInfoSprayDrop.Text); //This is used only in Pressurizer; othe
    rwise it is blank.
182)
183)              //Write Volume Cards
184)              ocshtInp.WriteCardsFromGridWhenColAreCards(  "*", szCompNo, szCompName
185)                                      , this.HydroComponent.HydroProperties.VolumesP
    roperties, dgrVolPropVal);
186)              //Write Junction Cards
187)              ocshtInp.WriteCardsFromGridWhenColAreCards( "*", szCompNo, szCompName
188)                                      , this.HydroComponent.HydroProperties.Junction
    sProperties, dgrJunPropVal);
189)          }
190)
191)
192)          protected void pnlShowHideProp_MouseDown(object sender, MouseEventArgs e)
193)          {
194)              Panel pnlPropSel = (tabAttributes.SelectedTab == tbpgVols) ? pnlShowHi
    deVolProp : pnlShowHideJunProp;
195)              btnCalc.Visible = false;
196)              if (e.Button == MouseButtons.Left)
197)              {
198)                  this.m_lDragStart = pnlPropSel.Left;
199)                  this.m_lDragStartX = e.X;
200)              }
201)          }
202)
203)          protected void pnlShowHideProp_MouseUp(object sender, MouseEventArgs e)
204)          {
205)              Panel pnlPropSel = (tabAttributes.SelectedTab == tbpgVols) ? pnlShowHi
    deVolProp : pnlShowHideJunProp;
206)              DataGridView dgrPropSel = (DataGridView)pnlPropSel.Tag;
207)              DataGridView dgrPropVal = (DataGridView)dgrPropSel.Tag;
```

```
208)                Label lblTopArrow;
209)                Label lblBottomArrow;
210)                if(pnlPropSel == pnlShowHideVolProp)
211)                {
212)                    lblTopArrow = lblPropArrowTopVol;
213)                    lblBottomArrow = lblPropArrowBottomVol;
214)                }
215)                else
216)                {
217)                    lblTopArrow = lblPropArrowTopJun;
218)                    lblBottomArrow = lblPropArrowBottomJun;
219)                }
220)
221)                if (!this.m_blnDragCapturing)
222)                {//Drag has not started, so hadle this as a regular Click event.
223)                    this.m_blnDragCapturing = false;
224)                    if (lblTopArrow.Text == CConstants.SZ_TXT_ARROW_RIGHT)
225)                    {
226)                        dgrPropSel.Width = (this.m_iDragLastSpltrPos != 0) ? this.m_iD
    ragLastSpltrPos : dgrPropSel.Parent.Width/4;
227)                        lblTopArrow.Text = CConstants.SZ_TXT_ARROW_LEFT;
228)                        lblBottomArrow.Text = CConstants.SZ_TXT_ARROW_LEFT;
229)                    }
230)                    else
231)                    {
232)                        dgrPropSel.Width = 2;
233)                        lblTopArrow.Text = CConstants.SZ_TXT_ARROW_RIGHT;
234)                        lblBottomArrow.Text = CConstants.SZ_TXT_ARROW_RIGHT;
235)                    }
236)                    dgrPropSel.Columns[1].Width = dgrPropSel.Width;
237)                    pnlPropSel.Left = dgrPropSel.Left + dgrPropSel.Width;
238)                    dgrPropVal.Left = pnlPropSel.Left + pnlPropSel.Width + 2;
239)                    dgrPropVal.Width = dgrPropVal.Parent.Width - dgrPropVal.Left - 8;
240)
241)                }
242)                this.m_blnDragCapturing = false;
243)                this.m_lDragStartX = 0;
244)            }
245)
246)
247)            protected void pnlShowHideProp_MouseMove(object sender, MouseEventArgs e)
248)            {
249)                Panel pnlPropSel;
250)                Label lblTopArrow;
251)                Label lblBottomArrow;
252)                if(tabAttributes.SelectedTab == tbpgVols)
253)                {
254)                    pnlPropSel = pnlShowHideVolProp;
255)                    lblTopArrow =lblPropArrowTopVol;
256)                    lblBottomArrow =lblPropArrowBottomVol;
257)                }
258)                else
259)                {
260)                    pnlPropSel = pnlShowHideJunProp;
261)                    lblTopArrow =lblPropArrowTopJun;
262)                    lblBottomArrow =lblPropArrowBottomJun;
263)                }
264)                DataGridView dgrPropSel = (DataGridView)pnlPropSel.Tag;
265)                DataGridView dgrPropVal = (DataGridView)dgrPropSel.Tag;
266)                if (this.m_blnDragCapturing)
267)                {
268)                    dgrPropSel.Width = e.X + pnlPropSel.Left - dgrPropSel.Left;
269)                    if (dgrPropSel.Width > 2)
270)                    {
271)                        lblTopArrow.Text = CConstants.SZ_TXT_ARROW_LEFT;
272)                        lblBottomArrow.Text = CConstants.SZ_TXT_ARROW_LEFT;
273)                    }
274)                    dgrPropSel.Columns[1].Width = dgrPropSel.Width;
275)                    pnlPropSel.Left = dgrPropSel.Left + dgrPropSel.Width;
276)                    dgrPropVal.Left = pnlPropSel.Left + pnlPropSel.Width + 2;
277)                    dgrPropVal.Width = dgrPropVal.Parent.Width - dgrPropVal.Left - 8;
278)                }
279)                else
280)                {
281)                    if (this.m_lDragStartX > 0)
282)                    {
283)                        if (Math.Abs(e.X - this.m_lDragStartX) >= 10)
```

```
284)                         {
285)                             this.m_blnDragCapturing = true;
286)                         }
287)                     }
288)                 }
289)             }
290)
291)
292)             protected void dgrPropSel_CurrentCellDirtyStateChanged(object sender, Even
    tArgs e)
293)             {
294)                 DataGridView dgrSender = (DataGridView)sender;
295)                 DataGridView dgrPropVal = (DataGridView)dgrSender.Tag;
296)                 if (dgrSender.IsCurrentCellDirty && dgrSender.CurrentCell is DataGridV
    iewCheckBoxCell)
297)                 {
298)                     if (dgrSender.CurrentCell.ColumnIndex == 0)
299)                     {
300)                         CCompProperty[] aobjProp = (tabAttributes.SelectedTab == tbpgV
    ols)
301)                                                         ?this.HydroComponent.HydroProper
    ties.VolumesOptionalProperties
302)                                                         :this.HydroComponent.HydroProper
    ties.JunctionsOptionalProperties;
303)                         if (Convert.ToBoolean(dgrSender.CurrentCell.EditedFormattedVal
    ue))
304)                         {
305)                             string szCard = aobjProp[dgrSender.CurrentCell.RowIndex].s
    zCardName;
306)                             CUdfGrid.AddPropertyOnGrid(dgrPropVal, this.HydroComponent
    .HydroProperties.GetPropertyByCardName(szCard));
307)                             dgrSender.CommitEdit(DataGridViewDataErrorContexts.Commit)
    ;
308)                         }
309)                         else
310)                         {
311)                             if (CUdfGrid.RemovePropertyFromGrid(dgrPropVal, aobjProp[d
    grSender.CurrentCell.RowIndex].szCardName))
312)                             {
313)                                 dgrSender.CommitEdit(DataGridViewDataErrorContexts.Com
    mit);
314)                             }
315)                             else
316)                                 dgrSender.CancelEdit();
317)                         }
318)                     }
319)                 }
320)             }
321)
322)             protected void dgrVolPropVal_CellValidating(object sender, DataGridViewCel
    lValidatingEventArgs e)
323)             {
324)                 DataGridView dgrSender = (DataGridView)sender;
325)                 string szMsg = "";
326)                 string szMsgRow = "";
327)                 bool blnRet = true;
328)
329)                 if (e.ColumnIndex == 0)
330)                 {//Volume Number
331)                     int iQtVol = 0;
332)                     string szVol = "";
333)                     szMsg = CGlobal.Translation.TXT_ERR_PIPE_VOL_NO.Translate();
334)                     blnRet = (e.FormattedValue != null);
335)                     if (blnRet)
336)                     {
337)                         szVol = e.FormattedValue.ToString();
338)                         blnRet = (szVol != "" && int.TryParse(szVol, out iQtVol));
339)                         if (blnRet)
340)                         {
341)                             blnRet = (iQtVol > 0 && iQtVol < 100);
342)                             if (blnRet)
343)                             {
344)                                 //Check if entered value for Volume is repeated
345)                                 for (int i = 0; i < dgrSender.RowCount; i++)
346)                                 {
347)                                     if (i != e.RowIndex)
348)                                     {
```

```
349)                                        int iVal = 0;
350)                                        int.TryParse(dgrSender[0, i].FormattedValue.To
      String(), out iVal);
351)                                        if (iVal == iQtVol)
352)                                        {
353)                                            blnRet = false;
354)                                            break;
355)                                        }
356)                                    }
357)                                }
358)                            }
359)                        }
360)                    }
361)                }
362)                else
363)                {//Validate properties according with its Type
364)                    if (e.FormattedValue.ToString() != "=")
365)                        blnRet = this.HydroComponent.HydroProperties.IsValidWord(e.For
      mattedValue.ToString(), dgrSender.Columns[e.ColumnIndex].Tag.ToString(), out szMsg
      );
366)                }
367)
368)                if ((e.ColumnIndex > 0) && (dgrSender[0 ,e.RowIndex].FormattedValue.To
      String() == mskQtVol.Text))
369)                {//Last volume cannot have its values set as default ("="), once defau
      lt values are set by following cards of the same type
370)                    string szVal = e.FormattedValue.ToString();
371)                    if (szVal == CConstants.SZ_DEFAULT_VAL)
372)                    {
373)                        szMsg = CGlobal.Translation.TXT_ERR_NO_DEFAULT_ON_LASTVOL.Tran
      slate();
374)                        blnRet = false;
375)                    }
376)                }
377)
378)                //Visually Reflecting Form Validation on controls
379)                CUdf.ReflectValidationOnGridCell(blnRet, dgrSender.CurrentCell, szMsg,
       ref this.p_szInvalidFields, this.btnAdd);
380)                if (this.p_szInvalidFields.IndexOf(dgrSender.Name)>=0)
381)                {//If theres any cell invalid on volume grid, show ErrPic.
382)                    picErrVol.Visible = true;
383)                    tipMsg.SetToolTip(picErrVol, CGlobal.Translation.TXT_ERR_PIPE_VOL_
      INVALID.Translate());
384)                }
385)                else
386)                {
387)                    picErrVol.Visible = false;
388)                    tipMsg.SetToolTip(picErrVol, null);
389)                }
390)                if (szMsgRow != "")
391)                    dgrSender.CurrentRow.ErrorText = szMsgRow;
392)
393)                e.Cancel = false;
394)            }
395)            protected void dgrVolPropVal_CellValidated(object sender, DataGridViewCell
      EventArgs e)
396)            {
397)                DataGridView dgrCur = (DataGridView)sender;
398)                DataGridViewCell celCur = dgrCur[e.ColumnIndex, e.RowIndex];
399)                switch (e.ColumnIndex)
400)                {
401)                    case 0:
402)                        if (!m_blnVolIdBeingValidated)
403)                        {
404)                            m_blnVolIdBeingValidated = true; //This will avoid this ev
      ent being recursivelly infinitely called, when changing Junction ID
405)                            int iLstMaxVolOld = GetGridRow(dgrCur, mskQtVol.Text);
406)                            if (iLstMaxVolOld == -
      1 && (mskQtVol.Text != "00" && mskQtVol.Text != "")) //If there is no MAX volume,
      so I might changed exactly who was the current MAX
407)                                iLstMaxVolOld = e.RowIndex;
408)                            UpdateNumberOfVolumes();
409)                            int iLstMaxVolCur = GetGridRow(dgrCur, mskQtVol.Text);
410)
411)                            if (iLstMaxVolOld != iLstMaxVolCur)
412)                            {
413)                                if (iLstMaxVolOld >= 0)
```

```
414)                                {
415)                                    for (int i = dgrCur.ColumnCount - 1; i > 0; i--)
416)                                        dgrCur.CurrentCell = dgrCur[i, iLstMaxVolOld];
417)                                }
418)                                if (iLstMaxVolCur >= 0)
419)                                {
420)                                    for (int i = dgrCur.ColumnCount - 1; i > 0; i--)
421)                                        dgrCur.CurrentCell = dgrCur[i, iLstMaxVolCur];
422)                                    dgrCur.CurrentCell = dgrCur[e.ColumnIndex, e.RowIn
    dex];
423)                                }
424)                            }
425)                            if (dgrCur.CurrentCell.FormattedValue != null)
426)                            {
427)                                int iVolNo = -1;
428)                                if (int.TryParse(dgrCur.CurrentCell.FormattedValue.ToS
    tring(), out iVolNo))
429)                                    dgrCur.CurrentCell.Value = iVolNo.ToString(CConsta
    nts.SZ_MSK_VOL_NO);
430)                            }
431)                            m_blnVolIdBeingValidated = false; //Now this event can be
    called again
432)                        }
433)                        break;
434)                    default:
435)                        break;
436)                }
437)            }
438)            protected void dgrJunPropVal_CellValidating(object sender, DataGridViewCel
    lValidatingEventArgs e)
439)            {
440)                DataGridView dgrSender = (DataGridView)sender;
441)                string szMsg = "";
442)                string szMsgRow = "";
443)                bool blnRet = true;
444)
445)                if (e.ColumnIndex == 0)
446)                {//Junction Number
447)                    int iQtVol = 0;
448)                    int.TryParse(mskQtVol.Text, out iQtVol);
449)                    int iQtJun = 0;
450)                    string szJun = "";
451)                    szMsg = CGlobal.Translation.TXT_ERR_PIPE_JUN_NO.Translate();
452)                    blnRet = (e.FormattedValue != null);
453)                    if (blnRet)
454)                    {
455)                        szJun = e.FormattedValue.ToString();
456)                        blnRet = (szJun != "" && int.TryParse(szJun, out iQtJun));
457)                        if (blnRet)
458)                        {
459)                            blnRet = (iQtJun > 0 && iQtJun < iQtVol);
460)                            if (blnRet)
461)                            {
462)                                //Check if entered value for Junction is repeated
463)                                for (int i = 0; i < dgrSender.RowCount; i++)
464)                                {
465)                                    if (i != e.RowIndex)
466)                                    {
467)                                        int iVal = 0;
468)                                        int.TryParse(dgrSender[0, i].FormattedValue.To
    String(), out iVal);
469)                                        if (iVal == iQtJun)
470)                                        {
471)                                            blnRet = false;
472)                                            break;
473)                                        }
474)                                    }
475)                                }
476)                                //Check if entered value for junction is Higher
477)                                if(e.RowIndex == GetHigherJuntionRow())
478)                                {//Higher junction must be equal Number of Volumes min
    us 1
479)                                    szMsg = CGlobal.Translation.TXT_ERR_PIPE_JUN_LST.T
    ranslate();
480)                                    blnRet = iQtJun == (GetMaxVolume() - 1);
481)                                }
482)                            }
```

```
483)                     }
484)                 }
485)             }
486)             else
487)             {//Validate properties according with its Type
488)                 if (e.FormattedValue.ToString() != "=")
489)                     blnRet = this.HydroComponent.HydroProperties.IsValidWord(e.For
    mattedValue.ToString(), dgrSender.Columns[e.ColumnIndex].Tag.ToString(), out szMsg
    );
490)             }
491) 
492)             if (e.ColumnIndex > 0)
493)             {
494)                 int iCurJun = 0;
495)                 int iMaxJun = GetMaxVolume() - 1;
496)                 int.TryParse(dgrSender[0, e.RowIndex].FormattedValue.ToString(), o
    ut iCurJun);
497)                 if(iCurJun == iMaxJun)
498)                 {//Last junction cannot have its values set as default ("="), once
     default values are set by following cards of the same type
499)                     string szVal = e.FormattedValue.ToString();
500)                     if (szVal == CConstants.SZ_DEFAULT_VAL)
501)                     {
502)                         szMsg = CGlobal.Translation.TXT_ERR_NO_DEFAULT_ON_LASTJUN.
    Translate();
503)                         blnRet = false;
504)                     }
505)                 }
506)             }
507) 
508)             //Visually Reflecting Form Validation on controls
509)             CUdf.ReflectValidationOnGridCell(blnRet, dgrSender.CurrentCell, szMsg,
     ref this.p_szInvalidFields, this.btnAdd);
510)             if (this.p_szInvalidFields.IndexOf(dgrSender.Name)>=0)
511)             {//If theres any cell invalid on volume grid, show ErrPic.
512)                 picErrJun.Visible = true;
513)                 tipMsg.SetToolTip(picErrJun, CGlobal.Translation.TXT_ERR_PIPE_VOL_
    INVALID.Translate());
514)             }
515)             else
516)             {
517)                 picErrJun.Visible = false;
518)                 tipMsg.SetToolTip(picErrJun, null);
519)             }
520)             if (szMsgRow != "")
521)                 dgrSender.CurrentRow.ErrorText = szMsgRow;
522)             //Do not cancel validation event: User must be able to fix errors when
     HE/SHE decides to do so.
523)             e.Cancel = false;
524)         }
525)         protected void dgrJunPropVal_CellValidated(object sender, DataGridViewCell
    EventArgs e)
526)         {
527)             DataGridView dgrCur = (DataGridView)sender;
528)             DataGridViewCell celCur = dgrCur[e.ColumnIndex, e.RowIndex];
529)             switch (e.ColumnIndex)
530)             {
531)                 case 0:
532)                     if (!m_blnJuncIdBeingValidated)
533)                     {
534)                         m_blnJuncIdBeingValidated = true; //This will avoid this e
    vent being recursivelly infinitely called, when changing Junction ID
535)                         //Loop thru all columns of current junction, so all proper
    ties for last junction will be properly validated
536)                         for (int i = dgrCur.ColumnCount - 1; i > 0; i--)
537)                             dgrCur.CurrentCell = dgrCur[i, e.RowIndex];
538)                         dgrCur.CurrentCell = dgrCur[e.ColumnIndex, e.RowIndex];
539) 
540)                         if (dgrCur.CurrentCell.FormattedValue != null)
541)                         {
542)                             int iJunNo = -1;
543)                             if (int.TryParse(dgrCur.CurrentCell.FormattedValue.ToS
    tring(), out iJunNo))
544)                                 dgrCur.CurrentCell.Value = iJunNo.ToString(CConsta
    nts.SZ_MSK_VOL_NO);
545)                         }
```

```
546)                         m_blnJuncIdBeingValidated = false; //Now this event can be
       called again
547)                     }
548)                     break;
549)                 default:
550)                     break;
551)             }
552)         }
553)         protected void UpdateNumberOfVolumes()
554)         {
555)             //Update Number Of Volumes
556)             int iMaxVol = GetMaxVolume();
557)             mskQtVol.Text = iMaxVol.ToString("00");
558)
559)             //Visually Reflecting Form Validation on controls
560)             string szMsg = CGlobal.Translation.TXT_ERR_PIPE_VOL_MIN2.Translate();
561)             bool blnValidVol = (iMaxVol > 1);
562)             CUdf.ReflectValidationOnControl(blnValidVol, mskQtVol, szMsg, tipMsg,
     ref p_szInvalidFields, this.btnAdd);
563)             mskQtVol.Enabled = !blnValidVol; //If it is invalid, enable control in
       order to allow ToolIip works properly.
564)         }
565)         protected int GetMaxVolume()
566)         {
567)             int iMaxVol = 0;
568)             for (int i = 0; i < dgrVolPropVal.RowCount; i++)
569)             {
570)                 if (dgrVolPropVal[0, i].FormattedValue != null)
571)                 {
572)                     int iCurVol = -1;
573)                     int.TryParse(dgrVolPropVal[0, i].FormattedValue.ToString(), ou
     t iCurVol);
574)                     if (iMaxVol < iCurVol)
575)                         iMaxVol = iCurVol;
576)                 }
577)             }
578)             return iMaxVol;
579)         }
580)         protected int GetMaxRowId(DataGridView v_dgr, int v_iColId)
581)         {
582)             int iRet = 0;
583)             int iMaxVol = 0;
584)             for (int i = 0; i < v_dgr.RowCount; i++)
585)             {
586)                 if (v_dgr[v_iColId, i].FormattedValue != null)
587)                 {
588)                     int iCurVol = -1;
589)                     int.TryParse(v_dgr[v_iColId, i].FormattedValue.ToString(), out
       iCurVol);
590)                     if (iMaxVol < iCurVol)
591)                     {
592)                         iMaxVol = iCurVol;
593)                         iRet = i;
594)                     }
595)                 }
596)             }
597)             return iRet;
598)         }
599)         protected int GetHigherJuntionRow()
600)         {
601)             return GetMaxRowId(dgrJunPropVal, 0);
602)         }
603)         protected int GetMaxVolumeRow()
604)         {
605)             return GetMaxRowId(dgrVolPropVal, 0);
606)         }
607)         protected int GetGridRow(DataGridView v_dgr, int v_iId)
608)         {
609)             int iRet = -1;
610)             for (int i = 0; i < v_dgr.RowCount; i++)
611)             {
612)                 if (v_dgr[0, i].FormattedValue != null)
613)                 {
614)                     int iCurId = -1;
615)                     int.TryParse(v_dgr[0, i].FormattedValue.ToString(), out iCurId
     );
616)                     if (v_iId == iCurId)
```

```
617)                         {
618)                             iRet = i;
619)                             break;
620)                         }
621)                     }
622)                 }
623)                 return iRet;
624)             }
625)             protected int GetGridRow(DataGridView v_dgr, string v_szId)
626)             {
627)                 int iId = -1;
628)                 int iRet = -1;
629)                 if (int.TryParse(v_szId, out iId))
630)                     iRet = GetGridRow(v_dgr, iId);
631)                 return iRet;
632)             }
633) 
634) 
635)             //Drag&Drop Rows on DataGrid
636)             protected int iRowIndexFromMouseDown;
637)             protected int iRowIndexOfItemUnderMouseToDrop;
638)             protected DataGridViewRow rwToBeDragged;
639) 
640)             protected void DataGridToDrag_MouseDown(object sender, MouseEventArgs e)
641)             {
642)                 DataGridView dgrSender = (DataGridView)sender;
643)                 int iX = e.Location.X;
644)                 int iY = e.Location.Y;
645)                 int iRowClicked = dgrSender.HitTest(iX, iY).RowIndex;
646)                 int iColClicked = dgrSender.HitTest(iX, iY).ColumnIndex;
647)                 if (dgrSender.SelectedRows.Count == 1)
648)                 {
649)                     if (e.Button == MouseButtons.Left && iRowClicked == dgrSender.Sele
    ctedRows[0].Index)
650)                     {
651)                         rwToBeDragged = dgrSender.SelectedRows[0];
652)                         iRowIndexFromMouseDown = dgrSender.SelectedRows[0].Index;
653)                         dgrSender.DoDragDrop(rwToBeDragged, DragDropEffects.Move);
654)                     }
655)                     else if (iColClicked == -1)
656)                     {//User clicked row header, so select it.
657)                         CUdfGrid.SelectSingleRowOnGrid(dgrSender, iRowClicked);
658)                     }
659)                 }
660)             }
661)             protected void DataGridToDrag_DragEnter(object sender, DragEventArgs e)
662)             {
663)                 DataGridView dgrSender = (DataGridView)sender;
664)                 if (dgrSender.SelectedRows.Count == 1)
665)                 {
666)                     e.Effect = DragDropEffects.Move;
667)                 }
668)             }
669)             protected void DataGridToDrag_DragDrop(object sender, DragEventArgs e)
670)             {
671)                 DataGridView dgrSender = (DataGridView)sender;
672)                 Point ptClientPoint = dgrSender.PointToClient(new Point(e.X, e.Y));
673)                 iRowIndexOfItemUnderMouseToDrop = dgrSender.HitTest(ptClientPoint.X, p
    tClientPoint.Y).RowIndex;
674) 
675)                 if (e.Effect == DragDropEffects.Move && iRowIndexFromMouseDown != iRow
    IndexOfItemUnderMouseToDrop)
676)                 {
677)                     if (iRowIndexOfItemUnderMouseToDrop < 0)
678)                         iRowIndexOfItemUnderMouseToDrop = 0;
679)                     dgrSender.Rows.RemoveAt(iRowIndexFromMouseDown);
680)                     dgrSender.Rows.Insert(iRowIndexOfItemUnderMouseToDrop, rwToBeDragg
    ed);
681)                     Task.Delay(TimeSpan.FromMilliseconds(100)).ContinueWith(task => CU
    dfGrid.SelectSingleRowOnGrid(dgrSender, iRowIndexOfItemUnderMouseToDrop));
682)                 }
683)             }
684) 
685)             protected void DataGridToDrag_QueryContinueDrag(object sender, QueryContin
    ueDragEventArgs e)
686)             {
687)                 int iX = Control.MousePosition.X;
```

```
688)                int iY = Control.MousePosition.Y;
689)
690)                DataGridView dgrSender = (DataGridView)sender;
691)                Point ptClientPoint = dgrSender.PointToClient(new Point(iX, iY));
692)                iRowIndexOfItemUnderMouseToDrop = dgrSender.HitTest(ptClientPoint.X, p
       tClientPoint.Y).RowIndex;
693)                if (Control.MouseButtons != MouseButtons.Left && iRowIndexOfItemUnderM
       ouseToDrop < 0)
694)                {
695)                    e.Action = DragAction.Cancel;
696)                }
697)            }
698)            private void DataGrid_CellEnter(object sender, DataGridViewCellEventArgs e
       )
699)            {
700)                if (!this.DesignMode)
701)                {
702)                    DataGridView dgrCur = (DataGridView)sender;
703)                    if (!dgrCur.Columns[e.ColumnIndex].Frozen) //There will be no Calc
        button on frozen columns.
704)                    {
705)                        CUdfGrid.ShowCalcOnCurrentCell(btnCalc, dgrCur);
706)                    }
707)                    else
708)                        btnCalc.Visible = false;
709)                }
710)            }
711)            private void DataGrid_CellPainting(object sender, DataGridViewCellPainting
       EventArgs e)
712)            {
713)                if (!this.DesignMode)
714)                {
715)                    System.Diagnostics.Debug.WriteLine("{0}: Cell being repainted: [R:
       {1}, C:{2}]", DateTime.Now.ToString(), e.RowIndex, e.ColumnIndex);
716)                    DataGridView dgrCur = (DataGridView)sender;
717)                    if(btnCalc.Visible) //Run only if Calc button is being shown on an
       y cell
718)                    {
719)                        if (dgrCur.ContainsFocus)
720)                        {
721)                            DataGridViewCell celCur = dgrCur.CurrentCell;
722)                            if ( ( e.RowIndex == celCur.RowIndex || e.RowIndex == -
       1 ) //Either Current Row or HeaderRow is being repainted
723)                                && e.ColumnIndex == celCur.ColumnIndex && celCur.Colum
       nIndex > 0 ) //There will be no Calc button on first column of grid.
724)                            {
725)                                CUdfGrid.ShowCalcOnCurrentCell(this.btnCalc, dgrCur);
726)                            }
727)                        }
728)                        else //If grid is being repainted because it lost focus, hide
       Calc button
729)                        {
730)                            Control ctlActive = this.FindForm().ActiveControl;
731)                            if(ctlActive.GetType() != typeof(CtlGrid) && ctlActive !=
       this.btnCalc)
732)                                btnCalc.Visible = false;
733)                        }
734)                    }
735)                }
736)            }
737)
738)            protected void btnAddRow_Click(object sender, EventArgs e)
739)            {
740)                Button btnAdd = (Button)sender;
741)                DataGridView dgrPropVal;
742)                DataGridView dgrPropSel;
743)                Button btnDel;
744)                PanelTransparent pnlTip;
745)                string szMsg = "";
746)                int iMaxRows = 0;
747)                if(tabAttributes.SelectedTab == tbpgVols)
748)                {
749)                    dgrPropVal = dgrVolPropVal;
750)                    dgrPropSel = dgrVolPropSel;
751)                    btnDel = btnDelVol;
752)                    pnlTip = pnlAddVolTip;
753)                    iMaxRows = CConstants.I_HYDRO_MAXVOL;
```

```
754)                 szMsg = CGlobal.Translation.TXT_ERR_PIPE_VOL_QT.Translate();
755)             }
756)             else
757)             {
758)                 dgrPropVal = dgrJunPropVal;
759)                 dgrPropSel = dgrJunPropSel;
760)                 btnDel = btnDelJun;
761)                 pnlTip = pnlAddJunTip;
762)                 iMaxRows = CConstants.I_HYDRO_MAXJUN;
763)                 szMsg = CGlobal.Translation.TXT_ERR_PIPE_JUN_QT.Translate();
764)             }
765)             this.m_blnVolIdBeingValidated = true;
766)             this.m_blnJuncIdBeingValidated = true;
767)             CUdfGrid.AddRow(dgrPropVal, btnAdd, pnlTip, btnDel, this.tipMsg, szMsg
    , v_iMaxRows:iMaxRows);
768)             this.m_blnVolIdBeingValidated = false;
769)             this.m_blnJuncIdBeingValidated = false;
770)         }
771)         protected void btnDelRow_Click(object sender, EventArgs e)
772)         {
773)             Button btnDel = (Button)sender;
774)             DataGridView dgrPropVal;
775)             DataGridView dgrPropSel;
776)             Button btnAdd;
777)             PanelTransparent pnlTip;
778)             int iMaxRows = 0;
779)             if(tabAttributes.SelectedTab == tbpgVols)
780)             {
781)                 dgrPropVal = dgrVolPropVal;
782)                 dgrPropSel = dgrVolPropSel;
783)                 btnAdd = btnAddVol;
784)                 pnlTip = pnlAddVolTip;
785)                 iMaxRows = CConstants.I_HYDRO_MAXVOL;
786)             }
787)             else
788)             {
789)                 dgrPropVal = dgrJunPropVal;
790)                 dgrPropSel = dgrJunPropSel;
791)                 btnAdd = btnAddJun;
792)                 pnlTip = pnlAddJunTip;
793)                 iMaxRows = CConstants.I_HYDRO_MAXJUN;
794)             }
795)             int iCurRow = dgrPropVal.CurrentCell.RowIndex;
796)             if (CUdfGrid.DelRow(dgrPropVal, btnAdd, pnlTip, btnDel, this.tipMsg, r
    ef p_szInvalidFields, iMaxRows))
797)             {
798)                 UpdateNumberOfVolumes();
799)                 //Search who is the new last volume, and validate it
800)                 int iMaxVolRow = GetMaxRowId(dgrPropVal, 0);
801)                 if(iMaxVolRow >=0)
802)                 {
803)                     if(dgrPropVal[0,iMaxVolRow].Value != null)
804)                     {
805)                         //Validate all properties for new Last Volume
806)                         for(int i= dgrPropVal.ColumnCount - 1; i>0;i--)
807)                             dgrPropVal.CurrentCell = dgrPropVal[i, iMaxVolRow];
808)                         if (iCurRow > dgrPropVal.RowCount - 1)
809)                             iCurRow = dgrPropVal.RowCount - 1;
810)                     }
811)                 }
812)             }
813)         }
814)
815)   //        protected override void btnCalc_Click(object sender, EventArgs e)
816)   //        {
817)   //            string szVal = "";
818)   //            DataGridViewCell celCur = (DataGridViewCell) btnCalc.Tag;
819)   //            string szCurTag = celCur.DataGridView.Columns[celCur.ColumnIndex].Ta
    g.ToString();
820)   //            szVal = CUdf.ShowCalcForms(szCurTag, HydroComponent, this);
821)   //            if (szVal != "")
822)   //                celCur.Value = szVal;
823)   //        }
824)
825)         protected void DataGrid_KeyDown(object sender, KeyEventArgs e)
826)         {
827)             DataGridView dgrCur = (DataGridView)sender;
```

```
828)
829)                if (e.Control && e.KeyCode == Keys.C)
830)                {
831)                    DataObject d = dgrCur.GetClipboardContent();
832)                    Clipboard.SetDataObject(d);
833)                    e.Handled = true;
834)                }
835)                else if (e.Control && e.KeyCode == Keys.V)
836)                {
837)                    string s = Clipboard.GetText();
838)                    string[] aszRows = s.Split('\n');
839)                    string[] aszCols;
840)                    int iRow = dgrCur.CurrentCell.RowIndex;
841)                    int iCol = dgrCur.CurrentCell.ColumnIndex;
842)                    for(int i=0; i < aszRows.Length; i++)
843)                    {
844)                        aszCols = aszRows[i].Split('\t');
845)                        int iCurCol = aszCols.Length;
846)                        for (int j = 0; j < iCurCol; j++)
847)                        {
848)                            dgrCur[iCol+j, iRow+i].Value = aszCols[j];
849)                        }
850)                    }
851)                }
852)            }
853)
854)            private void tabAttributes_Selected(object sender, TabControlEventArgs e)
855)            {
856)                btnCalc.Visible = false;
857)            }
858)        }
859)
860)        public class PanelTransparent : Panel
861)        {
862)            public PanelTransparent() : base()
863)            {
864)
865)            }
866)            protected override void OnPaintBackground(PaintEventArgs e)
867)            {
868)                base.OnPaintBackground(e);
869)                Graphics g = e.Graphics;
870)
871)                if (Parent != null)
872)                {
873)                    // Take each control in turn
874)                    int index = Parent.Controls.GetChildIndex(this);
875)                    for (int i = Parent.Controls.Count - 1; i > index; i--)
876)                    {
877)                        Control c = Parent.Controls[i];
878)
879)                        // Check it's visible and overlaps this control
880)                        if (c.Bounds.IntersectsWith(Bounds) && c.Visible)
881)                        {
882)                            // Load appearance of underlying control and redraw it on
    this background
883)                            Bitmap bmp = new Bitmap(c.Width, c.Height, g);
884)                            c.DrawToBitmap(bmp, c.ClientRectangle);
885)                            g.TranslateTransform(c.Left - Left, c.Top - Top);
886)                            g.DrawImageUnscaled(bmp, Point.Empty);
887)                            g.TranslateTransform(Left - c.Left, Top - c.Top);
888)                            bmp.Dispose();
889)                        }
890)                    }
891)                }
892)            }
893)        }
894)    }
```

APÊNDICE C - Programação da Estrutura de Troca de Calor

```
1)  using System;
2)  using System.Collections.Generic;
3)  using System.Linq;
4)  using System.Text;
5)  using System.Threading.Tasks;
6)  using Office = Microsoft.Office.Core;
7)  using Excel = Microsoft.Office.Interop.Excel;
8)  using Microsoft.Office.Tools.Excel;
9)
10) namespace FastLapAddin
11) {
12)     public class CHeat : CComponent
13)     {
14)         public EnumHeatGeometryType GeometryType = EnumHeatGeometryType.Geometry0N
    otInformed;
15)         //public new CHeatProperties PropertyCards;
16)
17)         public CHeat()
18)         {
19)             this.BaseShape = CBaseShape.ShapeHeat;
20)             this.PropertyCards = new CHeatProperties();
21)         }
22)         public override string Name { get { return CConstants.SZ_HEAT + this.Compo
    nentNumber; } }
23)         public override string FullDescription
24)         {
25)             get
26)             {
27)                 string szRet = "";
28)                 if (this.ComponentNumber != "")
29)                     szRet = this.ComponentNumber + CConstants.SZ_TXT_DESCR_SEP + t
    his.Description;
30)                 return szRet;
31)             }
32)         }
33)         public CHeatProperties HeatProperties
34)         {
35)             get
36)             {
37)                 CHeatProperties objRet = null;
38)                 try
39)                 {
40)                     objRet = (CHeatProperties)this.PropertyCards;
41)                 }
42)                 catch { }
43)                     return objRet;
44)             }
45)         }
46)
47)         public void Draw(string v_szCompNoAndGeomNo, string v_szDescription = "",
    EnumHeatGeometryType v_enuGeometryType = EnumHeatGeometryType.Geometry0NotInformed
    , CHydro v_szBoundaryHydroComponent = null)
48)         {
49)             this.ComponentNumber = v_szCompNoAndGeomNo;
50)             this.Description = v_szDescription;
51)             this.GeometryType = v_enuGeometryType;
52)             this.BoundaryHydroComponent = v_szBoundaryHydroComponent;
53)             this.NumberOfVolumes = 1;
54)             this.Draw();
55)         }
56)
57)
58)         public static Dictionary<string, string> ListGeometryTypes(bool v_blnInclu
    deemptyTrip = true)
59)         {
60)             Dictionary<string, string> dicRet = new Dictionary<string, string>
61)                 {   { "", null }
62)                   , { CConstants.SZ_HEAT_GEOM_RECTANGULAR , CConstants.SZ_HEAT_GEO
    M_RECTANGULAR.ToUpper() + CConstants.SZ_LBL_SUFIX + CGlobal.Translation.TXT_HEAT_G
    EOM_RECTANGULAR.Translate()  }
```

```
63)                     , { CConstants.SZ_HEAT_GEOM_CYLINDRICAL , CConstants.SZ_HEAT_GEO
      M_CYLINDRICAL.ToUpper() + CConstants.SZ_LBL_SUFIX + CGlobal.Translation.TXT_HEAT_G
      EOM_CYLINDRICAL.Translate()  }
64)                     , { CConstants.SZ_HEAT_GEOM_SPHERICAL   , CConstants.SZ_HEAT_GEO
      M_SPHERICAL.ToUpper()   + CConstants.SZ_LBL_SUFIX + CGlobal.Translation.TXT_HEAT_G
      EOM_SPHERICAL.Translate()    }
65)                 };
66)             if (!v_blnIncludeemptyTrip)
67)                 dicRet.Remove("");
68)             return dicRet;
69)         }
70)
71)         public static IList<CHeat> ListComponents(bool v_blnFirstItemAsEmpty = tru
      e)
72)         {
73)             IList<CHeat> laszRet = new List<CHeat>();
74)             CShtInp oshtInp = CModel.GetShtInp();
75)             Excel.Worksheet shtInp = oshtInp.TargetSheet;
76)             int iCurRow = CConstants.I_REF_SHT_INP_ROW_1STROW + 1;
77)             Excel.Range rngCur = shtInp.Cells[iCurRow, CConstants.I_REF_SHT_INP_CO
      L_CARD];
78)             CHeat objHeat;
79)
80)             string szVal;
81)             string szCompNo;
82)             string szCommentIdentif;
83)             string szCompDescr;
84)             string szQtVolJun;
85)             int iIxColon;
86)
87)             if (v_blnFirstItemAsEmpty)
88)             {
89)                 objHeat = new CHeat();
90)                 objHeat.ComponentNumber = "";
91)                 objHeat.Name = "";
92)                 objHeat.Description = "";
93)                 objHeat.RowNumber = -1;
94)                 laszRet.Add(objHeat);
95)             }
96)
97)             while (rngCur.Text != "")
98)             {
99)                 szCompNo = "";
100)                szCommentIdentif = "";
101)                szCompDescr = "";
102)                iIxColon = -1;
103)                szVal = rngCur.Text;
104)                if (szVal.Length == CConstants.SZ_INP_CARDLEN_HEAT
105)                    && szVal.Left(1) == CConstants.SZ_INP_PREFIX_HEAT
106)                    && szVal.Substring(5).Left(CConstants.SZ_INP_SEARCHMASK_HEATID
      .Length) == CConstants.SZ_INP_SEARCHMASK_HEATID)
107)                {//Found a HEAT Structure Data Card
108)                    szCommentIdentif = shtInp.Cells[iCurRow - 1, CConstants.I_REF_
      SHT_INP_COL_CARD].Text;
109)                    iIxColon = szCommentIdentif.IndexOf(CConstants.SZ_TXT_DESCR_SE
      P);
110)
111)                    szCompNo = szVal.Substring(1, 4);
112)                    szCompDescr = szCommentIdentif.Substring(iIxColon + CConstants
      .SZ_TXT_DESCR_SEP.Length);
113)
114)                    objHeat = new CHeat();
115)                    szQtVolJun = shtInp.Cells[iCurRow + 1, CConstants.I_REF_SHT_IN
      P_COL_WORD1].Text;
116)
117)                    objHeat.ComponentNumber = szCompNo;
118)                    objHeat.Description = szCompDescr;
119)                    objHeat.RowNumber = iCurRow;
120)                    laszRet.Add(objHeat);
121)                }
122)
123)                iCurRow++;
124)                rngCur = shtInp.Cells[iCurRow, CConstants.I_REF_SHT_INP_COL_CARD];
125)            }
126)
127)            return laszRet;
128)        }
```

```
129)        }
130)
131)        public enum EnumHeatGeometryType
132)        {
133)              Geometry0NotInformed = 0
134)            , Geometry1Rectangular = 1
135)            , Geometry2Cylindrical = 2
136)            , Geometry3Spherical   = 3
137)        }
138)
139)        public class CHeatProperties : CCompProperties
140)        {
141)            public override string CardPrefix {  get { return "1"; } }
142)
143)            public CCompProperty heatpHeatNumberAndGeom                    { get { retu
        rn this.CreateCompProperty(" "       ,       true , null , CGlobal.Translation.TXT_
        HEAT_PROP_COMPNO.Translate()        , EnumHydroPropertyType.WordCompGeomNo); } }
144)            public CCompProperty heatpHeatNumber                           { get { retu
        rn this.CreateCompProperty(" "       ,       true , null , CGlobal.Translation.TXT_
        HEAT_PROP_COMPNO.Translate()        , EnumHydroPropertyType.WordCompNumber); } }
145)            public CCompProperty heatpHeatGeometryNumber                   { get { retu
        rn this.CreateCompProperty(" "       ,       true , null , CGlobal.Translation.TXT_
        HEAT_PROP_GEOMNO.Translate()        , EnumHydroPropertyType.WordInteger1to9); } }
146)            public CCompProperty heatpHeatStruct_000W1_Nh                  { get { retu
        rn this.CreateCompProperty("000 W1" ,       true , null , CGlobal.Translation.TXT_
        HEAT_PROP_CCCG000_W1.Translate()    , EnumHydroPropertyType.WordInteger1to99); } }
147)            public CCompProperty heatpHeatStruct_000W2_Np                  { get { retu
        rn this.CreateCompProperty("000 W2" ,       true , null , CGlobal.Translation.TXT_
        HEAT_PROP_CCCG000_W2.Translate()    , EnumHydroPropertyType.WordInteger1to99); } }
148)            public CCompProperty heatpHeatStruct_000W3_GeomType            { get { retu
        rn this.CreateCompProperty("000 W3" ,       true , null , CGlobal.Translation.TXT_
        HEAT_PROP_CCCG000_W3.Translate()    , EnumHydroPropertyType.WordInteger123); } }
149)            public CCompProperty heatpHeatStruct_000W4_SteadSttFlg         { get { retu
        rn this.CreateCompProperty("000 W4" ,       true , null , CGlobal.Translation.TXT_
        HEAT_PROP_CCCG000_W4.Translate()    , EnumHydroPropertyType.WordInteger0or1); } }
150)            public CCompProperty heatpHeatStruct_000W5_LeftBoundCoord      { get { retu
        rn this.CreateCompProperty("000 W5" ,       true , null , CGlobal.Translation.TXT_
        HEAT_PROP_CCCG000_W5.Translate()    , EnumHydroPropertyType.WordReal); } }
151)            public CCompProperty heatpHeatStruct_000W6_RefloodFlg          { get { retu
        rn this.CreateCompProperty("000 W6" ,       true , null , CGlobal.Translation.TXT_
        HEAT_PROP_CCCG000_W6.Translate()    , EnumHydroPropertyType.WordInteger); } }
152)            public CCompProperty heatpHeatStruct_000W7_BoundVolInd         { get { retu
        rn this.CreateCompProperty("000 W7" ,       true , null , CGlobal.Translation.TXT_
        HEAT_PROP_CCCG000_W7.Translate()    , EnumHydroPropertyType.WordInteger0or1); } }
153)            public CCompProperty heatpHeatStruct_000W8_AxialInterv         { get { retu
        rn this.CreateCompProperty("000 W8" ,       true , null , CGlobal.Translation.TXT_
        HEAT_PROP_CCCG000_W8.Translate()    , EnumHydroPropertyType.WordInteger); } }
154)            public CCompProperty heatpHeatStruct_000                       { get { retu
        rn this.CreateCompProperty("000 "    ,       true , null , CGlobal.Translation.TXT_
        HEAT_PROP_CCCG000.Translate()       , new CCompProperty[] { this.heatpHeatStruct_0
        00W1_Nh, this.heatpHeatStruct_000W2_Np, this.heatpHeatStruct_000W3_GeomType, this.
        heatpHeatStruct_000W4_SteadSttFlg, this.heatpHeatStruct_000W5_LeftBoundCoord, this
        .heatpHeatStruct_000W6_RefloodFlg, this.heatpHeatStruct_000W7_BoundVolInd, this.he
        atpHeatStruct_000W8_AxialInterv }); } }
155)            public CCompProperty heatpGapDt_001W1_IniGapPressure           { get { retu
        rn this.CreateCompProperty("001 W1" ,       true , null , CGlobal.Translation.TXT_
        HEAT_PROP_CCCG001_W1.Translate()    , EnumHydroPropertyType.WordReal); } }
156)            public CCompProperty heatpGapDt_001W2_GapConductVol            { get { retu
        rn this.CreateCompProperty("001 W2" ,       true , null , CGlobal.Translation.TXT_
        HEAT_PROP_CCCG001_W2.Translate()    , EnumHydroPropertyType.WordIntComponent); } }
157)            public CCompProperty heatpGapDt_001                            { get { retu
        rn this.CreateCompProperty("001 "    ,       false, null , CGlobal.Translation.TXT_
        HEAT_PROP_CCCG001.Translate()       , new CCompProperty[] { this.heatpGapDt_001W1_
        IniGapPressure, heatpGapDt_001W2_GapConductVol }); } }
158)            public CCompProperty heatpMetalWaterReac_003_IniOxThick        { get { retu
        rn this.CreateCompProperty("003 "    ,       false, null , CGlobal.Translation.TXT_
        HEAT_PROP_CCCG003.Translate()       , EnumHydroPropertyType.WordReal); } }
159)            public CCompProperty heatpFuelDeform_004_FormLossF             { get { retu
        rn this.CreateCompProperty("004 "    ,       false, null , CGlobal.Translation.TXT_
        HEAT_PROP_CCCG004.Translate()       , EnumHydroPropertyType.WordInteger0or1); } }
160)            public CCompProperty heatpGapDefDt_099W1_FuelRough             { get { retu
        rn this.CreateCompProperty("099 W1" ,       true , null , CGlobal.Translation.TXT_
        HEAT_PROP_CCCG099_W1.Translate()    , EnumHydroPropertyType.WordReal); } }
161)            public CCompProperty heatpGapDefDt_099W2_CladRough             { get { retu
        rn this.CreateCompProperty("099 W2" ,       true , null , CGlobal.Translation.TXT_
        HEAT_PROP_CCCG099_W2.Translate()    , EnumHydroPropertyType.WordReal); } }
```

```
162)          public CCompProperty heatpGapDefDt_099W3_DisplFission         { get { retu
      rn this.CreateCompProperty("099 W3" ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG099_W3.Translate()    , EnumHydroPropertyType.WordReal); } }
163)          public CCompProperty heatpGapDefDt_099W4_DisplCreep           { get { retu
      rn this.CreateCompProperty("099 W4" ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG099_W4.Translate()    , EnumHydroPropertyType.WordReal); } }
164)          public CCompProperty heatpGapDefDt_099                        { get { retu
      rn this.CreateCompProperty("099 "   ,11,99, false, null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG099.Translate()       , new CCompProperty[] { this.heatpGapDefDt_099
      W1_FuelRough, this.heatpGapDefDt_099W2_CladRough, this.heatpGapDefDt_099W3_DisplFi
      ssion, this.heatpGapDefDt_099W4_DisplCreep }); } }
165)          public CCompProperty heatpMeshFlg_100W1_MeshLocation          { get { retu
      rn this.CreateCompProperty("100 W1" ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG100_W1.Translate()    , EnumHydroPropertyType.WordIntHeat); } }
166)          public CCompProperty heatpMeshFlg_100W2_MeshFormat            { get { retu
      rn this.CreateCompProperty("100 W2" ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG100_W2.Translate()    , EnumHydroPropertyType.WordInteger012); } }
167)          public CCompProperty heatpMeshFlg_100                         { get { retu
      rn this.CreateCompProperty("100 "   ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG100.Translate()       , new CCompProperty[] { this.heatpMeshFlg_100W
      1_MeshLocation, this.heatpMeshFlg_100W2_MeshFormat}); } }
168)          public CCompProperty heatpMeshInt_199_MeshInterval            { get { retu
      rn this.CreateCompProperty("199 "   ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG199.Translate()       , EnumHydroPropertyType.WordReal); } }
169)          public CCompProperty heatpComposition_299_CompositionNo       { get { retu
      rn this.CreateCompProperty("299 "   ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG299.Translate()       , EnumHydroPropertyType.WordIntComposition); }
       }
170)          public CCompProperty heatpDecay_300_DkHeat                    { get { retu
      rn this.CreateCompProperty("300 "   ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG300.Translate()       , EnumHydroPropertyType.WordAlphaDkheat); } }
171)          public CCompProperty heatpHeatDistr_399_SourceValOrGamma      { get { retu
      rn this.CreateCompProperty("399 "   ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG399.Translate()       , EnumHydroPropertyType.WordReal); } }
172)          public CCompProperty heatpIniTempFlg_400_Flg                  { get { retu
      rn this.CreateCompProperty("400 "   ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG400.Translate()       , EnumHydroPropertyType.WordIntHeat); } }
173)          public CCompProperty heatpIniTempDt_499                       { get { retu
      rn this.CreateCompProperty("499 "   ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG499.Translate()       , EnumHydroPropertyType.WordReal); } }
174)          public CCompProperty heatpLeftBound_599W1_BoundVol            { get { retu
      rn this.CreateCompProperty("599 W1" ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG599_W1.Translate()    , EnumHydroPropertyType.WordIntVolOrTbl); } }
175)          public CCompProperty heatpLeftBound_599W2_Increment           { get { retu
      rn this.CreateCompProperty("599 W2" ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG599_W2.Translate()    , EnumHydroPropertyType.WordInteger); } }
176)          public CCompProperty heatpLeftBound_599W3_CondType            { get { retu
      rn this.CreateCompProperty("599 W3" ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG599_W3.Translate()    , EnumHydroPropertyType.WordIntBoundCondition)
      ; } }
177)          public CCompProperty heatpLeftBound_599W4_SurfAreaCode        { get { retu
      rn this.CreateCompProperty("599 W4" ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG599_W4.Translate()    , EnumHydroPropertyType.WordInteger0or1); } }
178)          public CCompProperty heatpLeftBound_599W5_SorfAreaF           { get { retu
      rn this.CreateCompProperty("599 W5" ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG599_W5.Translate()    , EnumHydroPropertyType.WordReal); } }
179)          public CCompProperty heatpLeftBound_599                       { get { retu
      rn this.CreateCompProperty("599 "   ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG599.Translate()       , new CCompProperty[] { this.heatpLeftBound_59
      9W1_BoundVol, this.heatpLeftBound_599W2_Increment, this.heatpLeftBound_599W3_CondT
      ype, this.heatpLeftBound_599W4_SurfAreaCode, this.heatpLeftBound_599W5_SorfAreaF }
      ); } }
180)          public CCompProperty heatpRightBound_699W1_BoundVol           { get { retu
      rn this.CreateCompProperty("699 W1" ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG599_W1.Translate()    , EnumHydroPropertyType.WordIntVolOrTbl); } }
181)          public CCompProperty heatpRightBound_699W2_Increment          { get { retu
      rn this.CreateCompProperty("699 W2" ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG599_W2.Translate()    , EnumHydroPropertyType.WordInteger); } }
182)          public CCompProperty heatpRightBound_699W3_CondType           { get { retu
      rn this.CreateCompProperty("699 W3" ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG599_W3.Translate()    , EnumHydroPropertyType.WordIntBoundCondition)
      ; } }
183)          public CCompProperty heatpRightBound_699W4_SurfAreaCode       { get { retu
      rn this.CreateCompProperty("699 W4" ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG599_W4.Translate()    , EnumHydroPropertyType.WordInteger0or1); } }
```

```
184)          public CCompProperty heatpRightBound_699W5_SorfAreaF          { get { retu
      rn this.CreateCompProperty("699 W5" ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG599_W5.Translate()    , EnumHydroPropertyType.WordReal); } }
185)          public CCompProperty heatpRightBound_699                      { get { retu
      rn this.CreateCompProperty("699 "   ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG699.Translate()       , new CCompProperty[] { this.heatpRightBound_6
      99W1_BoundVol, this.heatpRightBound_699W2_Increment, this.heatpRightBound_699W3_Co
      ndType, this.heatpRightBound_699W4_SurfAreaCode, this.heatpRightBound_699W5_SorfAr
      eaF }); } }
186)          public CCompProperty heatpHeatSrc_799W1_SrcType               { get { retu
      rn this.CreateCompProperty("799 W1" ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG799_W1.Translate()    , EnumHydroPropertyType.WordInteger); } }
187)          public CCompProperty heatpHeatSrc_799W2_IntlSrcMultipl        { get { retu
      rn this.CreateCompProperty("799 W2" ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG799_W2.Translate()    , EnumHydroPropertyType.WordReal); } }
188)          public CCompProperty heatpHeatSrc_799W3_LeftMultipl           { get { retu
      rn this.CreateCompProperty("799 W3" ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG799_W3.Translate()    , EnumHydroPropertyType.WordReal); } }
189)          public CCompProperty heatpHeatSrc_799W4_RightMultipl          { get { retu
      rn this.CreateCompProperty("799 W4" ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG799_W4.Translate()    , EnumHydroPropertyType.WordReal); } }
190)          public CCompProperty heatpHeatSrc_799                         { get { retu
      rn this.CreateCompProperty("799 "   ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG799.Translate()       , new CCompProperty[] { this.heatpHeatSrc_799W
      1_SrcType, this.heatpHeatSrc_799W2_IntlSrcMultipl, this.heatpHeatSrc_799W3_LeftMul
      tipl, this.heatpHeatSrc_799W4_RightMultipl }); } }
191)          public CCompProperty heatpLeftBoundAdd_800                    { get { retu
      rn this.CreateCompProperty("800 "   ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG800.Translate()       , EnumHydroPropertyType.WordInteger012); } }
192)          public CCompProperty heatpLeftBoundAdd9_899W1_HydrDiam        { get { retu
      rn this.CreateCompProperty("899 W01",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_W1.Translate()    , EnumHydroPropertyType.WordReal); } }
193)          public CCompProperty heatpLeftBoundAdd9_899W2_HeatLenFwd      { get { retu
      rn this.CreateCompProperty("899 W02",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_W2.Translate()    , EnumHydroPropertyType.WordRealHigherThanZero
      ); } }
194)          public CCompProperty heatpLeftBoundAdd9_899W3_HeatLenRev      { get { retu
      rn this.CreateCompProperty("899 W03",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_W3.Translate()    , EnumHydroPropertyType.WordRealHigherThanZero
      ); } }
195)          public CCompProperty heatpLeftBoundAdd9_899W4_GrdSpcLenFwd    { get { retu
      rn this.CreateCompProperty("899 W04",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_W4.Translate()    , EnumHydroPropertyType.WordReal); } }
196)          public CCompProperty heatpLeftBoundAdd9_899W5_GrdSpcLenRev    { get { retu
      rn this.CreateCompProperty("899 W05",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_W5.Translate()    , EnumHydroPropertyType.WordReal); } }
197)          public CCompProperty heatpLeftBoundAdd9_899W6_GrdLossCoefFwd  { get { retu
      rn this.CreateCompProperty("899 W06",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_W6.Translate()    , EnumHydroPropertyType.WordReal); } }
198)          public CCompProperty heatpLeftBoundAdd9_899W7_GrdLossCoefRef  { get { retu
      rn this.CreateCompProperty("899 W07",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_W7.Translate()    , EnumHydroPropertyType.WordReal); } }
199)          public CCompProperty heatpLeftBoundAdd9_899W8_BoilingF        { get { retu
      rn this.CreateCompProperty("899 W08",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_W8.Translate()    , EnumHydroPropertyType.WordRealHigherThanZero
      ); } }
200)          public CCompProperty heatpLeftBoundAdd12_899W9_NatCircLen     { get { retu
      rn this.CreateCompProperty("899 W09",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_W9.Translate()    , EnumHydroPropertyType.WordReal); } }
201)          public CCompProperty heatpLeftBoundAdd12_899W10_RodPitRatio   { get { retu
      rn this.CreateCompProperty("899 W10",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_WA.Translate()    , EnumHydroPropertyType.WordReal); } }
202)          public CCompProperty heatpLeftBoundAdd12_899W11_FoulingF      { get { retu
      rn this.CreateCompProperty("899 W11",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_WB.Translate()    , EnumHydroPropertyType.WordRealHigherThanZero)
      ; } }
203)          public CCompProperty heatpLeftBoundAdd13_899W1_NotUsed        { get { retu
      rn this.CreateCompProperty("899 W01",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_13W1.Translate()   , EnumHydroPropertyType.WordRealZero); } }
204)          public CCompProperty heatpLeftBoundAdd13_899W2_RdcHeatLenFwd  { get { retu
      rn this.CreateCompProperty("899 W02",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_13W2.Translate()   , EnumHydroPropertyType.WordReal); } }
205)          public CCompProperty heatpLeftBoundAdd13_899W3_RdcHeatLenRev  { get { retu
      rn this.CreateCompProperty("899 W03",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_13W3.Translate()   , EnumHydroPropertyType.WordReal); } }
```

```
206)          public CCompProperty heatpLeftBoundAdd13_899W4_GrdSpcFFwd     { get { retu
      rn this.CreateCompProperty("899 W04",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_13W4.Translate()  , EnumHydroPropertyType.WordReal); } }
207)          public CCompProperty heatpLeftBoundAdd13_899W5_GrdSpcFRev     { get { retu
      rn this.CreateCompProperty("899 W05",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_13W5.Translate()  , EnumHydroPropertyType.WordReal); } }
208)          public CCompProperty heatpLeftBoundAdd13_899W6_RHeatFluxF     { get { retu
      rn this.CreateCompProperty("899 W06",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_13W6.Translate()  , EnumHydroPropertyType.WordReal); } }
209)          public CCompProperty heatpLeftBoundAdd13_899W7_ChUpVolNo       { get { retu
      rn this.CreateCompProperty("899 W07",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_13W7.Translate()  , EnumHydroPropertyType.WordIntComponent); } }
210)          public CCompProperty heatpLeftBoundAdd13_899W8_ChDwVolNo       { get { retu
      rn this.CreateCompProperty("899 W08",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_13W8.Translate()  , EnumHydroPropertyType.WordIntComponent); } }
211)          public CCompProperty heatpLeftBoundAdd13_899W12_ChfrOpt        { get { retu
      rn this.CreateCompProperty("899 W12",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_13WC.Translate() , EnumHydroPropertyType.WordIntMN); } }
212)          public CCompProperty heatpLeftBoundAdd9Word_899                { get { retu
      rn this.CreateCompProperty("899 "   ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899.Translate()       , new CCompProperty[] { this.heatpLeftBoundAdd
      9_899W1_HydrDiam, this.heatpLeftBoundAdd9_899W2_HeatLenFwd    , this.heatpLeftBoun
      dAdd9_899W3_HeatLenRev    , this.heatpLeftBoundAdd9_899W4_GrdSpcLenFwd, this.heatp
      LeftBoundAdd9_899W5_GrdSpcLenRev, this.heatpLeftBoundAdd9_899W6_GrdLossCoefFwd, th
      is.heatpLeftBoundAdd9_899W7_GrdLossCoefRef, this.heatpLeftBoundAdd9_899W8_BoilingF
        } ); } }
213)          public CCompProperty heatpLeftBoundAdd12Word_899               { get { retu
      rn this.CreateCompProperty("899 "   ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899.Translate()       , new CCompProperty[] { this.heatpLeftBoundAdd
      9_899W1_HydrDiam, this.heatpLeftBoundAdd9_899W2_HeatLenFwd    , this.heatpLeftBoun
      dAdd9_899W3_HeatLenRev    , this.heatpLeftBoundAdd9_899W4_GrdSpcLenFwd, this.heatp
      LeftBoundAdd9_899W5_GrdSpcLenRev, this.heatpLeftBoundAdd9_899W6_GrdLossCoefFwd, th
      is.heatpLeftBoundAdd9_899W7_GrdLossCoefRef, this.heatpLeftBoundAdd9_899W8_BoilingF
        , this.heatpLeftBoundAdd12_899W9_NatCircLen, this.heatpLeftBoundAdd12_899W10_Rod
      PitRatio, this.heatpLeftBoundAdd12_899W11_FoulingF } ); } }
214)          public CCompProperty heatpLeftBoundAdd13Word_899               { get { retu
      rn this.CreateCompProperty("899 "   ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899.Translate()       , new CCompProperty[] { this.heatpLeftBoundAdd
      13_899W1_NotUsed, this.heatpLeftBoundAdd13_899W2_RdcHeatLenFwd, this.heatpLeftBoun
      dAdd13_899W3_RdcHeatLenRev, this.heatpLeftBoundAdd13_899W4_GrdSpcFFwd , this.heatp
      LeftBoundAdd13_899W5_GrdSpcFRev , this.heatpLeftBoundAdd13_899W6_RHeatFluxF   , th
      is.heatpLeftBoundAdd13_899W7_ChUpVolNo    , this.heatpLeftBoundAdd13_899W8_ChDwVol
      No, this.heatpLeftBoundAdd12_899W9_NatCircLen, this.heatpLeftBoundAdd12_899W10_Rod
      PitRatio, this.heatpLeftBoundAdd12_899W11_FoulingF, this.heatpLeftBoundAdd13_899W1
      2_ChfrOpt } ); } }
215)          public CCompProperty heatpRightBoundAdd_900                    { get { retu
      rn this.CreateCompProperty("900 "   ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG900.Translate()       , EnumHydroPropertyType.WordInteger012); } }
216)          public CCompProperty heatpRightBoundAdd9_999W1_HydrDiam        { get { retu
      rn this.CreateCompProperty("999 W01",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_W1.Translate()    , EnumHydroPropertyType.WordReal); } }
217)          public CCompProperty heatpRightBoundAdd9_999W2_HeatLenFwd      { get { retu
      rn this.CreateCompProperty("999 W02",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_W2.Translate()    , EnumHydroPropertyType.WordRealHigherThanZero
      ); } }
218)          public CCompProperty heatpRightBoundAdd9_999W3_HeatLenRev      { get { retu
      rn this.CreateCompProperty("999 W03",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_W3.Translate()    , EnumHydroPropertyType.WordRealHigherThanZero
      ); } }
219)          public CCompProperty heatpRightBoundAdd9_999W4_GrdSpcLenFwd    { get { retu
      rn this.CreateCompProperty("999 W04",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_W4.Translate()    , EnumHydroPropertyType.WordReal); } }
220)          public CCompProperty heatpRightBoundAdd9_999W5_GrdSpcLenRev    { get { retu
      rn this.CreateCompProperty("999 W05",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_W5.Translate()    , EnumHydroPropertyType.WordReal); } }
221)          public CCompProperty heatpRightBoundAdd9_999W6_GrdLossCoefFwd { get { retu
      rn this.CreateCompProperty("999 W06",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_W6.Translate()    , EnumHydroPropertyType.WordReal); } }
222)          public CCompProperty heatpRightBoundAdd9_999W7_GrdLossCoefRef { get { retu
      rn this.CreateCompProperty("999 W07",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_W7.Translate()    , EnumHydroPropertyType.WordReal); } }
223)          public CCompProperty heatpRightBoundAdd9_999W8_BoilingF        { get { retu
      rn this.CreateCompProperty("999 W08",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_W8.Translate()    , EnumHydroPropertyType.WordRealHigherThanZero
      ); } }
```

```
224)          public CCompProperty heatpRightBoundAdd12_999W9_NatCircLen    { get { retu
      rn this.CreateCompProperty("999 W09",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_W9.Translate()    , EnumHydroPropertyType.WordReal); } }
225)          public CCompProperty heatpRightBoundAdd12_999W10_RodPitRatio  { get { retu
      rn this.CreateCompProperty("999 W10",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_WA.Translate()   , EnumHydroPropertyType.WordReal); } }
226)          public CCompProperty heatpRightBoundAdd12_999W11_FoulingF     { get { retu
      rn this.CreateCompProperty("999 W11",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_WB.Translate()   , EnumHydroPropertyType.WordRealHigherThanZero)
      ; } }
227)          public CCompProperty heatpRightBoundAdd13_999W1_NotUsed       { get { retu
      rn this.CreateCompProperty("999 W01",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_13W1.Translate()  , EnumHydroPropertyType.WordRealZero); } }
228)          public CCompProperty heatpRightBoundAdd13_999W2_RdcHeatLenFwd { get { retu
      rn this.CreateCompProperty("999 W02",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_13W2.Translate()  , EnumHydroPropertyType.WordReal); } }
229)          public CCompProperty heatpRightBoundAdd13_999W3_RdcHeatLenRev { get { retu
      rn this.CreateCompProperty("999 W03",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_13W3.Translate()  , EnumHydroPropertyType.WordReal); } }
230)          public CCompProperty heatpRightBoundAdd13_999W4_GrdSpcFFwd     { get { retu
      rn this.CreateCompProperty("999 W04",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_13W4.Translate()  , EnumHydroPropertyType.WordReal); } }
231)          public CCompProperty heatpRightBoundAdd13_999W5_GrdSpcFRev     { get { retu
      rn this.CreateCompProperty("999 W05",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_13W5.Translate()  , EnumHydroPropertyType.WordReal); } }
232)          public CCompProperty heatpRightBoundAdd13_999W6_RHeatFluxF     { get { retu
      rn this.CreateCompProperty("999 W06",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_13W6.Translate()  , EnumHydroPropertyType.WordReal); } }
233)          public CCompProperty heatpRightBoundAdd13_999W7_ChUpVolNo      { get { retu
      rn this.CreateCompProperty("999 W07",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_13W7.Translate()  , EnumHydroPropertyType.WordIntComponent); } }
234)          public CCompProperty heatpRightBoundAdd13_999W8_ChDwVolNo      { get { retu
      rn this.CreateCompProperty("999 W08",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_13W8.Translate()  , EnumHydroPropertyType.WordIntComponent); } }
235)          public CCompProperty heatpRightBoundAdd13_999W12_ChfrOpt       { get { retu
      rn this.CreateCompProperty("999 W12",       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG899_13WC.Translate() , EnumHydroPropertyType.WordIntMN); } }
236)          public CCompProperty heatpRightBoundAdd9Word_999               { get { retu
      rn this.CreateCompProperty("999 "   ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG999.Translate()       , new CCompProperty[] { this.heatpRightBoundAd
      d9_999W1_HydrDiam, this.heatpRightBoundAdd9_999W2_HeatLenFwd    , this.heatpRightB
      oundAdd9_999W3_HeatLenRev    , this.heatpRightBoundAdd9_999W4_GrdSpcLenFwd, this.h
      eatpRightBoundAdd9_999W5_GrdSpcLenRev, this.heatpRightBoundAdd9_999W6_GrdLossCoefF
      wd, this.heatpRightBoundAdd9_999W7_GrdLossCoefRef, this.heatpRightBoundAdd9_999W8_
      BoilingF } ); } }
237)          public CCompProperty heatpRightBoundAdd12Word_999              { get { retu
      rn this.CreateCompProperty("999 "   ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG999.Translate()       , new CCompProperty[] { this.heatpRightBoundAd
      d9_999W1_HydrDiam, this.heatpRightBoundAdd9_999W2_HeatLenFwd    , this.heatpRightB
      oundAdd9_999W3_HeatLenRev    , this.heatpRightBoundAdd9_999W4_GrdSpcLenFwd, this.h
      eatpRightBoundAdd9_999W5_GrdSpcLenRev, this.heatpRightBoundAdd9_999W6_GrdLossCoefF
      wd, this.heatpRightBoundAdd9_999W7_GrdLossCoefRef, this.heatpRightBoundAdd9_999W8_
      BoilingF  , this.heatpRightBoundAdd12_999W9_NatCircLen, this.heatpRightBoundAdd12_
      999W10_RodPitRatio, this.heatpRightBoundAdd12_999W11_FoulingF } ); } }
238)          public CCompProperty heatpRightBoundAdd13Word_999              { get { retu
      rn this.CreateCompProperty("999 "   ,       true , null , CGlobal.Translation.TXT_
      HEAT_PROP_CCCG999.Translate()       , new CCompProperty[] { this.heatpRightBoundAd
      d13_999W1_NotUsed, this.heatpRightBoundAdd13_999W2_RdcHeatLenFwd, this.heatpRightB
      oundAdd13_999W3_RdcHeatLenRev, this.heatpRightBoundAdd13_999W4_GrdSpcFFwd , this.h
      eatpRightBoundAdd13_999W5_GrdSpcFRev , this.heatpRightBoundAdd13_999W6_RHeatFluxF
        , this.heatpRightBoundAdd13_999W7_ChUpVolNo    , this.heatpRightBoundAdd13_999W8
      _ChDwVolNo, this.heatpRightBoundAdd12_999W9_NatCircLen, this.heatpRightBoundAdd12_
      999W10_RodPitRatio, this.heatpRightBoundAdd12_999W11_FoulingF, this.heatpRightBoun
      dAdd13_999W12_ChfrOpt } ); } }
239)
240)          public override CCompProperty[] RequiredProperties
241)          {
242)              get
243)              {
244)                  CCompProperty[] aobjRet = {
245)                      heatpHeatStruct_000
246)                    , heatpGapDefDt_099
247)                    , heatpMeshFlg_100
248)                    , heatpMeshInt_199_MeshInterval
249)                    , heatpLeftBound_599
250)                    , heatpRightBound_699
251)                    , heatpHeatSrc_799
```

```
252)                 };
253)                 return aobjRet;
254)             }
255)         }
256)         public override CCompProperty[] OptionalProperties
257)         {
258)             get
259)             {
260)                 CCompProperty[] aobjRet = {
261)                     heatpGapDt_001
262)                   , heatpMetalWaterReac_003_IniOxThick
263)                   , heatpFuelDeform_004_FormLossF
264)                   , heatpComposition_299_CompositionNo
265)                   , heatpDecay_300_DkHeat
266)                   , heatpHeatDistr_399_SourceValOrGamma
267)                   , heatpIniTempFlg_400_Flg
268)                   , heatpIniTempDt_499
269)                   , heatpLeftBoundAdd_800
270)                   , heatpLeftBoundAdd9Word_899
271)                   , heatpLeftBoundAdd12Word_899
272)                   , heatpLeftBoundAdd13Word_899
273)                   , heatpRightBoundAdd_900
274)                   , heatpRightBoundAdd9Word_999
275)                   , heatpRightBoundAdd12Word_999
276)                   , heatpRightBoundAdd13Word_999
277)                 };
278)                 return aobjRet;
279)             }
280)         }
281)     }
282)
283)     public class CHeatThermal : CHeat
284)     {
285)         public CHeatThermal()
286)         {
287)             this.BaseShape = null;
288)             this.PropertyCards = new CHeatThermalProperties();
289)         }
290)         public override string Name { get { return CConstants.SZ_HEATTHERMAL + thi
    s.ComponentNumber; } }
291)         public CHeatThermalProperties HeatThermalProperties
292)         {
293)             get
294)             {
295)                 CHeatThermalProperties objRet = null;
296)                 try
297)                 {
298)                     objRet = (CHeatThermalProperties)this.PropertyCards;
299)                 }
300)                 catch { }
301)                     return objRet;
302)             }
303)         }
304)
305)         public static Dictionary<string, string> ListMaterialTypes(bool v_blnInclu
    deemptyTrip = true)
306)         {
307)             Dictionary<string, string> dicRet = new Dictionary<string, string>
308)                 {   { "", null }
309)                   , { CConstants.SZ_MAT_CSTEEL , CConstants.SZ_MAT_CSTEEL.ToUpper(
    )  + CConstants.SZ_LBL_SUFIX + CGlobal.Translation.TXT_MAT_CSTEEL.Translate()  }
310)                   , { CConstants.SZ_MAT_SSTEEL , CConstants.SZ_MAT_SSTEEL.ToUpper(
    )  + CConstants.SZ_LBL_SUFIX + CGlobal.Translation.TXT_MAT_SSTEEL.Translate()  }
311)                   , { CConstants.SZ_MAT_UO2    , CConstants.SZ_MAT_UO2.ToUpper()
       + CConstants.SZ_LBL_SUFIX + CGlobal.Translation.TXT_MAT_UO2.Translate()     }
312)                   , { CConstants.SZ_MAT_ZR     , CConstants.SZ_MAT_ZR.ToUpper()
       + CConstants.SZ_LBL_SUFIX + CGlobal.Translation.TXT_MAT_ZR.Translate()      }
313)                   , { CConstants.SZ_MAT_TBLFCTN, CConstants.SZ_MAT_TBLFCTN.ToUpper
    () + CConstants.SZ_LBL_SUFIX + CGlobal.Translation.TXT_MAT_TBLFCTN.Translate() }
314)                 };
315)             if (!v_blnIncludeemptyTrip)
316)                 dicRet.Remove("");
317)             return dicRet;
318)         }
319)         public static Dictionary<string, string> ListGases(bool v_blnIncludeemptyT
    rip = true)
320)         {
```

```
321)                Dictionary<string, string> dicRet = new Dictionary<string, string>
322)                    {   { "", null }
323)                      , { CConstants.SZ_MATGAS_HELIUM  , CConstants.SZ_MATGAS_HELIUM.T
    oUpper()   + CConstants.SZ_LBL_SUFIX + CGlobal.Translation.TXT_MATGAS_HELIUM.Trans
    late()   }
324)                      , { CConstants.SZ_MATGAS_ARGON   , CConstants.SZ_MATGAS_ARGON.To
    Upper()    + CConstants.SZ_LBL_SUFIX + CGlobal.Translation.TXT_MATGAS_ARGON.Transl
    ate()    }
325)                      , { CConstants.SZ_MATGAS_KRYPTON , CConstants.SZ_MATGAS_KRYPTON.
    ToUpper()  + CConstants.SZ_LBL_SUFIX + CGlobal.Translation.TXT_MATGAS_KRYPTON.Tran
    slate()  }
326)                      , { CConstants.SZ_MATGAS_XENON   , CConstants.SZ_MATGAS_XENON.To
    Upper()    + CConstants.SZ_LBL_SUFIX + CGlobal.Translation.TXT_MATGAS_XENON.Transl
    ate()    }
327)                      , { CConstants.SZ_MATGAS_NITROGEN, CConstants.SZ_MATGAS_NITROGEN
    .ToUpper() + CConstants.SZ_LBL_SUFIX + CGlobal.Translation.TXT_MATGAS_NITROGEN.Tra
    nslate() }
328)                      , { CConstants.SZ_MATGAS_HYDROGEN, CConstants.SZ_MATGAS_HYDROGEN
    .ToUpper() + CConstants.SZ_LBL_SUFIX + CGlobal.Translation.TXT_MATGAS_HYDROGEN.Tra
    nslate() }
329)                      , { CConstants.SZ_MATGAS_OXYGEN  , CConstants.SZ_MATGAS_OXYGEN.T
    oUpper()   + CConstants.SZ_LBL_SUFIX + CGlobal.Translation.TXT_MATGAS_OXYGEN.Trans
    late()   }
330)                    };
331)                if (!v_blnIncludeemptyTrip)
332)                    dicRet.Remove("");
333)                return dicRet;
334)            }
335)            public static IList<CHeatThermal> ListThermal(bool v_blnFirstItemAsEmpty =
     true)
336)            {
337)                IList<CHeatThermal> laszRet = new List<CHeatThermal>();
338)                CShtInp oshtInp = CModel.GetShtInp();
339)                Excel.Worksheet shtInp = oshtInp.TargetSheet;
340)                int iCurRow = CConstants.I_REF_SHT_INP_ROW_1STROW + 1;
341)                Excel.Range rngCur = shtInp.Cells[iCurRow, CConstants.I_REF_SHT_INP_CO
    L_CARD];
342)                CHeatThermal objThermal;
343)
344)                string szVal;
345)                string szCompNo;
346)                string szCommentIdentif;
347)                string szCompDescr;
348)                string szQtVolJun;
349)                int iIxColon;
350)
351)                if (v_blnFirstItemAsEmpty)
352)                {
353)                    objThermal = new CHeatThermal();
354)                    objThermal.ComponentNumber = "";
355)                    objThermal.Name = "";
356)                    objThermal.Description = "";
357)                    objThermal.RowNumber = -1;
358)                    laszRet.Add(objThermal);
359)                }
360)
361)                while (rngCur.Text != "")
362)                {
363)                    szCompNo = "";
364)                    szCommentIdentif = "";
365)                    szCompDescr = "";
366)                    iIxColon = -1;
367)                    szVal = rngCur.Text;
368)                    if (szVal.Length == CConstants.SZ_INP_CARDLEN_HEAT
369)                        && szVal.Left(3) == CConstants.SZ_INP_PREFIX_HEATTHERMAL
370)                        && szVal.Substring(6).Left(CConstants.SZ_INP_SEARCHMASK_MISCID
    .Length) == CConstants.SZ_INP_SEARCHMASK_MISCID)
371)                    {//Found a HEAT Structure Data Card
372)                        szCommentIdentif = shtInp.Cells[iCurRow - 1, CConstants.I_REF_
    SHT_INP_COL_CARD].Text;
373)                        iIxColon = szCommentIdentif.IndexOf(CConstants.SZ_TXT_DESCR_SE
    P);
374)
375)                        szCompNo = szVal.Substring(3, 3);
376)                        szCompDescr = szCommentIdentif.Substring(iIxColon + CConstants
    .SZ_TXT_DESCR_SEP.Length);
377)
```

```
378)                        objThermal = new CHeatThermal();
379)                        szQtVolJun = shtInp.Cells[iCurRow + 1, CConstants.I_REF_SHT_IN
    P_COL_WORD1].Text;
380)
381)                        objThermal.ComponentNumber = szCompNo;
382)                        objThermal.Description = szCompDescr;
383)                        objThermal.RowNumber = iCurRow;
384)                        laszRet.Add(objThermal);
385)                    }
386)
387)                    iCurRow++;
388)                    rngCur = shtInp.Cells[iCurRow, CConstants.I_REF_SHT_INP_COL_CARD];
389)                }
390)
391)                return laszRet;
392)            }
393)        }
394)
395)        public class CHeatThermalProperties : CCompProperties
396)        {
397)            public override string CardPrefix {  get { return "201"; } }
398)
399)            public CCompProperty hthpCompositionNumber                  { get { return
     this.CreateCompProperty(" "      ,       true , null , CGlobal.Translation.TXT_HEA
    T_MAT_COMPOSITIONNO.Translate()      , EnumHydroPropertyType.WordCompNumber); } }
400)            public CCompProperty hthpCompostionType_00W1_Material       { get { return
     this.CreateCompProperty("00 W1" ,       true , null , CGlobal.Translation.TXT_HEA
    T_MAT_201MMM00_W1.Translate()        , EnumHydroPropertyType.WordAlphaMaterial); }
     }
401)            public CCompProperty hthpCompostionType_00W2_ThermalCondFlg { get { return
     this.CreateCompProperty("00 W2" ,       true , null , CGlobal.Translation.TXT_HEA
    T_MAT_201MMM00_W2.Translate()        , EnumHydroPropertyType.WordInteger); } }
402)            public CCompProperty hthpCompostionType_00W3_VolHeatCapFlg  { get { return
     this.CreateCompProperty("00 W3" ,       true , null , CGlobal.Translation.TXT_HEA
    T_MAT_201MMM00_W3.Translate()        , EnumHydroPropertyType.WordInteger); } }
403)            public CCompProperty hthpCompostionType_00                  { get { return
     this.CreateCompProperty("00 "    ,       true , null , CGlobal.Translation.TXT_HEA
    T_MAT_201MMM00.Translate()           , new CCompProperty[] { this.hthpCompostionTy
    pe_00W1_Material, this.hthpCompostionType_00W2_ThermalCondFlg, this.hthpCompostion
    Type_00W3_VolHeatCapFlg }); } }
404)            public CCompProperty hthpThermalCond_49W1_Temperature       { get { return
     this.CreateCompProperty("99 W1" ,       true , null , CGlobal.Translation.TXT_HEA
    T_MAT_201MMM49_W1TEMP.Translate()    , EnumHydroPropertyType.WordReal); } }
405)            public CCompProperty hthpThermalCond_49W2_ThermalConduct    { get { return
     this.CreateCompProperty("99 W2" ,       true , null , CGlobal.Translation.TXT_HEA
    T_MAT_201MMM49_W2THCOND.Translate()  , EnumHydroPropertyType.WordReal); } }
406)            public CCompProperty hthpThermalCond_49                     { get { return
     this.CreateCompProperty("99 "    ,  1,49,true , null , CGlobal.Translation.TXT_HEA
    T_MAT_201MMM49.Translate()           , new CCompProperty[] { this.hthpThermalCond_
    49W1_Temperature, this.hthpThermalCond_49W2_ThermalConduct }); } }
407)            public CCompProperty hthpThermalCond_49_Sngltbl             { get { return
     this.CreateCompProperty("99 "    ,  1,49,true , null , CGlobal.Translation.TXT_HEA
    T_MAT_201MMM49.Translate()           , new CCompProperty[] { this.hthpThermalCond_
    49W1_Temperature, this.hthpThermalCond_49W2_ThermalConduct, this.hthpVolHeatCap_99
    W2_HeatCap }); } }
408)            public CCompProperty hthpThermalCondFx_49W1_LowerLimTemp    { get { return
     this.CreateCompProperty("99 W1" ,       true , null , CGlobal.Translation.TXT_HEA
    T_MAT_201MMM49_W1.Translate()        , EnumHydroPropertyType.WordReal); } }
409)            public CCompProperty hthpThermalCondFx_49W2_UpperLimTemp    { get { return
     this.CreateCompProperty("99 W2" ,       true , null , CGlobal.Translation.TXT_HEA
    T_MAT_201MMM49_W2.Translate()        , EnumHydroPropertyType.WordReal); } }
410)            public CCompProperty hthpThermalCondFx_49W3_A0              { get { return
     this.CreateCompProperty("99 W3" ,       true , null , CGlobal.Translation.TXT_HEA
    T_MAT_201MMM49_W3.Translate()        , EnumHydroPropertyType.WordReal); } }
411)            public CCompProperty hthpThermalCondFx_49W4_A1              { get { return
     this.CreateCompProperty("99 W4" ,       true , null , CGlobal.Translation.TXT_HEA
    T_MAT_201MMM49_W4.Translate()        , EnumHydroPropertyType.WordReal); } }
412)            public CCompProperty hthpThermalCondFx_49W5_A2              { get { return
     this.CreateCompProperty("99 W5" ,       true , null , CGlobal.Translation.TXT_HEA
    T_MAT_201MMM49_W5.Translate()        , EnumHydroPropertyType.WordReal); } }
413)            public CCompProperty hthpThermalCondFx_49W6_A3              { get { return
     this.CreateCompProperty("99 W6" ,       true , null , CGlobal.Translation.TXT_HEA
    T_MAT_201MMM49_W6.Translate()        , EnumHydroPropertyType.WordReal); } }
414)            public CCompProperty hthpThermalCondFx_49W7_A4              { get { return
     this.CreateCompProperty("99 W7" ,       true , null , CGlobal.Translation.TXT_HEA
    T_MAT_201MMM49_W7.Translate()        , EnumHydroPropertyType.WordReal); } }
```

```
415)          public CCompProperty hthpThermalCondFx_49W8_A5              { get { return
       this.CreateCompProperty("99 W8" ,       true , null , CGlobal.Translation.TXT_HEA
      T_MAT_201MMM49_W8.Translate()        , EnumHydroPropertyType.WordReal); } }
416)          public CCompProperty hthpThermalCondFx_49W9_C               { get { return
       this.CreateCompProperty("99 W9" ,       true , null , CGlobal.Translation.TXT_HEA
      T_MAT_201MMM49_W9.Translate()        , EnumHydroPropertyType.WordReal); } }
417)          public CCompProperty hthpThermalCondFx_49                   { get { return
       this.CreateCompProperty("99 "   ,  1,49,true , null , CGlobal.Translation.TXT_HEA
      T_MAT_201MMM49.Translate()           , new CCompProperty[] { this.hthpThermalCondF
      x_49W1_LowerLimTemp, this.hthpThermalCondFx_49W2_UpperLimTemp, this.hthpThermalCon
      dFx_49W3_A0 , this.hthpThermalCondFx_49W4_A1 , this.hthpThermalCondFx_49W5_A2 , th
      is.hthpThermalCondFx_49W6_A3 , this.hthpThermalCondFx_49W7_A4 , this.hthpThermalCo
      ndFx_49W8_A5 , this.hthpThermalCondFx_49W9_C }); } }
418)          public CCompProperty hthpThermalCondMoleFr_49W1_GasName     { get { return
       this.CreateCompProperty("99 W1" ,       true , null , CGlobal.Translation.TXT_HEA
      T_MAT_201MMM49_W1GAS.Translate()     , EnumHydroPropertyType.WordAlphaMaterial, CH
      eatThermal.ListGases(false)); } }
419)          public CCompProperty hthpThermalCondMoleFr_49W2_MoleFr      { get { return
       this.CreateCompProperty("99 W2" ,       true , null , CGlobal.Translation.TXT_HEA
      T_MAT_201MMM49_W2MOLFR.Translate()   , EnumHydroPropertyType.WordInteger); } }
420)          public CCompProperty hthpThermalCondMoleFr_49               { get { return
       this.CreateCompProperty("99 "   ,  1,49,true , null , CGlobal.Translation.TXT_HEA
      T_MAT_201MMM49_MOLEFR.Translate()    , new CCompProperty[] { this.hthpThermalCondM
      oleFr_49W1_GasName, this.hthpThermalCondMoleFr_49W2_MoleFr }); } }
421)          public CCompProperty hthpVolHeatCap_99W1_Temperature        { get { return
       this.CreateCompProperty("99 W1" ,       true , null , CGlobal.Translation.TXT_HEA
      T_MAT_201MMM99_W1TEMP.Translate()    , EnumHydroPropertyType.WordReal); } }
422)          public CCompProperty hthpVolHeatCap_99W2_HeatCap            { get { return
       this.CreateCompProperty("99 W2" ,       true , null , CGlobal.Translation.TXT_HEA
      T_MAT_201MMM99_W2HEATCAP.Translate() , EnumHydroPropertyType.WordReal); } }
423)          public CCompProperty hthpVolHeatCap_99                      { get { return
       this.CreateCompProperty("99 "   , 51,99,true , null , CGlobal.Translation.TXT_HEA
      T_MAT_201MMM99.Translate()           , new CCompProperty[] { this.hthpVolHeatCap_9
      9W1_Temperature, this.hthpVolHeatCap_99W2_HeatCap }); } }
424)          public CCompProperty hthpVolHeatCapFx_99W1_LowerLimTemp     { get { return
       this.CreateCompProperty("99 W1" ,       true , null , CGlobal.Translation.TXT_HEA
      T_MAT_201MMM99_W1.Translate()        , EnumHydroPropertyType.WordReal); } }
425)          public CCompProperty hthpVolHeatCapFx_99W2_UpperLimTemp     { get { return
       this.CreateCompProperty("99 W2" ,       true , null , CGlobal.Translation.TXT_HEA
      T_MAT_201MMM99_W2.Translate()        , EnumHydroPropertyType.WordReal); } }
426)          public CCompProperty hthpVolHeatCapFx_99W3_A0               { get { return
       this.CreateCompProperty("99 W3" ,       true , null , CGlobal.Translation.TXT_HEA
      T_MAT_201MMM99_W3.Translate()        , EnumHydroPropertyType.WordReal); } }
427)          public CCompProperty hthpVolHeatCapFx_99W4_A1               { get { return
       this.CreateCompProperty("99 W4" ,       true , null , CGlobal.Translation.TXT_HEA
      T_MAT_201MMM99_W4.Translate()        , EnumHydroPropertyType.WordReal); } }
428)          public CCompProperty hthpVolHeatCapFx_99W5_A2               { get { return
       this.CreateCompProperty("99 W5" ,       true , null , CGlobal.Translation.TXT_HEA
      T_MAT_201MMM99_W5.Translate()        , EnumHydroPropertyType.WordReal); } }
429)          public CCompProperty hthpVolHeatCapFx_99W6_A3               { get { return
       this.CreateCompProperty("99 W6" ,       true , null , CGlobal.Translation.TXT_HEA
      T_MAT_201MMM99_W6.Translate()        , EnumHydroPropertyType.WordReal); } }
430)          public CCompProperty hthpVolHeatCapFx_99W7_A4               { get { return
       this.CreateCompProperty("99 W7" ,       true , null , CGlobal.Translation.TXT_HEA
      T_MAT_201MMM99_W7.Translate()        , EnumHydroPropertyType.WordReal); } }
431)          public CCompProperty hthpVolHeatCapFx_99W8_A5               { get { return
       this.CreateCompProperty("99 W8" ,       true , null , CGlobal.Translation.TXT_HEA
      T_MAT_201MMM99_W8.Translate()        , EnumHydroPropertyType.WordReal); } }
432)          public CCompProperty hthpVolHeatCapFx_99W9_C                { get { return
       this.CreateCompProperty("99 W9" ,       true , null , CGlobal.Translation.TXT_HEA
      T_MAT_201MMM99_W9.Translate()        , EnumHydroPropertyType.WordReal); } }
433)          public CCompProperty hthpVolHeatCapFx_99                    { get { return
       this.CreateCompProperty("99 "   , 51,99,true , null , CGlobal.Translation.TXT_HEA
      T_MAT_201MMM99.Translate()           , new CCompProperty[] { this.hthpVolHeatCapFx
      _99W1_LowerLimTemp, this.hthpVolHeatCapFx_99W2_UpperLimTemp, this.hthpVolHeatCapFx
      _99W3_A0 , this.hthpVolHeatCapFx_99W4_A1 , this.hthpVolHeatCapFx_99W5_A2 , this.ht
      hpVolHeatCapFx_99W6_A3 , this.hthpVolHeatCapFx_99W7_A4 , this.hthpVolHeatCapFx_99W
      8_A5 , this.hthpVolHeatCapFx_99W9_C }); } }
434)      }
435)  }
```

APÊNDICE D - Programação da tela da Estrutura de Troca de Calor

```
 1)  using System;
 2)  using System.Collections.Generic;
 3)  using System.ComponentModel;
 4)  using System.Data;
 5)  using System.Drawing;
 6)  using System.Linq;
 7)  using System.Text;
 8)  using System.Threading.Tasks;
 9)  using System.Windows.Forms;
10)
11)  namespace FastLapAddin
12)  {
13)      public partial class FrmHeat : FrmH
14)      {
15)
16)          public FrmHeat()
17)          {
18)              InitializeComponent();
19)              HeatComponent = new CHeat();
20)
21)              //Initializing events handlers
22)              txtLeftBound.Validating           += this.ValidateTxtControl;
23)              txtGapIniIntPressure.Validating   += this.ValidateTxtControl;
24)              txtIniOxThick.Validating          += this.ValidateTxtControl;
25)              cmbRefloodFlgTrip.Validating      += this.ValidateCmbControl;
26)              cmbIniTempSelected.Validating     += this.ValidateCmbControl;
27)              cmbMeshLocSelected.Validating     += this.ValidateCmbControl;
28)
29)              ctlGridIniTemp.dgrGrid.CellValidating     += this.GridNpKey_CellValida
    ting;
30)              ctlGridGapData.dgrGrid.CellValidating     += this.GridNhKey_CellValida
    ting;
31)              ctlGridIntvlMat.dgrGrid.CellValidating    += this.GridMeshIntKey_CellV
    alidating;
32)              ctlGridHeatDistrib.dgrGrid.CellValidating += this.GridMeshIntKey_CellV
    alidating;
33)              ctlGridHeatSource.dgrGrid.CellValidating  += this.GridNhKey_CellValida
    ting;
34)              ctlGridLeftBound.dgrGrid.CellValidating   += this.GridNhKey_CellValida
    ting;
35)              ctlGridRightBound.dgrGrid.CellValidating  += this.GridNhKey_CellValida
    ting;
36)          }
37)          protected void GridNhKey_CellValidating(object sender, DataGridViewCellVal
    idatingEventArgs e)
38)          {
39)              Grid_CellValidating(sender, e, (int)ctlNh.Value);
40)              e.Cancel = false;
41)          }
42)          protected void GridNpKey_CellValidating(object sender, DataGridViewCellVal
    idatingEventArgs e)
43)          {
44)              Grid_CellValidating(sender, e, (int)ctlNp.Value);
45)              e.Cancel = false;
46)          }
47)          protected void GridMeshIntKey_CellValidating(object sender, DataGridViewCe
    llValidatingEventArgs e)
48)          {
49)              Grid_CellValidating(sender, e, ((int)ctlNp.Value)- 1);
50)              e.Cancel = false;
51)          }
52)          protected void Grid_CellValidating(object sender, DataGridViewCellValidati
    ngEventArgs e
53)              , int v_iMaxPkValue
54)              , System.Windows.Forms.PictureBox v_picRelatedErrImg = null)
55)          {
56)              CUdfGrid.Grid_CellValidating(
57)                  (DataGridView)sender, e, this.HeatProperties
58)                  , ref this.p_szInvalidFields, this.btnAdd, this.tipMsg
59)                  , v_picRelatedErrImg, v_iMaxPkValue);
60)          }
61)          private void FrmHeat_Load(object sender, EventArgs e)
```

```
62)           {
63)               if (!this.DesignMode)
64)               {
65)                   this.lblTitle.Text = CGlobal.Translation.TXT_HEAT_TITLE.Translate(
      );
66)                   this.Text = CTranslation.SZ_APP_TITLE + CConstants.SZ_TXT_DESCR_SE
      P_SPC + CGlobal.Translation.TXT_HEAT_TITLE.Translate();
67)                   this.PdfNamedReference = CGlobal.Translation.SZ_PDF_HOOK_HEAT_STRU
      C;
68)                   CShtInp oshtInp = CModel.GetShtInp();
69)                   this.PrepareGridColumns();
70)                   ctlGapVolRef.ControlLayout = EnumControlLayout.VerticalLayout;
71)
72)                   //Populate Combo with TRIPS
73)                   CUdf.BindCombobox(cmbBaseHydro, CHydro.ListComponents(), "FullDesc
      ription", "ComponentNumber");
74)                   CUdf.BindCombobox(cmbRefloodFlgTrip, oshtInp.ListTrips());
75)                   CUdf.BindCombobox(cmbIniTempSelected, CHeat.ListComponents(), "Ful
      lDescription", "ComponentNumber");
76)                   CUdf.BindCombobox(cmbMeshLocSelected, CHeat.ListComponents(), "Ful
      lDescription", "ComponentNumber");
77)
78)                   this.Base_Load(sender, e);
79)               }
80)           }
81)           public CHeatProperties HeatProperties
82)           {
83)               get
84)               {
85)                   return (CHeatProperties)this.HeatComponent.HeatProperties;
86)               }
87)           }
88)           public CHeat HeatComponent
89)           {
90)               get
91)               {
92)                   CHeat objRet = null;
93)                   try
94)                   {
95)                       objRet = (CHeat)this.RelapComponent;
96)                   }
97)                   catch { }
98)                   return objRet;
99)               }
100)              set
101)              {
102)                  RelapComponent = (CComponent)value;
103)              }
104)          }
105)          public CHydro BoundaryHydroComponent
106)          {
107)              get
108)              {
109)                  CHydro objRet = null;
110)                  if (cmbBaseHydro.SelectedValue != null)
111)                  {
112)                      objRet = (CHydro)cmbBaseHydro.SelectedItem;
113)                      objRet.LeftBoundary = radHydroLeftFace.Checked;
114)                  }
115)                  return objRet;
116)              }
117)          }
118)          public EnumHeatGeometryType GeometryType
119)          {
120)              get
121)              {
122)                  return (radGeom1Rect.Checked  ? EnumHeatGeometryType.Geometry1Rect
      angular :
123)                         (radGeom2Cyl.Checked   ? EnumHeatGeometryType.Geometry2Cyli
      ndrical :
124)                         (radGeom3Spher.Checked ? EnumHeatGeometryType.Geometry3Sphe
      rical    : EnumHeatGeometryType.Geometry0NotInformed ) ) );
125)              }
126)          }
127)          public string RefloodFlag
128)          {
129)              get
```

```
130)                {
131)                    string szRet = "";
132)                    if (radRefloodFlg0.Checked)
133)                        szRet = "0";
134)                    else if (radRefloodFlg1.Checked)
135)                        szRet = "1";
136)                    else if (radRefloodFlg2.Checked)
137)                        szRet = "2";
138)                    else
139)                        szRet = (string)cmbRefloodFlgTrip.SelectedValue;
140)                    return szRet;
141)                }
142)            }
143)
144)            public override System.Windows.Forms.Control[] ControlsWithCalc
145)            {
146)                get
147)                {//Inform controls that will be attached with CALC button feature
148)                    return null;// new Control[] {txtArea, txtLength, txtVolume, txtAz
    im, txtInclination, txtElevation, txtWallRough, txtHydrDiam };
149)                }
150)            }
151)            protected override void AssociateTags()
152)            {//Code to attach right Cards onto form controls
153)                mskCompNo.Tag              = this.HeatProperties.heatpHeatNumberAndGeo
    m;
154)                txtLeftBound.Tag           = this.HeatProperties.heatpHeatStruct_000W5
    _LeftBoundCoord;
155)                txtGapIniIntPressure.Tag   = this.HeatProperties.heatpGapDt_001W1_IniG
    apPressure;
156)                txtIniOxThick.Tag          = this.HeatProperties.heatpMetalWaterReac_0
    03_IniOxThick;
157)                cmbRefloodFlgTrip.Tag      = this.HeatProperties.heatpHeatStruct_000W6
    _RefloodFlg;
158)                cmbIniTempSelected.Tag     = this.HeatProperties.heatpIniTempFlg_400_F
    lg;
159)                cmbMeshLocSelected.Tag     = this.HeatProperties.heatpMeshFlg_100W1_Me
    shLocation;
160)            }
161)            protected override void TranslateThis()
162)            {//Code to attach the right Translated messages to form labels.
163)                tipMsg.SetToolTip(btnAutoFill, CGlobal.Translation.TXT_FRM_HEAT_TIP_AU
    TOFILL.Translate());
164)                lblCompNo.Text                          = this.HeatProperties.heatpHea
    tNumber.Label;
165)                grpBaseHydro.Text                       = CGlobal.Translation.TXT_FRM_
    HEAT_LBL_AUTOFILL.Translate();
166)                lblBaseHydroConnectTo.Text              = CGlobal.Translation.TXT_FRM_
    HEAT_LBL_AUTOFILL_TO.Translate();
167)                radHydroLeftFace.Text                   = CGlobal.Translation.TXT_FRM_
    HEAT_LBL_AUTOFILL_LEFT.Translate();
168)                radHydroRightFace.Text                  = CGlobal.Translation.TXT_FRM_
    HEAT_LBL_AUTOFILL_RIGHT.Translate();
169)                grpGeomType.Text                        = this.HeatProperties.heatpHea
    tStruct_000W3_GeomType.LabelOnly;
170)                radGeom1Rect.Text                       = CGlobal.Translation.TXT_HEAT
    _GEOM_RECTANGULAR.Translate();
171)                radGeom2Cyl.Text                        = CGlobal.Translation.TXT_HEAT
    _GEOM_CYLINDRICAL.Translate();
172)                radGeom3Spher.Text                      = CGlobal.Translation.TXT_HEAT
    _GEOM_SPHERICAL.Translate();
173)                lblNh.Text                              = this.HeatProperties.heatpHea
    tStruct_000W1_Nh.Label;
174)                lblNp.Text                              = this.HeatProperties.heatpHea
    tStruct_000W2_Np.Label;
175)                tbpgMain.Text                           = CGlobal.Translation.TXT_FRM_
    HEAT_LBL_TBPGMAIN.Translate();
176)                tbpgRadial.Text                         = CGlobal.Translation.TXT_FRM_
    HEAT_LBL_TBPGRADIAL.Translate();
177)                tbpgAxialLeftBound.Text                 = CGlobal.Translation.TXT_FRM_
    HEAT_LBL_TBPGLEFT.Translate();
178)                tbpgAxialRightBound.Text                = CGlobal.Translation.TXT_FRM_
    HEAT_LBL_TBPGRIGHT.Translate();
179)                tbpgHeat.Text                           = CGlobal.Translation.TXT_FRM_
    HEAT_LBL_TBPGHEAT.Translate();
180)                tbpgGapConductModel.Text                = CGlobal.Translation.TXT_FRM_
    HEAT_LBL_TBPGGAP.Translate();
```

```
181)              grpSteadyFlag.Text                    = this.HeatProperties.heatpHea
      tStruct_000W4_SteadSttFlg.LabelOnly;
182)              radSteady0Entered.Text                = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_STEADY_0ENTERED.Translate();
183)              radSteady1Calculated.Text             = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_STEADY_1CALCULATED.Translate();
184)              lblLeftBound.Text                     = this.HeatProperties.heatpHea
      tStruct_000W5_LeftBoundCoord.Label;
185)              grpRefloodFlag.Text                   = this.HeatProperties.heatpHea
      tStruct_000W6_RefloodFlg.LabelOnly;
186)              radRefloodFlg0.Text                   = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_REFLOOD_0NONE.Translate();
187)              radRefloodFlg1.Text                   = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_REFLOOD_1EMPTY.Translate();
188)              radRefloodFlg2.Text                   = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_REFLOOD_2DRYOUT.Translate();
189)              radRefloodFlgTrip.Text                = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_REFLOOD_TRIP.Translate();
190)              grpBoundVolInd.Text                   = this.HeatProperties.heatpHea
      tStruct_000W7_BoundVolInd.LabelOnly;
191)              radReflBoundVolInd0Left.Text          = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_BOUNDVOL_0LEFT.Translate();
192)              radReflBoundVolInd1Right.Text         = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_BOUNDVOL_1RIGHT.Translate();
193)              lblAxialInterv.Text                   = this.HeatProperties.heatpHea
      tStruct_000W8_AxialInterv.Label;
194)              grpIniTempFlg.Text                    = this.HeatProperties.heatpIni
      TempFlg_400_Flg.LabelOnly;
195)              radIniTemp0This.Text                  = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_RAD_0THIS.Translate();
196)              radIniTempSelected.Text               = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_RAD_SELECTED.Translate();
197)              lblGridIniTemp.Text                   = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_GRD_INITEMP.Translate();
198)              lblMeshLoc.Text                       = this.HeatProperties.heatpMes
      hFlg_100W1_MeshLocation.Label;
199)              radMeshLoc0This.Text                  = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_RAD_0THIS.Translate();
200)              radMeshLocSelected.Text               = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_RAD_SELECTED.Translate();
201)              lblGridIntvlMat.Text                  = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_GRD_199299.Translate();
202)              lblAddLeftFormat.Text                 = this.HeatProperties.heatpLef
      tBoundAdd_800.Label;
203)              radAddLeftFormatNone.Text             = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_ADDBND_NONE.Translate();
204)              radAddLeftFormat0Nine.Text            = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_ADDBND_0NINE.Translate();
205)              radAddLeftFormat1Twelve.Text          = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_ADDBND_1TWELV.Translate();
206)              radAddLeftFormat2Thirteen.Text        = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_ADDBND_2THIRT.Translate();
207)              lblAddRightFormat.Text                = this.HeatProperties.heatpRig
      htBoundAdd_900.Label;
208)              radAddRightFormatNone.Text            = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_ADDBND_NONE.Translate();
209)              radAddRightFormat0Nine.Text           = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_ADDBND_0NINE.Translate();
210)              radAddRightFormat1Twelve.Text         = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_ADDBND_1TWELV.Translate();
211)              radAddRightFormat2Thirteen.Text       = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_ADDBND_2THIRT.Translate();
212)              lblGridHeatSource.Text                = this.HeatProperties.heatpHea
      tSrc_799.Label;
213)              lblGridHeatDistrib.Text               = CGlobal.Translation.TXT_FRM_
      HEAT_LBL_GRD_HEATDISTR.Translate();
214)              chkUseFission.Text                    = this.HeatProperties.heatpDec
      ay_300_DkHeat.LabelOnly;
215)              lblGridGapData.Text                   = this.HeatProperties.heatpGap
      DefDt_099.Label;
216)              grpGapData.Text                       = this.HeatProperties.heatpGap
      Dt_001.LabelOnly;
217)              lblGapIniIntPressure.Text             = this.HeatProperties.heatpGap
      Dt_001W1_IniGapPressure.Label;
218)              lblGapVolRef.Text                     = this.HeatProperties.heatpGap
      Dt_001W2_GapConductVol.Label;
```

```
219)             lblGapDeformationModel.Text             = this.HeatProperties.heatpFue
     lDeform_004_FormLossF.Label;
220)             radGapDeformFlag0.Text                  = CGlobal.Translation.TXT_FRM_
     HEAT_LBL_DEF_0NONE.Translate();
221)             radGapDeformFlag1.Text                  = CGlobal.Translation.TXT_FRM_
     HEAT_LBL_DEF_1RUPT.Translate();
222)             lblMetalWaterReaction.Text              = this.HeatProperties.heatpMet
     alWaterReac_003_IniOxThick.Label;
223)         }
224)
225)         private void PrepareGridColumns()
226)         {
227)             ctlGridIniTemp.CreateColumnKeyOnGrid(     "colNp"          , CGlobal.T
     ranslation.TXT_HEAT_PROP_NP.Translate());
228)             ctlGridIniTemp.AddPropertyOnGrid(      this.HeatProperties.heatpIniTem
     pDt_499);
229)
230)             ctlGridGapData.AddPropertiesOnGrid(    this.HeatProperties.heatpGapDef
     Dt_099, false, "colNh" , CGlobal.Translation.TXT_HEAT_PROP_NH.Translate());
231)
232)             ctlGridIntvlMat.CreateColumnKeyOnGrid(    "colMeshIntervNo", CGlobal.T
     ranslation.TXT_HEAT_PROP_MESHINTERVAL.Translate());
233)             ctlGridIntvlMat.AddPropertyOnGrid(     this.HeatProperties.heatpMeshIn
     t_199_MeshInterval);
234)             ctlGridIntvlMat.AddPropertyOnGrid(     this.HeatProperties.heatpCompos
     ition_299_CompositionNo);
235)
236)             ctlGridHeatDistrib.CreateColumnKeyOnGrid( "colMeshIntervNo", CGlobal.T
     ranslation.TXT_HEAT_PROP_MESHINTERVAL.Translate());
237)             ctlGridHeatDistrib.AddPropertyOnGrid(  this.HeatProperties.heatpHeatDi
     str_399_SourceValOrGamma);
238)
239)             ctlGridHeatSource.AddPropertiesOnGrid( this.HeatProperties.heatpHeatSr
     c_799, false, "colNh"   , CGlobal.Translation.TXT_HEAT_PROP_NH.Translate());
240)
241)             ctlGridLeftBound.AddPropertiesOnGrid(  this.HeatProperties.heatpLeftBo
     und_599, false, "colNh" , CGlobal.Translation.TXT_HEAT_PROP_NH.Translate());
242)             ctlGridLeftBound.AddPropertiesOnGrid(  this.HeatProperties.heatpLeftBo
     undAdd12Word_899, true);
243)             ctlGridLeftBound.AddPropertiesOnGrid(  this.HeatProperties.heatpLeftBo
     undAdd13Word_899, true);
244)
245)             ctlGridRightBound.AddPropertiesOnGrid( this.HeatProperties.heatpRightB
     ound_699.aobjWords, false, "colNh" , CGlobal.Translation.TXT_HEAT_PROP_NH.Translat
     e());
246)             ctlGridRightBound.AddPropertiesOnGrid( this.HeatProperties.heatpRightB
     oundAdd12Word_999, true);
247)             ctlGridRightBound.AddPropertiesOnGrid( this.HeatProperties.heatpRightB
     oundAdd13Word_999, true);
248)
249)             //Adding CALC functionality onto grids
250)             ctlGridIniTemp.AuxiliaryButton     = this.btnCalc;
251)             ctlGridGapData.AuxiliaryButton     = this.btnCalc;
252)             ctlGridIntvlMat.AuxiliaryButton    = this.btnCalc;
253)             ctlGridHeatDistrib.AuxiliaryButton = this.btnCalc;
254)             ctlGridHeatSource.AuxiliaryButton  = this.btnCalc;
255)             ctlGridLeftBound.AuxiliaryButton   = this.btnCalc;
256)             ctlGridRightBound.AuxiliaryButton  = this.btnCalc;
257)
258)             this.ShowProperBoundColumns(true, ctlGridLeftBound , null, this.HeatPr
     operties.heatpLeftBound_599.szCardName );
259)             this.ShowProperBoundColumns(true, ctlGridRightBound, null, this.HeatPr
     operties.heatpRightBound_699.szCardName);
260)         }
261)
262)         private void radGeom1Rect_CheckedChanged(object sender, EventArgs e)
263)         {
264)             picMeshDiagr.Image = Properties.Resources.res3dHeatRectangular3dWH;
265)             picMeshDiagr.Update();
266)         }
267)
268)         private void radGeom2Cyl_CheckedChanged(object sender, EventArgs e)
269)         {
270)             picMeshDiagr.Image = Properties.Resources.res3dHeatCylinder3dWH;
271)             picMeshDiagr.Update();
272)
```

```
273)                //Cylindrical Geometry Must be specified when gap conductance model is
     used.
274)                EnableGapConductanceControls(radGeom2Cyl.Checked);
275)            }
276)
277)            private void radGeom3Spher_CheckedChanged(object sender, EventArgs e)
278)            {
279)                picMeshDiagr.Image = Properties.Resources.res3dHeatSpherical3d;
280)                picMeshDiagr.Update();
281)
282)                //If using Spherical Geometry, Reflood calculation cannot be used, so,
     disable it.
283)                EnableRefloodControls(!radGeom3Spher.Checked);
284)            }
285)
286)            public void EnableRefloodControls(bool v_blnEnable)
287)            {
288)                radRefloodFlg1.Enabled = v_blnEnable;
289)                radRefloodFlg2.Enabled = v_blnEnable;
290)                radRefloodFlgTrip.Enabled = v_blnEnable;
291)            }
292)            public void EnableGapConductanceControls(bool v_blnEnable)
293)            {
294)                txtGapIniIntPressure.Enabled = v_blnEnable;
295)                ctlGapVolRef.Enabled = v_blnEnable;
296)                ctlGridGapData.Enabled = v_blnEnable;
297)            }
298)
299)            private void radRefloodFlgTrip_CheckedChanged(object sender, EventArgs e)
300)            {
301)                if (!radRefloodFlgTrip.Checked) //If Reflood is not allowed, unselect
     any value that may be previously selected on this Trip Combo
302)                    cmbRefloodFlgTrip.SelectedIndex = 0;
303)            }
304)            private void cmbRefloodFlgTrip_SelectedIndexChanged(object sender, EventAr
     gs e)
305)            {
306)                if (cmbRefloodFlgTrip.SelectedIndex > 0)
307)                    radRefloodFlgTrip.Checked = true;
308)            }
309)            private void cmbMeshLocSelected_SelectedIndexChanged(object sender, EventA
     rgs e)
310)            {
311)                if (cmbMeshLocSelected.SelectedIndex > 0)
312)                    radMeshLocSelected.Checked = true;
313)            }
314)
315)            private void radIniTemp0This_CheckedChanged(object sender, EventArgs e)
316)            {
317)                ctlGridIniTemp.Enabled = radIniTemp0This.Checked;
318)                if(radIniTemp0This.Checked)
319)                    cmbRefloodFlgTrip.SelectedIndex = 0;
320)            }
321)            private void cmbIniTempSelected_SelectedIndexChanged(object sender, EventA
     rgs e)
322)            {
323)                if (cmbIniTempSelected.SelectedIndex > 0)
324)                    radIniTempSelected.Checked = true;
325)            }
326)            private void radMeshLoc0This_CheckedChanged(object sender, EventArgs e)
327)            {
328)                ctlGridIntvlMat.Enabled = radMeshLoc0This.Checked;
329)                ctlGridHeatDistrib.Enabled = radMeshLoc0This.Checked;
330)                chkUseFission.Enabled = radMeshLoc0This.Checked;
331)            }
332)
333)            protected override void btnAdd_Click(object sender, EventArgs e)
334)            {
335)                if (p_szInvalidFields == "")
336)                {
337)                    CShtInp ocshtInp = CModel.GetShtInp();
338)                    HeatComponent.Draw(this.mskCompNo.Text, this.txtDescr.Text, this.G
     eometryType, this.BoundaryHydroComponent);
339)                    //Erase old versions of component from Input
340)                    ocshtInp.RemoveComponentFromInp(this.mskCompNo.Text, CConstants.SZ
     _INP_SEARCHMASK_MISCANY, this.HeatProperties.heatpHeatStruct_000.CardPrefix);
```

```
341)                    //Add recently configured component to Input, rewriting Card 1CCCG
       000 - General Heat Structure Data
342)                    ocshtInp.WriteCard(
343)                          CGlobal.Translation.TXT_FRM_HEAT_LBL_TITLE_SFX.Translate() +
        " " + this.HeatComponent.ComponentNumber + CConstants.SZ_TXT_DESCR_SEP + txtDescr
       .Text
344)                        , this.HeatComponent.ComponentNumber
345)                        , ""
346)                        , this.HeatProperties.heatpHeatStruct_000
347)                        , this.ctlNh.ValueAsText
348)                        , this.ctlNp.ValueAsText
349)                        , ((int)this.GeometryType).ToString()
350)                        , (radSteady0Entered.Checked)?"0":"1"
351)                        , this.txtLeftBound.Text
352)                        , this.RefloodFlag
353)                        , (radReflBoundVolInd0Left.Checked)?"0":"1"
354)                        , Math.Pow(2, trkAxialInterv.Value).ToString());
355)                    if (this.txtGapIniIntPressure.Text != "") //Gap Conductance Model
       Will be Used
356)                    {
357)                        ocshtInp.WriteCard(
358)                              "*"
359)                            , this.HeatComponent.ComponentNumber
360)                            , ""
361)                            , this.HeatProperties.heatpGapDt_001
362)                            , this.txtGapIniIntPressure.Text
363)                            , this.ctlGapVolRef.SelectedComponent.GetHydroRefVolume())
       ;
364)                        ocshtInp.WriteCardsFromGridWhenColAreCards(
365)                              "*"
366)                            , this.HeatComponent.ComponentNumber
367)                            , ""
368)                            , new CCompProperty[] { this.HeatProperties.heatpGapDefDt_
       099 }
369)                            , this.ctlGridGapData.dgrGrid);
370)                    }
371)                    if(this.txtIniOxThick.Text != "")
372)                        ocshtInp.WriteCard(
373)                              "*"
374)                            , this.HeatComponent.ComponentNumber
375)                            , ""
376)                            , this.HeatProperties.heatpMetalWaterReac_003_IniOxThick
377)                            , this.txtIniOxThick.Text);
378)                    if(this.radGapDeformFlag1.Checked)
379)                        ocshtInp.WriteCard(
380)                              "*"
381)                            , this.HeatComponent.ComponentNumber
382)                            , ""
383)                            , this.HeatProperties.heatpFuelDeform_004_FormLossF
384)                            , this.radGapDeformFlag1.Checked?"1":"0");
385)                    ocshtInp.WriteCard(
386)                            "*"
387)                        , this.HeatComponent.ComponentNumber
388)                        , ""
389)                        , this.HeatProperties.heatpMeshFlg_100
390)                        , this.radGapDeformFlag1.Checked?((CHeat)this.cmbMeshLocSelect
       ed.SelectedItem).ComponentNumber:"0"
391)                        , "1");
392)                    if (radMeshLoc0This.Checked)
393)                    {
394)                        ocshtInp.WriteCardsFromGridWhenColAreCards(
395)                              "*"
396)                            , this.HeatComponent.ComponentNumber
397)                            , ""
398)                            , new CCompProperty[] { this.HeatProperties.heatpMeshInt_1
       99_MeshInterval, this.HeatProperties.heatpComposition_299_CompositionNo }
399)                            , this.ctlGridIntvlMat.dgrGrid);
400)                    }
401)                    if (chkUseFission.Checked)
402)                    {
403)                        ocshtInp.WriteCard(
404)                              "*"
405)                            , this.HeatComponent.ComponentNumber
406)                            , ""
407)                            , this.HeatProperties.heatpDecay_300_DkHeat
408)                            , CConstants.SZ_HEAT_DKHEAT);
409)                    }
```

```
410)                    if (radGapDeformFlag0.Checked)
411)                    {
412)                        ocshtInp.WriteCardsFromGridWhenColAreCards(
413)                              "*"
414)                            , this.HeatComponent.ComponentNumber
415)                            , ""
416)                            , new CCompProperty[] { this.HeatProperties.heatpHeatDistr
    _399_SourceValOrGamma }
417)                            , this.ctlGridHeatDistrib.dgrGrid);
418)                    }
419)                    ocshtInp.WriteCard(
420)                            "*"
421)                        , this.HeatComponent.ComponentNumber
422)                        , ""
423)                        , this.HeatProperties.heatpIniTempFlg_400_Flg
424)                        , radIniTempSelected.Checked?((CHeat)cmbIniTempSelected.Select
    edItem).ComponentNumber:"0");
425)                    if (radIniTemp0This.Checked)
426)                    {
427)                        ocshtInp.WriteCardsFromGridWhenColAreCards(
428)                              "*"
429)                            , this.HeatComponent.ComponentNumber
430)                            , ""
431)                            , new CCompProperty[] { this.HeatProperties.heatpIniTempDt
    _499 }
432)                            , this.ctlGridIniTemp.dgrGrid);
433)                    }
434)                    ocshtInp.WriteCardsFromGridWhenColAreCards(
435)                          "*"
436)                        , this.HeatComponent.ComponentNumber
437)                        , ""
438)                        , new CCompProperty[] { this.HeatProperties.heatpLeftBound_599
     }
439)                        , this.ctlGridLeftBound.dgrGrid);
440)                    ocshtInp.WriteCardsFromGridWhenColAreCards(
441)                          "*"
442)                        , this.HeatComponent.ComponentNumber
443)                        , ""
444)                        , new CCompProperty[] { this.HeatProperties.heatpRightBound_69
    9 }
445)                        , this.ctlGridRightBound.dgrGrid);
446)                    ocshtInp.WriteCardsFromGridWhenColAreCards(
447)                          "*"
448)                        , this.HeatComponent.ComponentNumber
449)                        , ""
450)                        , new CCompProperty[] { this.HeatProperties.heatpHeatSrc_799 }
451)                        , this.ctlGridHeatSource.dgrGrid);
452)
453)                    CCompProperty aobjPropAddReg;
454)                    string szPropAddReg = "";
455)                    if (!radAddLeftFormatNone.Checked)
456)                    {
457)                        if (radAddLeftFormat1Twelve.Checked)
458)                        {
459)                            szPropAddReg = "1";
460)                            aobjPropAddReg = this.HeatProperties.heatpLeftBoundAdd12Wo
    rd_899;
461)                        }
462)                        else if (radAddLeftFormat2Thirteen.Checked)
463)                        {
464)                            szPropAddReg = "2";
465)                            aobjPropAddReg = this.HeatProperties.heatpLeftBoundAdd13Wo
    rd_899;
466)                        }
467)                        else
468)                        {
469)                            szPropAddReg = "0";
470)                            aobjPropAddReg = this.HeatProperties.heatpLeftBoundAdd9Wor
    d_899;
471)                        }
472)                        ocshtInp.WriteCard(
473)                                "*"
474)                            , this.HeatComponent.ComponentNumber
475)                            , ""
476)                            , this.HeatProperties.heatpLeftBoundAdd_800
477)                            , szPropAddReg);
478)                        ocshtInp.WriteCardsFromGridWhenColAreCards(
```

```
                            "*"
                        , this.HeatComponent.ComponentNumber
                        , ""
                        , new CCompProperty[] { aobjPropAddReg }
                        , this.ctlGridLeftBound.dgrGrid);
                }
                if (!radAddRightFormatNone.Checked)
                {
                    if (radAddRightFormat1Twelve.Checked)
                    {
                        szPropAddReg = "1";
                        aobjPropAddReg = this.HeatProperties.heatpRightBoundAdd12W
ord_999;
                    }
                    else if (radAddRightFormat2Thirteen.Checked)
                    {
                        szPropAddReg = "2";
                        aobjPropAddReg = this.HeatProperties.heatpRightBoundAdd13W
ord_999;
                    }
                    else
                    {
                        szPropAddReg = "0";
                        aobjPropAddReg = this.HeatProperties.heatpRightBoundAdd9Wo
rd_999;
                    }
                    ocshtInp.WriteCard(
                            "*"
                        , this.HeatComponent.ComponentNumber
                        , ""
                        , this.HeatProperties.heatpRightBoundAdd_900
                        , szPropAddReg);
                    ocshtInp.WriteCardsFromGridWhenColAreCards(
                            "*"
                        , this.HeatComponent.ComponentNumber
                        , ""
                        , new CCompProperty[] { aobjPropAddReg }
                        , this.ctlGridRightBound.dgrGrid);
                }
                this.Close();
            }
            else
            {
                CUdf.MsgBox(CGlobal.Translation.TXT_ERR_HYDRO_INVALID_FORM.Transla
te(), MessageBoxIcon.Exclamation);
            }
        }
        private void radRefloodFlg0_CheckedChanged(object sender, EventArgs e)
        {
            if(radRefloodFlg0.Checked) //No Reflood is Specified
            {
                ctlNp.Minimum = 2;
                ctlNp.MinText = "2";
            }
            else
            {
                ctlNp.Minimum = 3;
                ctlNp.MinText = "3";
            }
        }

        private void ShowProperBoundColumns(bool v_blnFormatChecked, CtlGrid v_ctl
Grid, CCompProperty[] v_aobjProp, string v_szCardToKeepVisible)
        {
            if (v_blnFormatChecked)
                v_ctlGrid.ShowSelectedProperties(v_aobjProp, v_szCardToKeepVisible
);
        }

        private void radAddLeftFormatNone_CheckedChanged(object sender, EventArgs
e)
        {
            if (!this.DesignMode)
            {
                ShowProperBoundColumns(radAddLeftFormatNone.Checked, ctlGridLeftBo
und, null
                    , this.HeatProperties.heatpLeftBound_599.szCardName);
```

```
548)             }
549)         }
550)         private void radAddLeftFormat0Nine_CheckedChanged(object sender, EventArgs
      e)
551)         {
552)             if (!this.DesignMode)
553)             {
554)                 ShowProperBoundColumns(radAddLeftFormat0Nine.Checked, ctlGridLeftB
     ound
555)                     , this.HeatProperties.heatpLeftBoundAdd9Word_899.aobjWords
556)                     , this.HeatProperties.heatpLeftBound_599.szCardName);
557)             }
558)         }
559)         private void radAddLeftFormat1Twelve_CheckedChanged(object sender, EventAr
     gs e)
560)         {
561)             if (!this.DesignMode)
562)             {
563)                 ShowProperBoundColumns(radAddLeftFormat1Twelve.Checked, ctlGridLef
     tBound
564)                     , this.HeatProperties.heatpLeftBoundAdd12Word_899.aobjWords
565)                     , this.HeatProperties.heatpLeftBound_599.szCardName);
566)             }
567)         }
568)         private void radAddLeftFormat2Thirteen_CheckedChanged(object sender, Event
     Args e)
569)         {
570)             if (!this.DesignMode)
571)             {
572)                 ShowProperBoundColumns(radAddLeftFormat2Thirteen.Checked, ctlGridL
     eftBound
573)                     , this.HeatProperties.heatpLeftBoundAdd13Word_899.aobjWords
574)                     , this.HeatProperties.heatpLeftBound_599.szCardName);
575)             }
576)         }
577)         private void radAddRightFormatNone_CheckedChanged(object sender, EventArgs
      e)
578)         {
579)             if (!this.DesignMode)
580)             {
581)                 ShowProperBoundColumns(radAddRightFormatNone.Checked, ctlGridRight
     Bound, null
582)                     , this.HeatProperties.heatpRightBound_699.szCardName);
583)             }
584)         }
585)         private void radAddRightFormat0Nine_CheckedChanged(object sender, EventArg
     s e)
586)         {
587)             if (!this.DesignMode)
588)             {
589)                 ShowProperBoundColumns(radAddRightFormat0Nine.Checked, ctlGridRigh
     tBound
590)                     , this.HeatProperties.heatpRightBoundAdd9Word_999.aobjWords
591)                     , this.HeatProperties.heatpRightBound_699.szCardName);
592)             }
593)         }
594)         private void radAddRightFormat1Twelve_CheckedChanged(object sender, EventA
     rgs e)
595)         {
596)             if (!this.DesignMode)
597)             {
598)                 ShowProperBoundColumns(radAddRightFormat1Twelve.Checked, ctlGridRi
     ghtBound
599)                     , this.HeatProperties.heatpRightBoundAdd12Word_999.aobjWords
600)                     , this.HeatProperties.heatpRightBound_699.szCardName);
601)             }
602)         }
603)         private void radAddRightFormat2Thirteen_CheckedChanged(object sender, Even
     tArgs e)
604)         {
605)             if (!this.DesignMode)
606)             {
607)                 ShowProperBoundColumns(radAddRightFormat2Thirteen.Checked, ctlGrid
     RightBound
608)                     , this.HeatProperties.heatpRightBoundAdd13Word_999.aobjWords
609)                     , this.HeatProperties.heatpRightBound_699.szCardName);
610)             }
```

```
611)            }
612)
613)            private void cmbBaseHydro_SelectedIndexChanged(object sender, EventArgs e)
614)            {
615)
616)            }
617)
618)            private void tabProp_Selected(object sender, TabControlEventArgs e)
619)            {
620)                btnCalc.Visible = false;
621)            }
622)        }
623)    }
```

APÊNDICE E - Programação da superclasse CComponent

```
 1)  using System;
 2)  using System.Collections.Generic;
 3)  using System.Linq;
 4)  using System.Text;
 5)  using System.Threading.Tasks;
 6)  using Office = Microsoft.Office.Core;
 7)  using Excel = Microsoft.Office.Interop.Excel;
 8)  using Microsoft.Office.Tools.Excel;
 9)  using System.Windows.Forms;
10)
11)  namespace FastLapAddin
12)  {
13)      public class CBaseShape
14)      {
15)          public string mszType;
16)          public Office.MsoAutoShapeType mmsoAutoShapeType;
17)          public long mlShapeRotation;
18)          public bool mblnDash;
19)          public double mdblWidth;
20)          public double mdblHeight;
21)          public string[] maszDescription;
22)
23)          public static CBaseShape ShapeHydroSnglvol  = new CBaseShape(CConstants.SZ
     _HYDRO_SNGLVOL , Office.MsoAutoShapeType.msoShapeFlowchartProcess                ,
        0, false, 1  , 1  , CGlobal.Translation.TXT_HYDRO_LBL_SNGLVOL  );
24)          public static CBaseShape ShapeHydroTmdpvol  = new CBaseShape(CConstants.SZ
     _HYDRO_TMDPVOL , Office.MsoAutoShapeType.msoShapeFlowchartProcess                ,
        0, true , 1  , 1  , CGlobal.Translation.TXT_HYDRO_LBL_TMDPVOL  );
25)          public static CBaseShape ShapeHydroSngljun  = new CBaseShape(CConstants.SZ
     _HYDRO_SNGLJUN , Office.MsoAutoShapeType.msoShapeOval                            ,
        0, false, .25, 1  , CGlobal.Translation.TXT_HYDRO_LBL_SNGLJUN  );
26)          public static CBaseShape ShapeHydroTmdpjun  = new CBaseShape(CConstants.SZ
     _HYDRO_TMDPJUN , Office.MsoAutoShapeType.msoShapeOval                            ,
        0, true , .25, 1  , CGlobal.Translation.TXT_HYDRO_LBL_TMDPJUN  );
27)          public static CBaseShape ShapeHydroPipe     = new CBaseShape(CConstants.SZ
     _HYDRO_PIPE    , Office.MsoAutoShapeType.msoShapeFlowchartDirectAccessStorage    ,
        0, false, 1  , .3 , CGlobal.Translation.TXT_HYDRO_LBL_PIPE     );
28)          public static CBaseShape ShapeHydroAnnulus  = new CBaseShape(CConstants.SZ
     _HYDRO_ANNULUS , Office.MsoAutoShapeType.msoShapeFlowchartMagneticDisk           ,
        0, false, .25, 1  , CGlobal.Translation.TXT_HYDRO_LBL_ANNULUS  );
29)          public static CBaseShape ShapeHydroPrizer   = new CBaseShape(CConstants.SZ
     _HYDRO_PRIZER  , Office.MsoAutoShapeType.msoShapeFlowchartTerminator             ,
         -90, false, .25, 1  , CGlobal.Translation.TXT_HYDRO_LBL_PRIZER   );
30)          public static CBaseShape ShapeHydroCanchan  = new CBaseShape(CConstants.SZ
     _HYDRO_CANCHAN , Office.MsoAutoShapeType.msoShapeFlowchartDirectAccessStorage    ,
        0, false, 1  , .25, CGlobal.Translation.TXT_HYDRO_LBL_CANCHAN  );
31)          public static CBaseShape ShapeHydroBranch   = new CBaseShape(CConstants.SZ
     _HYDRO_BRANCH  , Office.MsoAutoShapeType.msoShapeFlowchartSummingJunction        ,
        0, false, .3 , .3 , CGlobal.Translation.TXT_HYDRO_LBL_BRANCH   );
32)          public static CBaseShape ShapeHydroSeparatr = new CBaseShape(CConstants.SZ
     _HYDRO_SEPARATR, Office.MsoAutoShapeType.msoShapeFlowchartStoredData             ,
          90, false, .25, 1  , CGlobal.Translation.TXT_HYDRO_LBL_SEPARATR );
33)          public static CBaseShape ShapeHydroJetmixer = new CBaseShape(CConstants.SZ
     _HYDRO_JETMIXER, Office.MsoAutoShapeType.msoShapeCross                           ,
        0, false, .6 , .6 , CGlobal.Translation.TXT_HYDRO_LBL_JETMIXER );
34)          public static CBaseShape ShapeHydroTurbine  = new CBaseShape(CConstants.SZ
     _HYDRO_TURBINE , Office.MsoAutoShapeType.msoShapeFlowchartManualOperation        ,
         -90, false, 1  , .6 , CGlobal.Translation.TXT_HYDRO_LBL_TURBINE  );
35)          public static CBaseShape ShapeHydroEccmix   = new CBaseShape(CConstants.SZ
     _HYDRO_ECCMIX  , Office.MsoAutoShapeType.msoShapePlaque                          ,
        0, false, .6 , .6 , CGlobal.Translation.TXT_HYDRO_LBL_ECCMIX   );
36)          public static CBaseShape ShapeHydroValve    = new CBaseShape(CConstants.SZ
     _HYDRO_VALVE   , Office.MsoAutoShapeType.msoShapeFlowchartCollate                ,
         -90, false, .4 , .9 , CGlobal.Translation.TXT_HYDRO_LBL_VALVE    );
37)          public static CBaseShape ShapeHydroPump     = new CBaseShape(CConstants.SZ
     _HYDRO_PUMP    , Office.MsoAutoShapeType.msoShapeFlowchartSequentialAccessStorage,
        0, false, .8 , .8 , CGlobal.Translation.TXT_HYDRO_LBL_PUMP     );
38)          public static CBaseShape ShapeHydroMtpljun  = new CBaseShape(CConstants.SZ
     _HYDRO_MTPLJUN , Office.MsoAutoShapeType.msoShapeQuadArrow                       ,
        0, false, .8 , .8 , CGlobal.Translation.TXT_HYDRO_LBL_MTPLJUN  );
```

```
39)          public static CBaseShape ShapeHydroAccum    = new CBaseShape(CConstants.SZ
     _HYDRO_ACCUM   , Office.MsoAutoShapeType.msoShapeFlowchartManualOperation        ,
        0, false, .7 , 1   , CGlobal.Translation.TXT_HYDRO_LBL_ACCUM    );
40)          public static CBaseShape ShapeHeat          = new CBaseShape(CConstants.SZ
     _HEAT          , Office.MsoAutoShapeType.msoShapeFlowchartProcess                ,
        0, false, 1   , 1   , CGlobal.Translation.TXT_HEAT_LBL_HEAT  );
41)
42)          public static CBaseShape[] BaseTypes =
43)              {     ShapeHydroSnglvol
44)                  , ShapeHydroTmdpvol
45)                  , ShapeHydroSngljun
46)                  , ShapeHydroTmdpjun
47)                  , ShapeHydroPipe
48)                  , ShapeHydroAnnulus
49)                  , ShapeHydroPrizer
50)                  , ShapeHydroCanchan
51)                  , ShapeHydroBranch
52)                  , ShapeHydroSeparatr
53)                  , ShapeHydroJetmixer
54)                  , ShapeHydroTurbine
55)                  , ShapeHydroEccmix
56)                  , ShapeHydroValve
57)                  , ShapeHydroPump
58)                  , ShapeHydroMtpljun
59)                  , ShapeHydroAccum
60)                  , ShapeHeat
61)              };
62)          public CBaseShape(string vszType, Office.MsoAutoShapeType vmsoAutoShapeTyp
     e, long vlShapeRotation, bool vblnDash, double vdblWidth, double vdblHeight, strin
     g[] vaszDescr)
63)          {
64)              this.LoadProperties(vszType, vmsoAutoShapeType, vlShapeRotation, vblnD
     ash, vdblWidth, vdblHeight, vaszDescr);
65)          }
66)          public CBaseShape(string vszType)
67)          {
68)              foreach (CBaseShape obj in CBaseShape.BaseTypes)
69)              {
70)                  if (vszType == obj.mszType)
71)                  {
72)                      this.LoadProperties(obj.mszType, obj.mmsoAutoShapeType, obj.ml
     ShapeRotation, obj.mblnDash, obj.mdblWidth, obj.mdblHeight, obj.maszDescription);
73)                      break;
74)                  }
75)              }
76)              return;
77)          }
78)          public string Name
79)          {
80)              get
81)              {
82)                  return this.maszDescription.Translate();
83)              }
84)          }
85)          void LoadProperties(string v_szType, Office.MsoAutoShapeType v_msoAutoShap
     eType, long v_lShapeRotation, bool v_blnDash, double v_dblWidth, double v_dblHeigh
     t, string[] v_aszDescr)
86)          {
87)              this.mszType = v_szType;
88)              this.mmsoAutoShapeType = v_msoAutoShapeType;
89)              this.mlShapeRotation = v_lShapeRotation;
90)              this.mblnDash = v_blnDash;
91)              this.mdblWidth = v_dblWidth;
92)              this.mdblHeight = v_dblHeight;
93)              this.maszDescription = v_aszDescr;
94)          }
95)      }
96)
97)      public class CComponent
98)      {
99)          public CBaseShape BaseShape;
100)         public string Description = "";
101)         public string ComponentNumber = "";
102)         public CCompProperties PropertyCards;
103)         public int RowNumber = -1;
104)         public int NumberOfVolumes = 0;
105)         public CHydroSelectedVolume[] FromComponents = null;
```

```
106)          public CHydroSelectedVolume FromComponent
107)          {
108)              get { return (this.FromComponents!=null)?this.FromComponents[0]:null;
      }
109)              set { this.FromComponents = new CHydroSelectedVolume[] { value as CHyd
      roSelectedVolume }; }
110)          }
111)          public CHydroSelectedVolume[] ToComponents = null;
112)          public CHydroSelectedVolume ToComponent
113)          {
114)              get { return (this.ToComponents!=null)?this.ToComponents[0]:null; }
115)              set { this.ToComponents = new CHydroSelectedVolume[] { value as CHydro
      SelectedVolume }; }
116)          }
117)          public CHydro BoundaryHydroComponent;
118)          public bool p_leftboundary = true;
119)          public bool LeftBoundary { get { return this.p_leftboundary; } set { this.
      p_leftboundary = value; } }
120)          public bool RightBoundary { get { return !this.p_leftboundary; } set { thi
      s.p_leftboundary = !value; } }
121)          protected string p_name = "";
122)          public virtual string Name { get { return p_name; } set { this.p_name = va
      lue; } }
123)
124)          public virtual string FullDescription
125)          {
126)              get
127)              {
128)                  string szRet = "";
129)                  if (this.ComponentNumber != "" && this.Name != "")
130)                      szRet = this.ComponentNumber + CConstants.SZ_INP_DASH + this.N
      ame + CConstants.SZ_TXT_DESCR_SEP + this.Description;
131)                  else if(this.ComponentNumber != "" && this.Name == "" && this.Desc
      ription != "")
132)                      szRet = this.ComponentNumber + CConstants.SZ_TXT_DESCR_SEP + t
      his.Description;
133)                  return szRet;
134)              }
135)          }
136)          public virtual void Draw()
137)          {
138)              CBaseShape objShpBaseCfg = this.BaseShape;
139)
140)              double dblW = objShpBaseCfg.mdblWidth;
141)              double dblH = objShpBaseCfg.mdblHeight;
142)              double dblDefaultSize = CGlobal.DefaultComponentsSizeInPoints;
143)              bool blnGetPositionFromPrevious = false;
144)              float flPrevW = 0;
145)              float flPrevH = 0;
146)              float flPrevRotation = 0;
147)              float flRotationSign = -1;
148)              float flDefaultX = CConstants.L_DEFAULT_SHAPE_LEFT;
149)              float flDefaultY = CConstants.L_DEFAULT_SHAPE_TOP;
150)              double dblX = 0;
151)              double dblY = 0;
152)              Excel.Shape shpVol;
153)              string[] aszVolsNames = new string[this.NumberOfVolumes + 1];
154)              Excel.Shape shpComp;
155)              Excel.Shape shpCompLabel;
156)              Excel.Worksheet shtCanvas;
157)
158)              shtCanvas = CModel.TargetExcelApp.ActiveSheet;
159)              CDiagram.UnprotectDiagram(shtCanvas);
160)
161)              //Before starting, check if this is a new component, or if it is repla
      cing/updating a existing one
162)              //If replacing a existing component, Default X and Y positions will be
       the Left/Top of the actual component.
163)              Excel.Shape shpExisting = null;
164)              try
165)              {
166)                  shpExisting = CModel.TargetExcelApp.ActiveSheet.Shapes(this.Compon
      entNumber + this.Name);
167)              } catch { }
168)              if(shpExisting != null)
169)              {
170)                  flDefaultX = shpExisting.Top;
```

```
171)                    flDefaultY = shpExisting.Left;
172)                    flPrevW = shpExisting.Width;
173)                    flPrevH = shpExisting.Height;
174)                    flPrevRotation = shpExisting.Rotation;
175)                    blnGetPositionFromPrevious = true;
176)                    shpExisting.Delete();
177)                }
178)
179)                if (objShpBaseCfg.mlShapeRotation == -90
180)                    || objShpBaseCfg.mlShapeRotation == 90
181)                    || objShpBaseCfg.mlShapeRotation == 270)
182)                {
183)                    dblH = objShpBaseCfg.mdblWidth;
184)                    dblW = objShpBaseCfg.mdblHeight;
185)                }
186)                else
187)                {
188)                    dblW = objShpBaseCfg.mdblWidth;
189)                    dblH = objShpBaseCfg.mdblHeight;
190)                }
191)                dblW *= dblDefaultSize;
192)                dblH *= dblDefaultSize;
193)                for (int i = 0; i < this.NumberOfVolumes; i++)
194)                {
195)                    dblX = ((dblW - (dblW / 3)) * i) + flDefaultX;
196)                    dblY = flDefaultY;
197)                    shpVol = CModel.TargetExcelApp.ActiveSheet.Shapes.AddShape(
198)                                objShpBaseCfg.mmsoAutoShapeType
199)                                , dblX, dblY, dblW, dblH);
200)
201)                    shpVol.IncrementRotation(objShpBaseCfg.mlShapeRotation);
202)                    if (objShpBaseCfg.mblnDash)
203)                    {
204)                        shpVol.Line.DashStyle = Office.MsoLineDashStyle.msoLineSysDash
    ;
205)                        shpVol.Line.Weight = 3;
206)                    }
207)                    FormatBaseShape(shpVol, this.ComponentNumber + this.Name + CConsta
    nts.SZ_HYDRO_VOLUME_PREFIX + (i + 1).ToString(), (i + 1).ToString(CConstants.SZ_MS
    K_VOL_NO));
208)                    shpVol.AlternativeText = CConstants.SZ_HYDRO_COMP + CConstants.SZ_
    HYDRO_TYPE_SEP + objShpBaseCfg.Name + CConstants.SZ_HYDRO_TYPE_SEP + CConstants.SZ
    _HYDRO_VOLUME_PREFIX + (i + 1).ToString();
209)
210)                    aszVolsNames[i] = shpVol.Name;
211)                }
212)                dblW += (dblW * 2 / 3) * (this.NumberOfVolumes - 1);
213)                shpCompLabel = CModel.TargetExcelApp.ActiveSheet.Shapes.AddLabel(
214)                                Office.MsoTextOrientation.msoTextOrientationHorizont
    al
215)                                , flDefaultX, flDefaultY + dblH, dblW, CConstants.DB
    L_DEFAULT_LABEL_HEIGHT);
216)                FormatBaseShape(shpCompLabel, this.Name, this.Name);
217)                aszVolsNames[this.NumberOfVolumes] = shpCompLabel.Name;
218)                shpComp = CModel.TargetExcelApp.ActiveSheet.Shapes.Range(aszVolsNames)
    .Group;
219)                shpComp.Name = this.ComponentNumber + this.Name;
220)                shpComp.AlternativeText = CConstants.SZ_HYDRO_COMP + CConstants.SZ_HYD
    RO_TYPE_SEP + objShpBaseCfg.Name;
221)                if (objShpBaseCfg.mszType == CConstants.SZ_HEAT)
222)                    FormatHeatShape(shpComp);
223)                if (blnGetPositionFromPrevious)
224)                {
225)                    shpComp.Width = flPrevW;
226)                    shpComp.Height = flPrevH;
227)                    shpComp.Rotation = flPrevRotation;
228)                }
229)                shpComp.Locked = false;
230)
231)                //shpComp.ControlFormat.LockedText = false;
232)
233)                //This part is commonly used on Junction Hydro Components, such as sin
    gle junctions, valves, etc.
234)                if (this.FromComponents != null && this.ToComponents != null && this.F
    romComponents.Length > 0 && this.ToComponents.Length > 0)
235)                {
```

```
236)                 Excel.Shape shpTgt = CModel.TargetExcelApp.ActiveSheet.Shapes(shpC
    omp.Name + CConstants.SZ_HYDRO_VOLUME_PREFIX + "1");
237)
238)                 for(int i=0;i<this.FromComponents.Length;i++)
239)                 {
240)                     Excel.Shape shpFrom = CModel.TargetExcelApp.ActiveSheet.Shapes
    (this.FromComponents[i].ShapeVolumeName);
241)                     Excel.Shape shpArrowFrom = CModel.TargetExcelApp.ActiveSheet.S
    hapes.AddConnector(Office.MsoConnectorType.msoConnectorElbow, 10, 10, 100, 100);
242)                     FormatArrow(shpArrowFrom);
243)                     shpArrowFrom.Name = shpComp.Name + "_FROM" + i.ToString();
244)                     ConnectShapesFrom(shpTgt, shpFrom, this.FromComponent.ShapeVol
    umeFace, shpArrowFrom);
245)                 }
246)
247)                 for (int i = 0; i < this.FromComponents.Length; i++)
248)                 {
249)                     Excel.Shape shpTo = CModel.TargetExcelApp.ActiveSheet.Shapes(t
    his.ToComponents[i].ShapeVolumeName);
250)                     Excel.Shape shpArrowTo = CModel.TargetExcelApp.ActiveSheet.Sha
    pes.AddConnector(Office.MsoConnectorType.msoConnectorElbow, 10, 10, 100, 100);
251)                     FormatArrow(shpArrowTo);
252)                     shpArrowTo.Name = shpComp.Name + "_TO" + i.ToString();
253)                     ConnectShapesTo(shpTgt, shpTo, this.ToComponent.ShapeVolumeFac
    e, shpArrowTo);
254)                 }
255)             }
256)
257)             //This part is commonly used on Heat Components
258)             Excel.Shape shpLeftBound = null;
259)             Excel.Shape shpLeftBoundLabel = null;
260)             float flRightShiftLeft = 0, flRotation = 0, flRadius = 0, flRadAngle =
     0;
261)             if(this.BoundaryHydroComponent != null && this.BoundaryHydroComponent.
    ComponentNumber != "")
262)             {
263)                 shpLeftBound      = CModel.TargetExcelApp.ActiveSheet.Shapes(this.
    BoundaryHydroComponent.ComponentNumber + this.BoundaryHydroComponent.Name);
264)                 shpLeftBoundLabel = CModel.TargetExcelApp.ActiveSheet.Shapes(this.
    BoundaryHydroComponent.Name);
265)                 flRotation = shpLeftBound.Rotation;
266)                 //Set rotation to zero was necessary in order to allow grouping be
    tween Hydro and Heat. In this case, none of the Trigonometric calculation would be
     necessary, though.
267)                 // However, if one day grouping is removed, the following code wil
    l still draw Heat component correctly attached to a Hydro Component.
268)                 shpLeftBound.Rotation = 0;
269)                 shpComp.Rotation = shpLeftBound.Rotation;
270)                 flRadAngle = (float)(shpComp.Rotation / 180 * Math.PI);
271)                 flRotationSign = (this.BoundaryHydroComponent.LeftBoundary) ? 1 :
    -1;
272)                 if(shpLeftBound.Height > shpLeftBound.Width)
273)                 {//Left means LEFT side of component
274)                     shpComp.Height = shpLeftBound.Height;
275)                     shpComp.Width = (float)CConstants.DBL_DEFAULT_HEAT_WIDTH;
276)                     flRadius = (shpComp.Width + shpLeftBound.Width ) / 2;
277)                     if (this.BoundaryHydroComponent.LeftBoundary)
278)                         flRightShiftLeft = -shpComp.Width;
279)                     else
280)                         flRightShiftLeft = shpLeftBound.Width;
281)                     shpComp.Top  = shpLeftBound.Top  + (flRotationSign * (-
    flRadius * (float)Math.Sin(flRadAngle)));
282)                     shpComp.Left = shpLeftBound.Left + flRightShiftLeft + (flRotat
    ionSign * (flRadius * (1 - (float)Math.Cos(flRadAngle))));
283)                 }
284)                 else
285)                 { //In this situation, Hydro component is larger than higher; so,
    in this case Left means ABOVE component
286)                     shpComp.Width = shpLeftBound.Width;
287)                     shpComp.Height = (float)CConstants.DBL_DEFAULT_HEAT_WIDTH;
288)                     flRadius = (shpComp.Height + shpLeftBound.Height) / 2;
289)                     if (this.BoundaryHydroComponent.LeftBoundary)
290)                         flRightShiftLeft = -shpComp.Height;
291)                     else
292)                         flRightShiftLeft = shpLeftBound.Height;
293)                     shpComp.Top  = shpLeftBound.Top  + flRightShiftLeft + (flRotat
    ionSign * (flRadius * (1 - (float)Math.Cos(flRadAngle))));
```

```
294)                        shpComp.Left = shpLeftBound.Left + (flRotationSign * (flRadius
       * (float)Math.Sin(flRadAngle)));
295)                    }
296)                    string szHeatGroupName = shpLeftBound.Name;
297)                    string[] aszHeatGroup = new string[] { shpComp.Name, szHeatGroupNa
       me };
298)                    Excel.Shape shpHeatComp = CModel.TargetExcelApp.ActiveSheet.Shapes
       .Range(aszHeatGroup).Group;
299)                    shpHeatComp.Rotation = flRotation;
300)                    shpHeatComp.Name = szHeatGroupName;
301)                }
302)
303)                CDiagram.ProtectDiagram(shtCanvas);
304)            }
305)            public static void FormatArrow(Excel.Shape v_shpArrow)
306)            {
307)                v_shpArrow.Line.EndArrowheadStyle = Office.MsoArrowheadStyle.msoArrowh
       eadTriangle;
308)                v_shpArrow.Line.EndArrowheadLength = Office.MsoArrowheadLength.msoArro
       wheadLong;
309)                v_shpArrow.Line.EndArrowheadWidth = Office.MsoArrowheadWidth.msoArrowh
       eadWide;
310)                v_shpArrow.Locked = false;
311)            }
312)            public static void ConnectShapesFrom(Excel.Shape v_shpTgt, Excel.Shape v_s
       hpFrom, int v_iFromConnectionNumber, Excel.Shape v_shpFromArrow)
313)            {
314)                v_shpFromArrow.ConnectorFormat.BeginConnect(v_shpFrom, v_iFromConnecti
       onNumber);
315)                v_shpFromArrow.ConnectorFormat.EndConnect(v_shpTgt, 1);
316)            }
317)            public static void ConnectShapesTo(Excel.Shape v_shpTgt, Excel.Shape v_shp
       To, int v_iToConnectionNumber, Excel.Shape v_shpToArrow)
318)            {
319)                v_shpToArrow.ConnectorFormat.BeginConnect(v_shpTgt, 3);
320)                v_shpToArrow.ConnectorFormat.EndConnect(v_shpTo, v_iToConnectionNumber
       );
321)            }
322)
323)            public static void FormatBaseShape(Excel.Shape v_shp, string v_szName, str
       ing v_szCaption)
324)            {
325)                v_shp.Name = v_szName;
326)                v_shp.TextFrame2.TextRange.Characters.Text = v_szCaption;
327)                v_shp.TextFrame2.TextRange.Font.Size = CConstants.L_DEFAULT_FONT_SIZE;
328)                v_shp.TextFrame2.MarginTop = 2;
329)                v_shp.TextFrame2.MarginBottom = 0;
330)                v_shp.TextFrame2.MarginLeft = 0;
331)                v_shp.TextFrame2.MarginRight = 0;
332)                v_shp.TextFrame.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;
333)                v_shp.TextFrame.VerticalAlignment = Excel.XlVAlign.xlVAlignTop;
334)                v_shp.Locked = false;
335)                v_shp.ControlFormat.LockedText = false;
336)            }
337)            public static void FormatHeatShape(Excel.Shape v_shp)
338)            {
339)                v_shp.Width = (float)(v_shp.Height / 10);
340)                foreach (Excel.Shape shpVolChild in v_shp.GroupItems)
341)                {
342)                    if (shpVolChild.Name.IndexOf(v_shp.Name + CConstants.SZ_HYDRO_VOLU
       ME_PREFIX) >= 0)
343)                    {
344)                        shpVolChild.Fill.Visible = Office.MsoTriState.msoTrue;
345)                        shpVolChild.Fill.ForeColor.RGB = System.Drawing.ColorTranslato
       r.ToOle(System.Drawing.Color.FromArgb(255, 0, 0));
346)                        shpVolChild.Fill.Transparency = 0;
347)                        shpVolChild.Fill.Solid();
348)                        shpVolChild.Fill.Patterned(Office.MsoPatternType.msoPatternDar
       kUpwardDiagonal);
349)
350)                        shpVolChild.Line.Visible = Office.MsoTriState.msoTrue;
351)                        shpVolChild.Line.ForeColor.RGB = System.Drawing.ColorTranslato
       r.ToOle(System.Drawing.Color.FromArgb(192, 0, 0));
352)                        shpVolChild.Line.Transparency = 0;
353)
354)                    }
355)                }
```

```
356)           }
357)       }
358)
359)       public class CCompProperties
360)       {
361)           public virtual CBaseShape ShapeType { get { return null; } }
362)           public virtual string CardPrefix {  get { return ""; } }
363)
364)           public virtual CCompProperty CreateCompProperty(string v_szCard, int v_iCa
       rdStart, int v_iCardEnd, bool v_blnMandatory, string v_szDefaultValue, string v_sz
       Description, EnumHydroPropertyType v_enuType, CCompProperty[] v_aobjWords, object
       v_objBindSource, string v_szValueMember, string v_szDisplayMember)
365)           {
366)               CCompProperty objPropRet = new CCompProperty(this.ShapeType, v_szCard,
        v_iCardStart, v_iCardEnd, v_blnMandatory, v_szDefaultValue, v_szDescription, v_en
       uType, v_aobjWords, v_objBindSource, v_szValueMember, v_szDisplayMember);
367)               objPropRet.CardPrefix = this.CardPrefix;
368)               return objPropRet;
369)           }
370)           public virtual CCompProperty CreateCompProperty(string v_szCard, bool v_bl
       nMandatory, string v_szDefaultValue, string v_szDescription, EnumHydroPropertyType
        v_enuType, object v_objBindSource = null, string v_szValueMember = CConstants.SZ_
       BINDING_DEFAULT_VALUEMEMBER, string v_szDisplayMember = CConstants.SZ_BINDING_DEFA
       ULT_DISPLAYEMEMBER)
371)           {
372)               return this.CreateCompProperty(v_szCard, -1, -
       1, v_blnMandatory, v_szDefaultValue, v_szDescription, v_enuType, null, v_objBindSo
       urce, v_szValueMember, v_szDisplayMember);
373)           }
374)           public virtual CCompProperty CreateCompProperty(string v_szCard, bool v_bl
       nMandatory, string v_szDefaultValue, string v_szDescription, CCompProperty[] v_aob
       jWords)
375)           {
376)               return this.CreateCompProperty(v_szCard, -1, -
       1, v_blnMandatory, v_szDefaultValue, v_szDescription, EnumHydroPropertyType.MultiW
       ord, v_aobjWords, null, "", "");
377)           }
378)           public virtual CCompProperty CreateCompProperty(string v_szCard, int v_iCa
       rdStart, int v_iCardEnd, bool v_blnMandatory, string v_szDefaultValue, string v_sz
       Description, CCompProperty[] v_aobjWords)
379)           {
380)               return this.CreateCompProperty(v_szCard, v_iCardStart, v_iCardEnd, v_b
       lnMandatory, v_szDefaultValue, v_szDescription, EnumHydroPropertyType.MultiWord, v
       _aobjWords, null, "", "");
381)           }
382)
383)           public virtual CCompProperty[] Properties
384)           {
385)               get
386)               {
387)                   CCompProperty[] aobjRet = CHydroProperties.PackProperties(  this.R
       equiredProperties
388)                                                                                   , this.O
       ptionalProperties);
389)                   return aobjRet;
390)               }
391)           }
392)           public virtual CCompProperty[] RequiredProperties
393)           {
394)               get
395)               {
396)                   return null;
397)               }
398)           }
399)           public virtual CCompProperty[] OptionalProperties
400)           {
401)               get
402)               {
403)                   return null;
404)               }
405)           }
406)
407)           public static CCompProperty[] PackProperties(params CCompProperty[][] v_ao
       bjProperties)
408)           {
409)               int iQtParams = v_aobjProperties.Length;
410)               int iQtProps = 0;
```

```
411)                for (int i = 0; i < iQtParams; i++)
412)                {
413)                    if (v_aobjProperties[i] != null)
414)                        iQtProps += v_aobjProperties[i].Length;
415)                }
416)                CCompProperty[] aobjRet = new CCompProperty[iQtProps];
417)                int iCurrProp = 0;
418)                for (int i = 0; i < iQtParams; i++)
419)                {
420)                    if (v_aobjProperties[i] != null)
421)                    {
422)                        for (int j = 0; j < v_aobjProperties[i].Length; j++)
423)                        {
424)                            aobjRet[iCurrProp] = v_aobjProperties[i][j];
425)                            aobjRet[iCurrProp].aobjWords = v_aobjProperties[i][j].aobj
    Words;
426)                            iCurrProp++;
427)                        }
428)                    }
429)                }
430)                return aobjRet;
431)            }
432)
433)            public CCompProperty GetPropertyByCardName(string v_szCard)
434)            {
435)                CCompProperty objRet = null;
436)                CCompProperty[] aobjAll = this.Properties;
437)                for (int i = 0; i < aobjAll.Length; i++)
438)                {
439)                    if (v_szCard == aobjAll[i].szCardName)
440)                    {
441)                        objRet = aobjAll[i];
442)                        break;
443)                    }
444)                    if (objRet == null)
445)                    {
446)                        if (aobjAll[i].aobjWords != null)
447)                        {
448)                            for (int j = 0; j < aobjAll[i].aobjWords.Length; j++)
449)                            {
450)                                if (v_szCard == aobjAll[i].aobjWords[j].szCardName)
451)                                {
452)                                    objRet = aobjAll[i].aobjWords[j];
453)                                    break;
454)                                }
455)                            }
456)                            if (objRet != null)
457)                                break;
458)                        }
459)                    }
460)                }
461)                return objRet;
462)            }
463)
464)            public bool IsValidWord(System.Windows.Forms.ComboBox cmbWord, out string
    v_szErrMsg)
465)            {
466)                string szWord = "";
467)                if(cmbWord.SelectedValue != null)
468)                    szWord = cmbWord.SelectedValue.ToString();
469)                CCompProperty objProp = (CCompProperty) cmbWord.Tag;
470)                bool blnRet = IsValidWord(ref szWord, objProp, out v_szErrMsg);
471)                return blnRet;
472)            }
473)            public bool IsValidWord(System.Windows.Forms.Control ctlWord, out string v
    _szErrMsg)
474)            {
475)                string szWord = ctlWord.Text;
476)                CCompProperty objProp = (CCompProperty) ctlWord.Tag;
477)                bool blnRet = IsValidWord(ref szWord, objProp, out v_szErrMsg);
478)                ctlWord.Text = szWord;
479)                return blnRet;
480)            }
481)            public bool IsValidWord(string v_szWord, string v_szCardname, out string v
    _szErrMsg)
482)            {
483)                return IsValidWord(ref v_szWord, v_szCardname, out v_szErrMsg);
```

```
484)          }
485)          public bool IsValidWord(ref string v_szWord, string v_szCardname, out stri
      ng v_szErrMsg)
486)          {
487)              bool blnRet = false;
488)              CCompProperty objProp = this.GetPropertyByCardName(v_szCardname);
489)              blnRet = this.IsValidWord(ref v_szWord, objProp, out v_szErrMsg);
490)              return blnRet;
491)          }
492)          public bool IsValidWord(string v_szWord, CCompProperty v_objHydroProperty,
       out string v_szErrMsg)
493)          {
494)              return IsValidWord(ref v_szWord, v_objHydroProperty, out v_szErrMsg);
495)          }
496)          public bool IsValidWord(ref string v_szWord, CCompProperty v_objHydroPrope
      rty, out string v_szErrMsg)
497)          {
498)              return CCompProperties.VerifyIfWordIsValid(ref v_szWord, v_objHydroPro
      perty, out v_szErrMsg);
499)          }
500)          public static bool VerifyIfWordIsValid(ref string v_szWord, CCompProperty
      v_objHydroProperty, out string v_szErrMsg)
501)          {
502)              bool blnRet = false;
503)              string szFormula = "";
504)              v_szErrMsg = "";
505)
506)              //Check if current word was typed as Formula
507)              if(v_szWord.Length > 1 && v_szWord.Left(1) == "=")
508)              {
509)                  string szEval = "";
510)                  int iEval = 0;
511)                  szFormula = v_szWord;
512)                  szEval = CModel.GetShtInp().TargetSheet.Evaluate(szFormula).ToStri
      ng();
513)                  if (int.TryParse(szEval, out iEval) && iEval < CConstants.I_SHT_EV
      ALERR)
514)                  {
515)                      v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_FORMULA.Trans
      late();
516)                      blnRet = false;
517)                  }
518)                  else
519)                  {
520)                      v_szWord = szEval;
521)                  }
522)              }
523)              if (v_szErrMsg == "") //Keep running evaluation only if Formula was su
      cessfully evaluated, or if passed word is not a formula at all
524)              {
525)                  if (v_objHydroProperty == null || (v_objHydroProperty.OptionalProp
      erty && v_szWord == ""))
526)                      blnRet = true;
527)                  else
528)                  {
529)                      switch (v_objHydroProperty.enuType)
530)                      {
531)                          case EnumHydroPropertyType.WordCompNumber:
532)                              v_szErrMsg = CGlobal.Translation.TXT_ERR_COMP_NUMBER.T
      ranslate();
533)                              blnRet = CValidation.IsWordCompNumber(ref v_szWord);
534)                              break;
535)                          case EnumHydroPropertyType.WordCompGeomNo:
536)                              v_szErrMsg = CGlobal.Translation.TXT_ERR_COMPGEOM_NUMB
      ER.Translate();
537)                              blnRet = CValidation.IsWordCompNumber(ref v_szWord, 99
      99);
538)                              break;
539)                          case EnumHydroPropertyType.WordCompName:
540)                              v_szErrMsg = CGlobal.Translation.TXT_ERR_COMP_NAME.Tra
      nslate();
541)                              blnRet = (v_szWord != "");
542)                              break;
543)                          case EnumHydroPropertyType.WordCompType:
544)                              break;
545)                          case EnumHydroPropertyType.WordIntVolOrTbl:
```

```
546)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_TABLE
      .Translate();
547)                                blnRet = CValidation.IsWordVolOrTbl(v_szWord);
548)                                break;
549)                            case EnumHydroPropertyType.WordIntComposition:
550)                            case EnumHydroPropertyType.WordInteger:
551)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_INTEG
      ER.Translate();
552)                                blnRet = CValidation.IsWordInteger(v_szWord);
553)                                break;
554)                            case EnumHydroPropertyType.WordIntHeat:
555)                            case EnumHydroPropertyType.WordInteger0orHigher:
556)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_INTEG
      ER0ORHIGHER.Translate();
557)                                blnRet = CValidation.IsWordInteger(v_szWord, 0);
558)                                break;
559)                            case EnumHydroPropertyType.WordInteger1to999:
560)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_INTEG
      ER1TO999.Translate();
561)                                blnRet = CValidation.IsWordInteger(v_szWord, 1, 999);
562)                                break;
563)                            case EnumHydroPropertyType.WordIntegerNotUsedZero:
564)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_INTNO
      TUSEDZERO.Translate();
565)                                blnRet = CValidation.IsWordInteger(v_szWord, 0, 0);
566)                                break;
567)                            case EnumHydroPropertyType.WordInteger1to99:
568)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_INTEG
      ER1TO99.Translate();
569)                                blnRet = CValidation.IsWordInteger(v_szWord, 1, 99);
570)                                break;
571)                            case EnumHydroPropertyType.WordInteger1to9:
572)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_INTEG
      ER1TO99.Translate();
573)                                blnRet = CValidation.IsWordInteger(v_szWord, 1, 9);
574)                                break;
575)                            case EnumHydroPropertyType.WordInteger0or1:
576)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_INTEG
      ER0OR1.Translate();
577)                                blnRet = CValidation.IsWordInteger(v_szWord, 0, 1);
578)                                break;
579)                            case EnumHydroPropertyType.WordInteger101:
580)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_INTEG
      ER101.Translate();
581)                                blnRet = CValidation.IsWordInteger(v_szWord, -1, 1);
582)                                break;
583)                            case EnumHydroPropertyType.WordInteger012:
584)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_INTEG
      ER012.Translate();
585)                                blnRet = CValidation.IsWordInteger(v_szWord, 0, 2);
586)                                break;
587)                            case EnumHydroPropertyType.WordInteger123:
588)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_INTEG
      ER123.Translate();
589)                                blnRet = CValidation.IsWordInteger(v_szWord, 1, 3);
590)                                break;
591)                            case EnumHydroPropertyType.WordInteger12:
592)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_INTEG
      ER12.Translate();
593)                                blnRet = CValidation.IsWordInteger(v_szWord, 1, 2);
594)                                break;
595)                            case EnumHydroPropertyType.WordIntComponent:
596)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_INTCO
      MP.Translate();
597)                                blnRet = CValidation.IsWordIntComponent(v_szWord);
598)                                break;
599)                            case EnumHydroPropertyType.WordIntTrip:
600)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_INTTR
      IP.Translate();
601)                                blnRet = CValidation.IsWordIntTrip(v_szWord);
602)                                break;
603)                            case EnumHydroPropertyType.WordIntCurveRegime:
604)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_INTCU
      RVEREGIME.Translate();
605)                                blnRet = CValidation.IsWordInteger(v_szWord, 1, 8);
606)                                break;
607)                            case EnumHydroPropertyType.WordIntBoundCondition:
```

```
608)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_INTBO
    UNDCOND.Translate();
609)                                blnRet = CValidation.IsWordIntBoundCondition(v_szWord)
    ;
610)                                break;
611)                            case EnumHydroPropertyType.WordIntMN:
612)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_INTMN
    .Translate();
613)                                blnRet = CValidation.IsWordIntMN(v_szWord);
614)                                break;
615)                            case EnumHydroPropertyType.WordInt210Pump:
616)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_INT21
    0PUMP.Translate();
617)                                blnRet = CValidation.IsWordIntPump(v_szWord, -2, -
    1, 0);
618)                                break;
619)                            case EnumHydroPropertyType.WordInt10Pump:
620)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_INT10
    PUMP.Translate();
621)                                blnRet = CValidation.IsWordIntPump(v_szWord, -1, 0);
622)                                break;
623)                            case EnumHydroPropertyType.WordInt310Pump:
624)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_INT31
    0PUMP.Translate();
625)                                blnRet = CValidation.IsWordIntPump(v_szWord, -3, -
    1, 0);
626)                                break;
627)                            case EnumHydroPropertyType.WordAlphaDkheat:
628)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_ALPHA
    DKHEAT.Translate();
629)                                blnRet = CValidation.IsWordAlphaDkheat(v_szWord);
630)                                break;
631)                            case EnumHydroPropertyType.WordReal:
632)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_REAL.
    Translate();
633)                                blnRet = CValidation.IsWordReal(v_szWord);
634)                                break;
635)                            case EnumHydroPropertyType.WordAlphanumeric:
636)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_ALPHA
    .Translate();
637)                                blnRet = CValidation.IsWordAlphanumeric(v_szWord);
638)                                break;
639)                            case EnumHydroPropertyType.WordEBT:
640)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_EBT.T
    ranslate();
641)                                blnRet = CValidation.IsWordEBT(v_szWord);
642)                                break;
643)                            case EnumHydroPropertyType.WordTLPVBFE:
644)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_TLPVB
    FE.Translate();
645)                                blnRet = CValidation.IsWordTLPVBFE(v_szWord);
646)                                break;
647)                            case EnumHydroPropertyType.Word00000F0:
648)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_00000
    F0.Translate();
649)                                blnRet = CValidation.IsWord00000F0(v_szWord);
650)                                break;
651)                            case EnumHydroPropertyType.Word0EF0CAHS:
652)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_0EF0C
    AHS.Translate();
653)                                blnRet = CValidation.IsWord0EF0CAHS(v_szWord);
654)                                break;
655)                            case EnumHydroPropertyType.WordReal0to2:
656)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_REAL0
    TO2.Translate();
657)                                blnRet = CValidation.IsWordReal(v_szWord, 0, 2);
658)                                break;
659)                            case EnumHydroPropertyType.WordRealZero:
660)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_REALZ
    ERO.Translate();
661)                                blnRet = CValidation.IsWordReal(v_szWord, 0, 0);
662)                                break;
663)                            case EnumHydroPropertyType.WordRealNonZero:
664)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_REALN
    ONZERO.Translate();
665)                                blnRet = CValidation.IsWordRealNonZero(v_szWord);
666)                                break;
```

```
667)                            case EnumHydroPropertyType.WordRealAngle360:
668)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_ANGLE
    360.Translate();
669)                                blnRet = CValidation.IsWordReal(v_szWord, 0, 360);
670)                                break;
671)                            case EnumHydroPropertyType.WordRealAngle90:
672)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_ANGLE
    90.Translate();
673)                                blnRet = CValidation.IsWordReal(v_szWord, 0, 90);
674)                                break;
675)                            case EnumHydroPropertyType.WordRealHigherThanZero:
676)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_REALH
    IGHERZERO.Translate();
677)                                blnRet = CValidation.IsWordReal(v_szWord, 0, v_blnLoBo
    undInclusive: false);
678)                                break;
679)                            case EnumHydroPropertyType.WordReal1to1:
680)                                blnRet = CValidation.IsWordReal(v_szWord, -1, 1);
681)                                break;
682)                            case EnumHydroPropertyType.WordReal0to1:
683)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_REAL0
    TO1.Translate();
684)                                blnRet = CValidation.IsWordReal(v_szWord, 0, 1);
685)                                break;
686)                            case EnumHydroPropertyType.WordReal0orHigher:
687)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_REAL0
    ORHIGH.Translate();
688)                                blnRet = CValidation.IsWordReal(v_szWord, 0);
689)                                break;
690)                            case EnumHydroPropertyType.WordReal0orLower:
691)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_REAL0
    ORLOWER.Translate();
692)                                blnRet = CValidation.IsWordReal(v_szWord, v_dblHiBound
    : 0);
693)                                break;
694)                            case EnumHydroPropertyType.WordRealAngle9090:
695)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_ANGLE
    9090.Translate();
696)                                blnRet = CValidation.IsWordReal(v_szWord, -90, 90);
697)                                break;
698)                            case EnumHydroPropertyType.WordRealNotImplementedZero:
699)                                v_szErrMsg = CGlobal.Translation.TXT_ERR_INVALID_REALN
    OTIMPLEMENTED.Translate();
700)                                blnRet = CValidation.IsWordReal(v_szWord, 0, 0);
701)                                break;
702)                        }
703)                    }
704)                }
705)                //If word was typed as formula, it cannot be changed by validation fun
    ctions, so restore it to original entered formula
706)                if(szFormula != "")
707)                {
708)                    v_szWord = szFormula;
709)                }
710)
711)                return blnRet;
712)            }
713)        }
714)
715)        public class CCompProperty
716)        {
717)            public CBaseShape objParentType;
718)            public string szCardName;
719)            public string szDescription;
720)            public EnumHydroPropertyType enuType;
721)            public CCompProperty[] aobjWords;
722)            public bool MandatoryProperty;
723)            public string szDefaultValue;
724)            public bool OptionalProperty { get { return !MandatoryProperty; } }
725)            public int iCardStart;
726)            public int iCardEnd;
727)            public string CardPrefix = "";
728)            public BindingSource objBindSource = null;
729)            public string szBindValue = "";
730)            public string szBindDisplay = "";
731)
```

```
732)          public CCompProperty(CBaseShape v_objParentType, string v_szCard, int v_iC
      ardStart, int v_iCardEnd, bool v_blnMandatory, string v_szDefaultValue, string v_s
      zDescription, EnumHydroPropertyType v_enuType, CCompProperty[] v_aobjWords, object
       v_objBindSource, string v_szValueMember, string v_szDisplayMember)
733)          {
734)              this.objParentType = v_objParentType;
735)              this.szCardName = v_szCard;
736)              if(v_iCardStart < 0) //Determine Default Value
737)              {
738)                  if (szCardName.Right(CConstants.SZ_INP_MULTICARD.Length) == CConst
      ants.SZ_INP_MULTICARD) //This Card Allows Multiple Instances, eg. 1CCCG801-899
739)                      v_iCardStart = 1;
740)              }
741)              this.iCardStart = v_iCardStart;
742)              if(v_iCardEnd < 0) //Determine Default Value
743)              {
744)                  if (szCardName.Right(CConstants.SZ_INP_MULTICARD.Length) == CConst
      ants.SZ_INP_MULTICARD) //This Card Allows Multiple Instances, eg. 1CCCG801-899
745)                      v_iCardEnd = 99;
746)              }
747)              this.iCardEnd = v_iCardEnd;
748)              this.szDescription = v_szDescription;
749)              this.enuType = v_enuType;
750)              this.aobjWords = v_aobjWords;
751)              if(this.aobjWords != null && this.szCardName.Right(1) == " ")
752)              {//If CardName end with whitespace, this is probably a CARD, and not a
       subword
753)                  foreach(CCompProperty objWord in this.aobjWords)
754)                  {//In this case, when I'm adding words to a card, if WORD does not
       specify a cardnumber (only a word order), include parent card name to child WORD
755)                      if (objWord.szCardName.Left(1) == "W")
756)                          objWord.szCardName = this.szCardName + objWord.szCardName;
757)                  }
758)              }
759)              this.MandatoryProperty = v_blnMandatory;
760)              this.szDefaultValue = v_szDefaultValue;
761)              //Binding info
762)              if(v_objBindSource != null)
763)              {
764)                  this.objBindSource = new BindingSource(v_objBindSource, null);
765)                  this.szBindValue = v_szValueMember;
766)                  this.szBindDisplay = v_szDisplayMember;
767)              }
768)          }
769)          public CCompProperty(CBaseShape v_objParentType,  string v_szCard, bool v_
      blnMandatory, string v_szDefaultValue, string v_szDescription, EnumHydroPropertyTy
      pe v_enuType, object v_objBindSource = null, string v_szValueMember = CConstants.S
      Z_BINDING_DEFAULT_VALUEMEMBER, string v_szDisplayMember = CConstants.SZ_BINDING_DE
      FAULT_DISPLAYEMEMBER) : this(v_objParentType, v_szCard, -1, -
      1, v_blnMandatory, v_szDefaultValue, v_szDescription, v_enuType, null, v_objBindSo
      urce, v_szValueMember, v_szDisplayMember)
770)          { }
771)          public CCompProperty(CBaseShape v_objParentType,  string v_szCard, bool v_
      blnMandatory, string v_szDefaultValue, string v_szDescription, CCompProperty[] v_a
      objWords) : this(v_objParentType, v_szCard, -1, -
      1, v_blnMandatory, v_szDefaultValue, v_szDescription, EnumHydroPropertyType.MultiW
      ord, v_aobjWords, null, "", "")
772)          { }
773)          public CCompProperty(CBaseShape v_objParentType,  string v_szCard, int v_i
      CardStart, int v_iCardEnd, bool v_blnMandatory, string v_szDefaultValue, string v_
      szDescription, CCompProperty[] v_aobjWords) : this(v_objParentType, v_szCard, v_iC
      ardStart, v_iCardEnd, v_blnMandatory, v_szDefaultValue, v_szDescription, EnumHydro
      PropertyType.MultiWord, v_aobjWords, null, "", "")
774)          { }
775)
776)          public string LabelOnly
777)          {   get { return this.szDescription + (this.MandatoryProperty?"*":"") + "
      (" + this.szCardName.Trim() + ")"; } }
778)          public string Label
779)          {   get { return this.LabelOnly + CConstants.SZ_LBL_SUFIX; } }
780)      }
781)
782)      public enum EnumHydroPropertyType
783)      {
784)          MultiWord //This card has more than 1 word as parameter
785)          , WordCompNumber
786)          , WordCompGeomNo
```

```
787)           , WordCompName
788)           , WordCompType
789)           , WordInteger
790)           , WordIntegerNotUsedZero
791)           , WordInteger0orHigher
792)           , WordInteger1to999
793)           , WordInteger1to99
794)           , WordInteger1to9
795)           , WordInteger0or1
796)           , WordInteger101
797)           , WordInteger012
798)           , WordInteger123
799)           , WordInteger12
800)           , WordIntComponent
801)           , WordIntTrip
802)           , WordIntHeat
803)           , WordIntComposition
804)           , WordIntVolOrTbl
805)           , WordIntBoundCondition
806)           , WordIntMN
807)           , WordInt210Pump
808)           , WordInt10Pump
809)           , WordInt310Pump
810)           , WordIntCurveRegime
811)           , WordAlphanumeric
812)           , WordAlphaVariable
813)           , WordAlphaMaterial
814)           , WordAlphaDkheat
815)           , WordEBT //Full EBT Parameter, with flags and Words 2 to 7
816)           , WordEBTflag //Only EBT Parameter
817)           , WordEBTargs //Only EBT arguments (Words 2 to 7)
818)           , WordTLPVBFE
819)           , Word00000F0
820)           , WordZeroTLPVBFE //TLPVBFE retricted to 0000000
821)           , Word000001E
822)           , Word00000FE
823)           , Word00P00FE
824)           , Word000000E
825)           , Word00110F0
826)           , WordJEFVCAHS
827)           , Word0E000000
828)           , Word0EFVCAHS
829)           , Word0EF0CAHS
830)           , Word00F0CAH0
831)           , Word0000CAHS
832)           , Word0000CAH0
833)           , WordReal
834)           , WordRealZero
835)           , WordRealNonZero
836)           , WordReal0orHigher
837)           , WordReal0orLower
838)           , WordReal0to1
839)           , WordReal0to2
840)           , WordReal1to1
841)           , WordRealAngle360
842)           , WordRealAngle90
843)           , WordRealAngle180
844)           , WordRealAngle9090 //May accept from -90 to 90 (inclusive).
845)           , WordRealHigherThanZero
846)           , WordRealNotImplementedZero //Some functionalities were not implemented y
       et on RELAP5. In these cases, user guide recommends to input as zero.
847)       }
848)   }
```

APÊNDICE F - Programação da superclasse de telas FrmH

```
1)   using System;
2)   using System.Collections.Generic;
3)   using System.ComponentModel;
4)   using System.Data;
5)   using System.Drawing;
6)   using System.Linq;
7)   using System.Text;
8)   using System.Threading.Tasks;
9)   using System.Windows.Forms;
10)  using System.Windows.Input;
11)
12)  namespace FastLapAddin
13)  {
14)      public partial class FrmH : Form
15)      {
16)          protected CComponent RelapComponent;
17)          public string PdfNamedReference = "";
18)          protected string p_szInvalidFields = ""; //To avoid validating all fields
     again when user clicks ADD button, this variable will be filled with the name of f
     ields with error during usage. If all fields are valid, this variable should be em
     pty
19)          public string ComponentNumberToLoad = "";
20)          public Image PictureIcon;
21)
22)          //To visually control Drag'n drop
23)          protected bool m_blnDragCapturing;
24)          protected long m_lDragStart = 0;
25)          protected long m_lDragStartX = 0;
26)          protected int m_iDragLastSpltrPos = 0;
27)
28)          public virtual int NumberOfVolumes
29)          {
30)              get
31)              {
32)                  return 1;
33)              }
34)          }
35)          public virtual Control[] ControlsWithCalc
36)          {
37)              get
38)              {
39)                  return new Control[] { };
40)              }
41)          }
42)
43)          public CHydro HydroComponent
44)          {
45)              get
46)              {
47)                  CHydro objRet = null;
48)                  try
49)                  {
50)                      objRet = (CHydro)this.RelapComponent;
51)                  }
52)                  catch { }
53)                  return objRet;
54)              }
55)              set
56)              {
57)                  RelapComponent = (CComponent)value;
58)              }
59)          }
60)
61)          public FrmH()
62)          {
63)              this.Icon = global::FastLapAddin.Properties.Resources.icoFastLap;
64)              InitializeComponent();
65)              this.SetStyle(ControlStyles.OptimizedDoubleBuffer, true);
66)              this.DoubleBuffered = true;
67)              this.StartPosition = FormStartPosition.CenterScreen;
68)          }
69)
```

```
70)         protected override CreateParams CreateParams //Avoid Form Flickering when
    dragging controls on the screen, enabling DoubleBuffering to ALL child controls at
     once
71)         {
72)             get
73)             {
74)                 CreateParams handleParam = base.CreateParams;
75)                 if (!this.DesignMode)
76)                 {
77)                     handleParam.ExStyle |= 0x02000000;   // WS_EX_COMPOSITED

78)                 }
79)                 return handleParam;
80)             }
81)         }
82)
83)         protected void Base_Load(object sender, EventArgs e)
84)         {
85)             if (!this.DesignMode)
86)             {
87)                 if (this.PdfNamedReference == "")
88)                     this.PdfNamedReference = CGlobal.Translation.SZ_PDF_HOOK_HYDRO
    ;
89)                 if (this.PictureIcon != null)
90)                     this.picIco.Image = this.PictureIcon;
91)
92)                 //Associate tags with controls
93)                 this.BaseAssociateTags();
94)                 this.AssociateTags();
95)
96)                 //Attach Calc functionality on textboxes
97)                 this.AttachCalcEventHandlers(this.ControlsWithCalc);
98)
99)                 if (this.HydroComponent != null)
100)                {
101)                    this.Text = CTranslation.SZ_APP_TITLE + CConstants.SZ_TXT_DESC
    R_SEP_SPC + RelapComponent.BaseShape.Name + " " + CGlobal.Translation.TXT_FRM_HYDR
    O_LBL_TITLE_SFX.Translate();
102)                    this.lblTitle.Text = RelapComponent.BaseShape.mszType.ToUpper(
    ) + CConstants.SZ_TXT_DESCR_SEP_SPC + RelapComponent.BaseShape.Name + " " + CGloba
    l.Translation.TXT_FRM_HYDRO_LBL_TITLE_SFX.Translate();
103)                }
104)
105)                this.BaseTranslateThis();
106)                this.TranslateThis();
107)
108)                //If form was instantiated with a Component Number to be loaded, n
    ow it is time to do it, once form controls are properly initiated
109)                if (this.ComponentNumberToLoad != "")
110)                    this.LoadComponent();
111)
112)                this.ValidateChildren();
113)            }
114)        }
115)        public CHydroProperties HydroProperties
116)        {
117)            get
118)            {
119)                return (CHydroProperties)this.HydroComponent.HydroProperties;
120)            }
121)        }
122)        protected void BaseAssociateTags()
123)        {
124)            //Associate tags with controls
125)            if (this.HydroComponent != null)
126)            {
127)                mskCompNo.Tag = this.HydroProperties.hydpHydroNumber;
128)                mskCompName.Tag = this.HydroProperties.hydpHydroName_0000W1;
129)            }
130)        }
131)        protected virtual void AssociateTags()
132)        {//This method was implemented in only to be overriden by child classes, i
    n order to better organize Cards x Controls associations
133)        }
134)        protected void BaseTranslateThis()
135)        {
```

```
136)                lblCompNo.Text = CGlobal.Translation.TXT_HYDRO_PROP_COMPNO.Translate()
    ;
137)                lblName.Text   = CGlobal.Translation.TXT_HYDRO_PROP_CCC0000_W1.Transla
    te();
138)                lblDescr.Text  = CGlobal.Translation.TXT_FRM_HYDRO_LBL_DESCR.Translate
    ();
139)                btnAdd.Text    = CGlobal.Translation.TXT_FRM_HYDRO_LBL_ADD.Translate()
    ;
140)            }
141)            protected virtual void TranslateThis()
142)            {//This method was implemented in only to be overriden by child classes, i
    n order to better organize labels translations
143)            }
144)
145)            protected virtual void ShowCalcOnGetFocus(object sender, EventArgs e)
146)            {
147)                CUdf.ShowCalcButton(btnCalc, (Control)sender);
148)            }
149)            protected virtual void HideCalcOnLostFocus(object sender, EventArgs e)
150)            {
151)                if(this.ActiveControl != btnCalc)
152)                    this.btnCalc.Visible = false;
153)            }
154)            protected virtual void AttachCalcEventHandlers(Control ctlObj)
155)            {
156)                ctlObj.Enter += this.ShowCalcOnGetFocus;
157)                ctlObj.Leave += this.HideCalcOnLostFocus;
158)            }
159)            protected virtual void AttachCalcEventHandlers(Control[] actlObj)
160)            {
161)                if(actlObj != null)
162)                {
163)                    foreach (Control ctl in actlObj)
164)                    {
165)                        this.AttachCalcEventHandlers(ctl);
166)                    }
167)                }
168)            }
169)
170)
171)            protected virtual void ValidateCmbControl(object sender, EventArgs e)
172)            {
173)                ComboBox cmbSender = (ComboBox)sender;
174)                IsCmbControlValid(cmbSender);
175)            }
176)            protected virtual void ValidateCmbControl(object sender, CancelEventArgs e
    )
177)            {
178)                ComboBox cmbSender = (ComboBox)sender;
179)                e.Cancel = !IsCmbControlValid(cmbSender);
180)            }
181)            protected virtual bool IsCmbControlValid(ComboBox v_cmbSender)
182)            {
183)                string szMsg = "";
184)                bool blnRet = false ;
185)                blnRet = true; //this.RelapComponent.PropertyCards.IsValidWord(v_cmbSe
    nder, out szMsg);
186)                //Visually Reflecting Form Validation on controls
187)                CUdf.ReflectValidationOnControl(blnRet, v_cmbSender, szMsg, tipMsg, re
    f p_szInvalidFields, this.btnAdd);
188)                return blnRet;
189)            }
190)            protected virtual void ValidateTxtControl(object sender, CancelEventArgs e
    )
191)            {
192)                if (!this.DesignMode)
193)                {
194)                    Control txtSender = (Control)sender;
195)                    string szMsg = "";
196)                    bool blnRet = false;
197)                    blnRet = this.RelapComponent.PropertyCards.IsValidWord(txtSender,
    out szMsg);
198)                    //Visually Reflecting Form Validation on controls
199)                    CUdf.ReflectValidationOnControl(blnRet, txtSender, szMsg, tipMsg,
    ref p_szInvalidFields, this.btnAdd);
200)                    e.Cancel = !blnRet;
201)                }
```

```
202)           }
203)
204)           protected virtual void btnCalc_Click(object sender, EventArgs e)
205)           {
206)               string szCurTag = "";
207)               CCompProperty objPropFromCurCell = null;
208)               string szVal = "";
209)               object objCur = btnCalc.Tag;
210)               if (objCur is Control)
211)               {
212)                   Control ctlCur = (Control) btnCalc.Tag;
213)                   objPropFromCurCell = ctlCur.Tag as CCompProperty;
214)                   if(objPropFromCurCell != null)
215)                       szVal = CUdf.ShowCalcForms(objPropFromCurCell, RelapComponent,
       this);
216)                   else
217)                   {
218)                       szCurTag = ctlCur.Tag.ToString();
219)                       szVal = CUdf.ShowCalcForms(szCurTag, RelapComponent, this);
220)                   }
221)                   if (szVal != "")
222)                   {
223)                       ctlCur.Text = szVal;
224)                       ctlCur.Focus();
225)                   }
226)               }
227)               else if (objCur is DataGridViewCell)
228)               {
229)                   DataGridViewCell celCur = (DataGridViewCell) btnCalc.Tag;
230)                   szCurTag = celCur.DataGridView.Columns[celCur.ColumnIndex].Tag.ToS
       tring();
231)                   szVal = CUdf.ShowCalcForms(szCurTag, RelapComponent, this);
232)                   if (szVal != "")
233)                       celCur.Value = szVal;
234)               }
235)           }
236)
237)           protected virtual void btnAdd_Click(object sender, EventArgs e)
238)           {
239)               if (p_szInvalidFields == "")
240)               {
241)                   CShtInp ocshtInp = CModel.GetShtInp();
242)                   //Erase old versions of component from Input
243)                   ocshtInp.RemoveComponentFromInp(this.mskCompNo.Text);
244)                   //Add recently configured component to Input, rewriting Card CCC00
       00 - Identification Component
245)                   ocshtInp.WriteCard(  CGlobal.Translation.TXT_FRM_HYDRO_LBL_TITLE_S
       FX.Translate() + " " + this.HydroComponent.ComponentNumber + CConstants.SZ_INP_DAS
       H + this.HydroComponent.Name + CConstants.SZ_TXT_DESCR_SEP + txtDescr.Text
246)                                       , this.HydroComponent.ComponentNumber
247)                                       , this.HydroComponent.Name
248)                                       , this.HydroProperties.hydpHydroIdentification
       _0000
249)                                       , this.HydroComponent.Name, this.HydroComponen
       t.BaseShape.mszType);
250)                   this.UpdateInp();
251)                   HydroComponent.Draw(this.mskCompNo.Text, this.mskCompName.Text, th
       is.txtDescr.Text, this.NumberOfVolumes);
252)                   this.Close();
253)               }
254)               else
255)               {
256)                   CUdf.MsgBox(CGlobal.Translation.TXT_ERR_HYDRO_INVALID_FORM.Transla
       te(), MessageBoxIcon.Exclamation);
257)               }
258)           }
259)           public virtual void UpdateInp()
260)           {//This method should be overrided by child forms, on order to put logic t
       o write RELAP5 input data
261)
262)           }
263)           public void LoadComponent()
264)           {
265)               string szCompNo = this.ComponentNumberToLoad;
266)               CHydro objComp = new CHydro(szCompNo);
267)
268)               this.mskCompNo.Text = szCompNo;
```

```
269)              this.mskCompName.Text = objComp.Name;
270)              this.txtDescr.Text = objComp.Description;
271)              this.ReadInp(szCompNo);
272)          }
273)          public virtual void ReadInp(string v_szCompNo)
274)          {//This method should be overrided by child forms, on order to put logic t
      o read RELAP5 input data and fill form controls
275)
276)          }
277)          public static FrmH InstantiateFromCompNo(string v_szCompNumber)
278)          {
279)              string szHydroType;
280)              FrmH frmHret;
281)              CHydro objHydro = new CHydro(v_szCompNumber);
282)              szHydroType = objHydro.BaseShape.mszType;
283)
284)              frmHret = FrmH.InstantiateFromType(szHydroType);
285)              frmHret.ComponentNumberToLoad = v_szCompNumber;
286)              return frmHret;
287)          }
288)          public static FrmH InstantiateFromType(string v_szHydroType)
289)          {
290)              FrmH frmHret;
291)
292)              switch (v_szHydroType)
293)              {
294)                  case CConstants.SZ_HYDRO_TMDPVOL:
295)                      frmHret = new FrmHTmdpvol();
296)                      break;
297)                  case CConstants.SZ_HYDRO_VALVE:
298)                      frmHret = new FrmHValve();
299)                      break;
300)                  case CConstants.SZ_HYDRO_SNGLVOL:
301)                      frmHret = new FrmHSnglvol();
302)                      break;
303)                  case CConstants.SZ_HYDRO_SNGLJUN:
304)                      frmHret = new FrmHSngljun();
305)                      break;
306)                  case CConstants.SZ_HYDRO_TMDPJUN:
307)                      frmHret = new FrmHTmdpjun();
308)                      break;
309)                  case CConstants.SZ_HYDRO_PIPE:
310)                  case CConstants.SZ_HYDRO_ANNULUS:
311)                  case CConstants.SZ_HYDRO_PRIZER:
312)                  case CConstants.SZ_HYDRO_CANCHAN:
313)                      frmHret = new FrmHPipe(v_szHydroType);
314)                      break;
315)                  case CConstants.SZ_HYDRO_JETMIXER:
316)                  case CConstants.SZ_HYDRO_ECCMIX:
317)                      frmHret = new FrmHMixer(v_szHydroType);
318)                      break;
319)                  case CConstants.SZ_HYDRO_BRANCH:
320)                      frmHret = new FrmHBranch();
321)                      break;
322)                  case CConstants.SZ_HYDRO_SEPARATR:
323)                  case CConstants.SZ_HYDRO_TURBINE:
324)                  case CConstants.SZ_HYDRO_PUMP:
325)                      frmHret = new FrmHPump();
326)                      break;
327)                  case CConstants.SZ_HYDRO_MTPLJUN:
328)                  case CConstants.SZ_HYDRO_ACCUM:
329)                      frmHret = new FrmH();
330)                      break;
331)                  default:
332)                      frmHret = new FrmH();
333)                      break;
334)              }
335)
336)              return frmHret;
337)          }
338)
339)          protected virtual void btnRef_Click(object sender, EventArgs e)
340)          {
341)              CUdf.OpenPdfManual(this.PdfNamedReference);
342)          }
343)
344)          private void btnPresDrop_Click(object sender, EventArgs e)
```

```
345)            {
346)                CUdf.RunExe(CGlobal.UserSettings.PathPresDropExe);
347)            }
348)
349)            protected virtual void Form_FormClosing(object sender, FormClosingEventArg
       s e)
350)            {
351)                e.Cancel = false;
352)            }
353)        }
354)    }
```

APÊNDICE G - Dados de entrada do código RELAP5 para a Experiência SUPER CANON

```
=Experimento CANON
*
100 new transnt
101 run
102 si,si
110 air
*
* controle do tempo e da impressao
*      end    min     max    ssdtt   minor   major   restart
*      time   tsp     tsp            edit    edit     freq
201     100.  1.0-8   0.2       00003     10    100    1000000
202     120.  1.0-8   0.002     00003     10    100    1000000
*
* edicoes menores (usado para ver determinadas variaveis no arq .OUT)
301  mflowj    100030000    *m' em kg/s
302  p         100030000    *pressao em Pa
303  tempf     100030000    *Temperatura Fluido em K
304  voidg     100030000    *fracao de vazio (de 0 a 1)
*
* controle dos trips
401  time  0 gt null   0   100.01  1    *abre 100% a valvula instantaneamente,
apos 1s.
*
*volumes hidrodinamicos
*Pipe com 23 volumes
1000000  PP01, pipe
1000001  23  *Nro de volumes do Pipe PP01
1000101  0.00785  23  *Area dos 23 volumes
1000201  0.0      22 *Area das juncoes entre os volumes. 0 signif. =AreaPipe
*Comprimentos dos volumes
*        L (m)     Nó/Volume
1000301  200.e-3   3
1000302  230.e-3   5
1000303  223.e-3   6
1000304  146.e-3   9
1000305  171.5e-3 10
1000306  200.e-3  13
1000307  100.e-3  14
1000308  192.5e-3 16
1000309  200.e-3  21
1000310  144.e-3  22
1000311  260.e-3  23
1000601  0.0      23       *angulo vertical dos 23 volumes
1000701  0.       23
1000801  0.000015  0.   23 *dados de friccao
1001001  0        23       *pipe volume ctrl flags
1001101  00000000 22       *pipe junctions ctrl flags
1001201  003   1.5e7  573.15  0. 0. 0. 23 *pipe cond.iniciais (150bar+300oC)
1001300  0
1001301  0.  0.  0.     22    *cond.iniciais nas juncoes (vel.inic.liq,
vel.ini.vapor)
*Valvula conectando Pipe com o ambiente
1050000  V01   valve
1050101  100230002 110010000 0.00785  0.    0.     0000200   1.0   0.14
1050201   1  0.0    0.0   0.0     *cond. inic. valvula em fluxo de massa
1050300  trpvlv  *tipo de valvula=Trip
1050301  401
*Ambiente
1100000  ambiente  tmdpvol
1100101  0.  10.0 10.0 0.0 0.0 0.0 0.0  0.0  0000000
1100200  004
1100201  0.0   1.0e5   291.1   1.0
1100202  1.e6  1.0e5   291.1   1.0
******
*HEAT PROPERTIES
******
11000000 23 10 2 1 0.05
11000100 0  1
11000101 9 0.053
11000201 1 9
11000301 0.0 9
```

```
 70)   11000400 0
 71)   11000401 573.15 10
 72)   11000501 100010000 10000 1 1 200.e-3   3
 73)   11000502 100040000 10000 1 1 230.e-3   5
 74)   11000503 100060000 10000 1 1 223.e-3   6
 75)   11000504 100070000 10000 1 1 146.e-3   9
 76)   11000505 100100000 10000 1 1 171.5e-3 10
 77)   11000506 100110000 10000 1 1 200.e-3  13
 78)   11000507 100140000 10000 1 1 100.e-3  14
 79)   11000508 100150000 10000 1 1 192.5e-3 16
 80)   11000509 100170000 10000 1 1 200.e-3  21
 81)   11000510 100220000 10000 1 1 144.e-3  22
 82)   11000511 100230000 10000 1 1 260.e-3  23
 83)   11000601 -700 0 3800 1 200.e-3   3
 84)   11000602 -700 0 3800 1 230.e-3   5
 85)   11000603 -700 0 3800 1 223.e-3   6
 86)   11000604 -700 0 3800 1 146.e-3   9
 87)   11000605 -700 0 3800 1 171.5e-3 10
 88)   11000606 -700 0 3800 1 200.e-3  13
 89)   11000607 -700 0 3800 1 100.e-3  14
 90)   11000608 -700 0 3800 1 192.5e-3 16
 91)   11000609 -700 0 3800 1 200.e-3  21
 92)   11000610 -700 0 3800 1 144.e-3  22
 93)   11000611 -700 0 3800 1 260.e-3  23
 94)   11000701 0 0.  0. 0. 23
 95)   11000801 0. 10. 10. 0. 0. 0. 0. 1. 23
 96)   11000901 0. 10. 10. 0. 0. 0. 0. 1. 23
 97)   *Material do Pipe 100=ACO
 98)   20100100 s-steel
 99)   *Tabela 700: Temperatura Ambiente
100)   20270000 temp
101)   20270001 0.     0.
102)   20270002 99.999 0.
103)   20270003 100.00 2.93e2
104)   20270004 1.e6   2.93e2
105)   *Tabela 800: Coef. Transf. Calor do Ar
106)   20280000 htc-t
107)   20280001 0.     0.
108)   20280002 99.999 0.
109)   20280003 100.0  1.e1
110)   20280004 1.e6   1.e1
111)   .
```

APÊNDICE H - Dados de entrada do código RELAP5 para a Experiência SUPER CANON criados com o pré-processador

```
 1)   =Experiencia SUPER CANON
 2)   *** Author.......: Daniel Monaco
 3)   *** Creation Date: 2019-05-31
 4)   *** Version......: v.1.0
 5)   *** Remarks ***
 6)   ***Demonstracao de uso do FastLAP na montagem dos cartoes de entrada
 7)   ***referentes a Experiencia SUPER CANON
 8)   ***
 9)   *Initialization
10)   100 new transnt
11)   101 run
12)   102 si si
13)   110 air
14)   *Time and Print Control
15)   ***Use these rows below as templates to setup Time and Print Control
16)   201 100. 1.0-8 0.2 00003 10 100 1000000
17)   202 120. 1.0-8 0.002 00003 10 100 1000000
18)   *Minor Edits
19)   ***Use the rows below as templates to control Minor Edits
20)   301 mflowj 100030000
21)   302 p 100030000
22)   303 tempf 100030000
23)   304 voidg 100030000
24)   *Trip Control
25)   ***Use the rows below as templates to create new Trips
26)   401 time 0 gt null 0 100.01 l *Trip 401 Remarks
27)   *Hydrodynamic Components
28)   *Heat Structures
29)   *Heat Structures Thermal Properties
30)   *General Table
31)   *Component 100-PP01::
32)   1000000 PP01 pipe
33)   1000001 23
34)   *Component 100-PP01: X-Coord Area (m?, ft?)
35)   1000101 0.007853982 23
36)   *Component 100-PP01: X-Coord Length (m, ft)
37)   1000301 200.e-3 03
38)   1000302 230.e-3 05
39)   1000303 223.e-3 06
40)   1000304 146.e-3 09
41)   1000305 171.5e-3 10
42)   1000306 200.e-3 13
43)   1000307 100.e-3 14
44)   1000308 192.e-3 16
45)   1000309 200.e-3 21
46)   1000310 144.e-3 22
47)   1000311 260.e-3 23
48)   *Component 100-PP01: Vertical Angle ?
49)   1000601 0. 23
50)   *Component 100-PP01: X-Coord Friction Data
51)   1000801 0.000015 0. 23
52)   *Component 100-PP01: X-Coord Control Flag
53)   1001001 0 23
54)   *Component 100-PP01: Volume Initial Conditions
55)   1001201 003 1.5e7 573.15 0. 0. 0. 23
56)   *Component 100-PP01: X-Coord Elevation Changes
57)   1000701 0. 23
58)   *Component 100-PP01: Junction Control Flag
59)   1001101 00000000 22
60)   *Component 100-PP01: Junction Initial Conditions
61)   1001301 0. 0. 0. 22
62)   *Component 100-PP01: Junction Area (m?, ft?)
63)   1000201 0. 22
64)   *Component 105-VV01::valvula trip simulando rompimento tubulacao
65)   1100000 AMB tmdpvol
66)   *Component 110-AMBIENTE: Area (m?, ft?)
67)   1100101 0.
68)   *Component 110-AMBIENTE: Length (m, ft)
69)   1100102 10.
70)   *Component 110-AMBIENTE: Volume (m?, ft?)
71)   1100103 10.
```

```
 72)   *Component 110-AMBIENTE: Azimuthal Angle ?
 73)   1100104 0.
 74)   *Component 110-AMBIENTE: Inclination Angle
 75)   1100105 0.
 76)   *Component 110-AMBIENTE: Elev. Change (m, ft)
 77)   1100106 0.
 78)   *Component 110-AMBIENTE: Wall roughn. (m, ft)
 79)   1100107 0.
 80)   *Component 110-AMBIENTE: Hydr. Diam. (m, ft)
 81)   1100108 0.
 82)   *Component 110-AMBIENTE: Volume control flags
 83)   1100109 0
 84)   *Component 110-AMBIENTE: Volume Data Control Word
 85)   1100200 004
 86)   *Component 110-AMBIENTE: Volume Data
 87)   1100201 0. 1.e5 291.1 1.
 88)   1100202 1.e6 1.e5 291.1 1.
 89)   *Component 105-VV01::valvula trip simulando rompimento tubulacao
 90)   1050000 VV01 valve
 91)   *Component 105-VV01: Connect From
 92)   1050101 100230002
 93)   *Component 105-VV01: Connect To
 94)   1050102 110010001
 95)   *Component 105-VV01: Area (m?, ft?)
 96)   1050103 0.007853982
 97)   *Component 105-VV01: Reynolds Af coef.
 98)   1050104 0.
 99)   *Component 105-VV01: Reynolds Ar coef.
100)   1050105 0.
101)   *Component 105-VV01: Junction control flags
102)   1050106 00000200
103)   *Component 105-VV01: Henry-Fauske Critical Flow Mode Variables
104)   1050108
105)   *Component 105-VV01: Junction Initial Conditions
106)   1050201 1 0. 0. 0
107)   *Component 105-VV01: Valve Type
108)   1050300 "trpvlv"
109)   *Component 105-VV01: Trip Valve Data
110)   1050311 401
111)   *Heat Composition 001::Casco do Pipe PP01
112)   20100100 "s-steel" 1 1
113)   *General Table 700::Temperatura ambiente
114)   20270000 "temp"
115)   *General Table 700: General Table Data
116)   20270001 0. 0.
117)   20270002 99.999 0.
118)   20270003 100. 2.93e2
119)   20270004 1.e6 2.93e2
120)   *General Table 800::Coeficientes de Transferencia de Calor do Ar
121)   20280000 "htc-t"
122)   *General Table 800: General Table Data
123)   20280001 0. 0.
124)   20280002 99.999 0.
125)   20280003 100. 1.e1
126)   20280004 1.e6 1.e1
127)   *Heat Structure 1000::Casco do PIPE PP01 trocando calor entre o fluido e o
       ambiente
128)   11000000 23 10 2 1 0.05 0 0 2
129)   *Heat Structure 1000: Heat Structure Mesh Flags
130)   11000100 0 1
131)   *Heat Structure 1000: Mesh Interval (m, ft)
132)   11000101 9 0.053
133)   *Heat Structure 1000: Composition Number
134)   11000201 1 9
135)   *Heat Structure 1000: Heat Source Value Or Gamma Attenuation Coef.
136)   11000301 0. 9
137)   *Heat Structure 1000: Initial Temperature Flag
138)   11000400 0
139)   *Heat Structure 1000: Temperature (K, ?F)
140)   11000401 573.15 10
141)   *Heat Structure 1000: Left Boundary
142)   11000501 100010000 10000 1 1 200.e-3 3
143)   11000502 100040000 10000 1 1 230.e-3 5
144)   11000503 100060000 10000 1 1 223.e-3 6
145)   11000504 100070000 10000 1 1 146.e-3 9
146)   11000505 100100000 10000 1 1 171.5e-3 10
147)   11000506 100110000 10000 1 1 200.e-3 13
```

```
148)   11000507 100140000 10000 1 1 100.e-3 14
149)   11000508 100150000 10000 1 1 192.e-3 16
150)   11000509 100170000 10000 1 1 200.e-3 21
151)   11000510 100220000 10000 1 1 144.e-3 22
152)   11000511 100230000 10000 1 1 260.e-3 23
153)   *Heat Structure 1000: Right Boundary
154)   11000601 -700 0 3800 1 200.e-3 3
155)   11000602 -700 0 3800 1 230.e-3 5
156)   11000603 -700 0 3800 1 223.e-3 6
157)   11000604 -700 0 3800 1 146.e-3 9
158)   11000605 -700 0 3800 1 171.5e-3 10
159)   11000606 -700 0 3800 1 200.e-3 13
160)   11000607 -700 0 3800 1 100.e-3 14
161)   11000608 -700 0 3800 1 192.e-3 16
162)   11000609 -700 0 3800 1 200.e-3 21
163)   11000610 -700 0 3800 1 144.e-3 22
164)   11000611 -700 0 3800 1 260.e-3 23
165)   *Heat Structure 1000: Heat Source Data
166)   11000701 0 0. 0. 0. 23
167)   *Heat Structure 1000: Additional Left Boundary Option
168)   11000800 0
169)   *Heat Structure 1000: Additional Left Boundary Parameters
170)   11000801 0. 10. 10. 0. 0. 0. 0. 1. 23
171)   *Heat Structure 1000: Additional Right Boundary Parameters
172)   11000901 0. 10. 10. 0. 0. 0. 0. 1. 23
173)   .
```

APÊNDICE I - Visão Geral de Curvas Homólogas de Potência

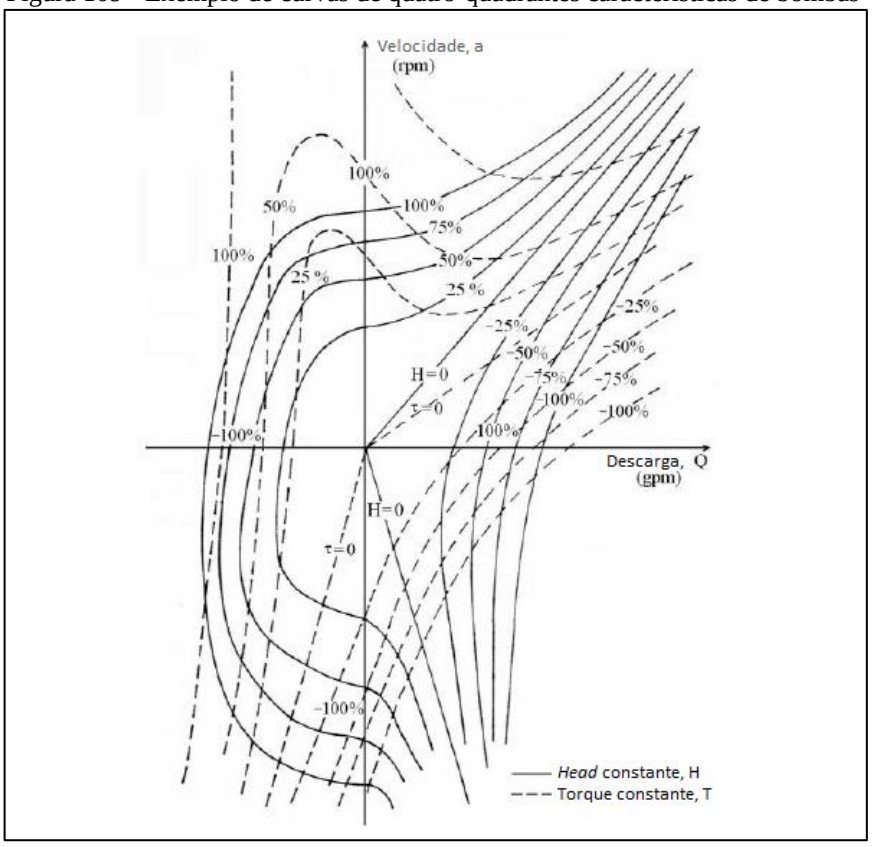
Em geral, quando usados códigos de análise de transiente, tais como o código RELAP5, as características de uma bomba são inseridas no código no formato de curvas homólogas. Esses dados, em geral, são fornecidos pelos próprios fabricantes das bombas hidráulicas em uso no estudo. Tal método será brevemente explanado aqui.

A fim de obtermos um mapa completo das características de uma bomba, costuma-se plotar a taxa de fluxo normalizada ν sobre a o eixo das abcissas, e sobre o eixo das ordenadas costuma-se plotar a velocidade normalizada de rotação ω. Um exemplo de curvas de quatro-quadrantes é mostrado na Figura 109.

O plano ν x ω resultante é então dividido em quatro quadrantes, a saber:

- Quadrante Normal (N), onde $\nu > 0$ e $\omega > 0$;
- Quadrante de Dissipação (D), onde $\nu < 0$ e $\omega > 0$;
- Quadrante de Turbina (T), onde $\nu < 0$ e $\omega < 0$;
- Quadrante Reverso (R), onde $\nu > 0$ e $\omega < 0$.

Figura 109 - Exemplo de curvas de quatro-quadrantes características de bombas



Fonte: VELOSO; MATTOS, 2015.

Uma vez definidos esses quatro quadrantes, cada um desses quadrantes é dividido em dois intervalos, através de linhas retas de 45º partindo-se da origem e indo para o extremo de cada quadrante. Cada uma dessas subdivisões receberá um determinado nome caso obedeçam às seguintes condições:

- se $|\nu/\omega| \leq 1$ => intervalo A;
- se $|\omega/\nu| < 1$ => intervalo V.

Seguindo esse conceito, o plano ν x ω pode ser separado em oito regiões, recebendo os nomes conforme a definição acima, a saber: VN, NA, AD, VD, VT, AT, AR e VR.

De acordo com VELOSO e MATTOS (2015), a desvantagem em usar as curvas de quatro quadrantes para proposições numéricas são:

- a necessidade de interpolação bidimensional;
- a grande quantidade de pontos necessárias para definir todo o domínio;
- e o fato de o domínio ser infinito em extensão.

Essas dificuldades, no entanto, podem ser superadas pelo desenvolvimento das curvas homólogas, usando uma transformação homóloga baseada nas relações de semelhança com uma bomba centrífuga. Essa transformação colapsa as curvas características dos quatro quadrantes em uma curva simples, limitada e adimensional cobrindo os oito intervalos definidos. Assim se obtém um conjunto de oito curvas para *head* (ou torque).

Novamente, pode-se reescrever as grandezas ν e ω, normalizando-as em relação às medidas de referências conforme medidas no gráfico. Por exemplo, pode-se assumir um determinado valor de velocidade onde ocorre o ponto máximo de eficiência da bomba, e adotamos esses pontos como nosso referencial. A partir daí, reescrevemos as grandezas não mais como valores absolutos, mas como proporções em relação a esses referenciais adotados (representados abaixo com o símbolo $_0$). E assim, teremos:

- $v = Q/Q_0$, sendo a taxa de fluxo volumétrico;
- $a = \omega/\omega_0$, ou a taxa de velocidade;
- $h = H/H_0$, ou a taxa de *head* dinâmico;
- $b = T/T_0$, ou taxa de torque hidráulico.

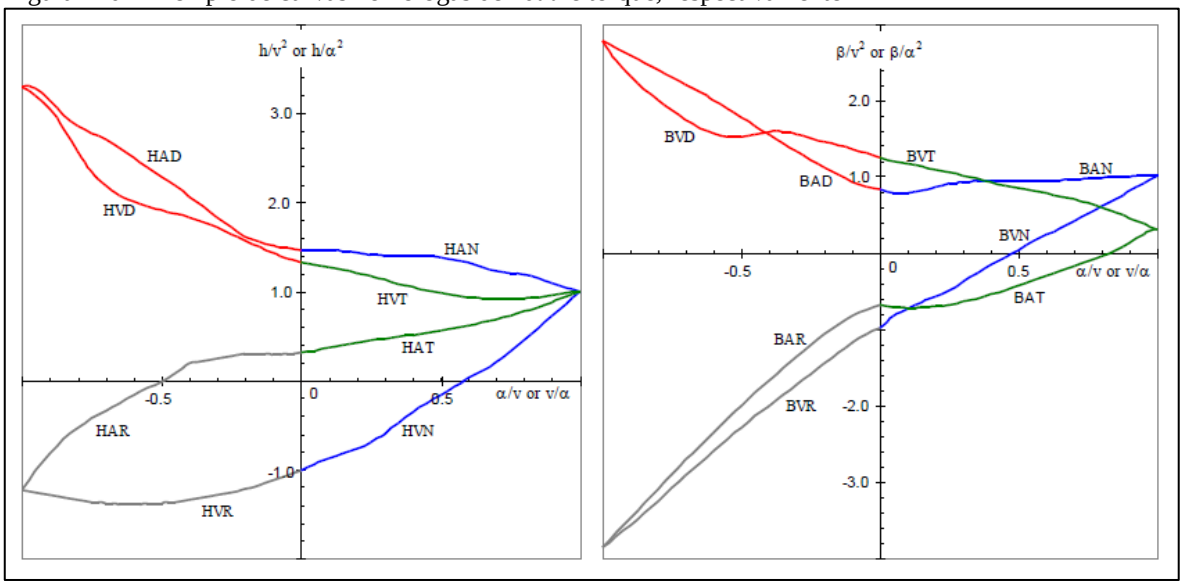
Via de regra, as bombas podem ser descritas com a taxa de fluxo sendo proporcional à velocidade da bomba, e o *head* e o torque como proporcionais ao quadrado

da velocidade da bomba. Desta forma, podemos definir um conjunto de oito curvas, uma para cada um dos oito intervalos, usando as seguintes definições:

- intervalos A: X=v/a, Y=h/a^2 ou b/a^2;
- intervalos B: X=a/v; Y=h/v^2 ou b/v^2.

As curvas homólogas são em geral representadas por um conjunto de três letras, sendo a primeira H (para curvas de *head*) ou B (para curvas de torque), a segunda letra sendo A ou B conforme visto acima, e a terceira letra sendo N, D, T ou R, conforme os quadrantes definidos no início. Um exemplo típico de curvas homólogas de torque e *head* para bombas centrífugas similares às usadas em ciclos de refrigeração primários de reatores refrigerados à água do tipo PWR são mostrados na Figura 110.

Figura 110 - Exemplo de curvas homólogas de *head* e torque, respectivamente



Fonte: VELOSO; MATTOS, 2015.

As curvas homólogas formam, portanto, uma espécie de mapeamento, onde as diversas curvas sobre o plano ν x ω são convertidas em oito curvas correspondentes aos oito intervalos mencionados anteriormente.

APÊNDICE J - Programação orientada a objetos

A Programação Orientada a Objetos (OOP) é um conceito de programação que visa organizar os programas de maneira a agrupar tanto os dados quanto os operadores destes em unidades lógicas chamadas de objetos, encapsulando tanto informações quanto comportamentos dentro de uma unidade. Um programa orientado a objetos é um programa que resolve um ou mais problemas por meio da interação entre diferentes objetos com diferentes funcionalidades.

Como foi observado por NELSON (1990), uma das principais características das linguagens orientadas a objeto são a presença de objetos, classes e o conceito de herança. Atualmente, são consideradas linguagens orientadas a objetos quando elas dão suporte a cinco características principais:

- Classes;
- Objetos;
- Encapsulamento;
- Herança;
- Polimorfismo.

As linguagens orientadas a objeto também implementam atualmente outras funcionalidades, tais como classes abstratas, eventos e outros. Porém, as cinco características acima são consideradas essenciais para uma linguagem orientada a objetos.

Objetos e classes

Uma forma fácil de apresentar o conceito de um objeto é pensar em alguns objetos do mundo real, por exemplo, um rádio. Um rádio possui diversas características e propriedades. Você não precisa abrir o rádio para que ele funcione adequadamente; o rádio tem uma série de componentes internos para seu funcionamento, porém apenas alguns componentes externos para que você possa controla-lo; um rádio pode ser conectado a outros objetos (como um microfone, por exemplo), desde que as interfaces disponíveis por ambos sejam iguais; o rádio está completo quando adquirido e todos os seus requisitos externos estão bem documentados; desde que usado conforme especificado, então, espera-se que o rádio também funcione como especificado pelo fabricante.

Outro comportamento esperado em um rádio conforme apresentado é como suas funcionalidades afetam somente o objeto a elas ligado. O botão de sintonia, por exemplo, só altera a sintonia do rádio ao qual o controle está acoplado: não é esperado que, mesmo que

“A” e “B” possuam rádios idênticos, quando “A” utilizar o botão de sintonia do seu rádio, o rádio de “B” venha a ter sua sintonia alterada. Apesar de ambos serem rádios idênticos, tratam-se de dois objetos distintos.

Pode-se definir o conceito de classe como a provedora de um modelo de objeto: é a classe quem define a interface de como seus objetos serão acessados, como eles podem ser usados, que informações serão disponibilizadas ao público. Às informações de estado desse objeto, atualmente dá-se o nome de propriedades; aos controles para manipulação dos objetos oriundos dessa classe, dá-se o nome de métodos. Pode-se pensar também na classe como sendo a planta de uma casa: a planta não é uma casa, mas define quais características essa casa terá; a casa é construída a partir dessa planta, cada casa será um objeto, mas guardará as características conforme definida na planta; quando encontradas duas casas criadas a partir da mesma planta, sabe-se que ambas as casas possuem as mesmas características, conforme definido na planta.

Dessa forma, pode-se conceituar um objeto como sendo uma instância de uma determinada classe. O rádio de “A” é uma instância da classe “rádio”, enquanto o rádio de “B” é uma outra instância da mesma classe; desta forma, ao usar os métodos do objeto rádio de “A”, somente se verificará alterações no objeto rádio de “A”, e nunca no rádio de “B”.

Encapsulamento

Voltando ao exemplo do rádio, esse equipamento possui uma série de componentes internos para seu adequado funcionamento. Porém, esses componentes podem ser acessados somente internamente, e não devem ser acessados de fora. Para manipular o rádio, é possível fazê-lo somente pelos controles disponibilizados pelo mesmo para tal fim, tais como volume, sintonia, ligar e desligar, entre outras funções.

Define-se como encapsulamento a característica de agrupar todos os dados e componentes necessários para um correto funcionamento de determinado objeto, de forma que todo o mecanismo de funcionamento do objeto fique oculto. Assim, seus usuários não precisam entender a completa implementação de um objeto para poder utilizá-lo, mas tão somente entender as interfaces que esse objeto apresenta: da mesma forma que ninguém precisa ser engenheiro elétrico para ouvir rádio, ou dirigir um carro.

Herança

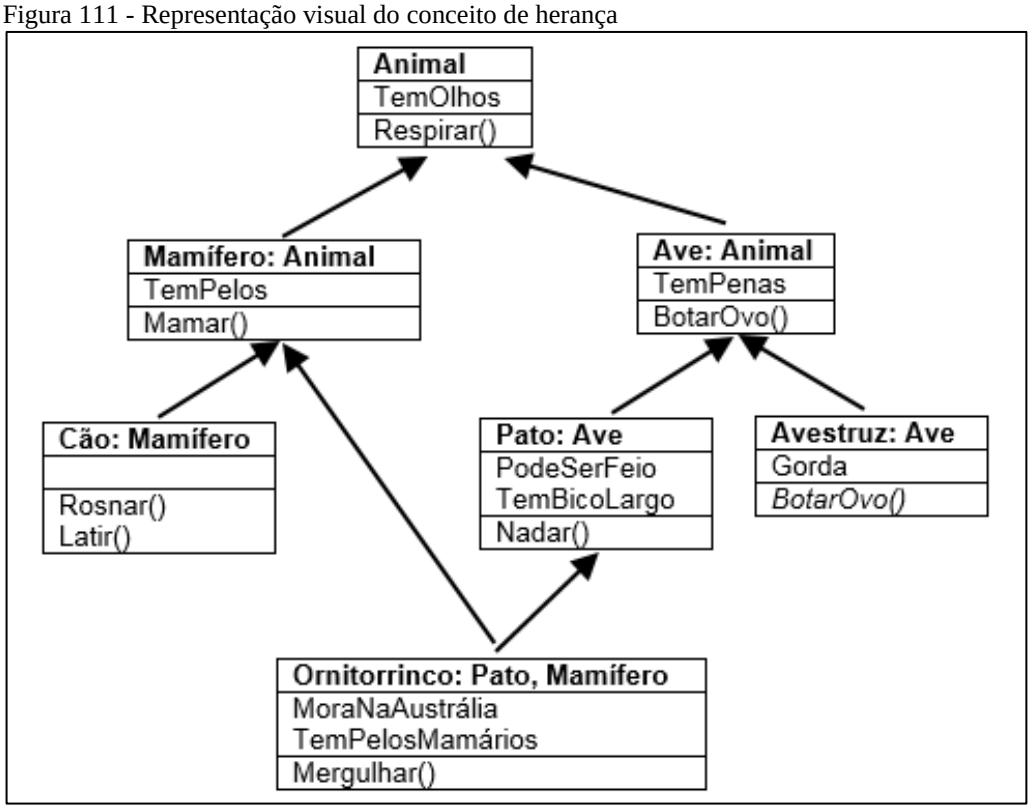
Uma das características principais de uma linguagem orientada a objetos é a Herança (NELSON, 1990). Sem ela, dificilmente uma linguagem pode ser considerada como orientada a objetos.

De forma bem simplista, a herança é um tipo de mecanismo de compartilhamento de funcionalidade. A herança permite que uma nova classe possa ser definida como uma especialização de uma classe existente, sem que seja necessário que todas as linhas de programação da classe existente sejam copiadas para a nova classe. A classe herdeira, também chamada de subclasse, herdará todos os métodos e propriedades da classe raiz (ou superclasse), incluindo até mesmo os métodos e propriedades que essa superclasse tenha herdado de outra classe.

A herança pode ser simples (chamada também somente de herança), ou pode ser uma herança múltipla, onde uma subclasse pode herdar propriedades de várias superclasses simultaneamente.

A Figura 111 exemplifica o conceito de herança simples e múltipla.

Figura 111 - Representação visual do conceito de herança



Fonte: autor da dissertação.

Na Figura 111 é mostrado que a classe “cão” é uma subclasse de “mamífero”, que por sua vez é uma subclasse de “animal”. Sendo assim, pode-se afirmar com segurança que uma instância de “cão” pode respirar, mamar, rosnar e latir, pois as características de mamar e respirar foram herdadas das superclasses de “cão” por meio de herança simples. Da mesma forma, qualquer instância da classe “pato” (até mesmo o “Patinho Feio”), também pode respirar e botar ovos (desde que a propriedade fêmea seja igual a verdadeiro). Já a

classe “ornitorrinco” define uma classe bizarra (tão bizarra quanto o animal em questão), uma vez que ele faz uma herança múltipla tanto de um mamífero quanto de um pato. Assim, uma instância da classe “ornitorrinco” tem bico de pato e pode nadar, botar ovos e mamar, uma vez que todos esses métodos foram implementados nas superclasses que o precederam, além de poder mergulhar, conforme definido na classe “ornitorrinco”.

Desta forma, caso futuramente a classe “ave” acima venha a ser alterada para implementação do método “voar” na mesma, automaticamente as instâncias de “avestruz” e “ornitorrincos” passariam a poder voar também, uma vez que ambas são classes herdeiras de “ave”.

Polimorfismo

Polimorfismo significa “múltiplas formas”. Em geral, o polimorfismo é implementado pelas linguagens orientadas a objetos por meio de dois outros conceitos: *overriding* (sobreposição) e *overloading* (sobrecarga). Como foi dito no item anterior, quando uma classe herda a estrutura de outra classe ela herda tanto suas propriedades quanto seus métodos. Assim, voltando à Figura 111, a classe “cão” herdará todo o comportamento da classe “mamíferos” quanto da classe “animal”. Analogamente, um objeto da classe “avestruz” apresenta as propriedades e métodos tanto da classe “ave” quanto da classe “animal”. De outra forma, caso o programador deseje fazer uma listagem de todas as instâncias de animais, ele poderia listar todos as instâncias de “avestruz” tratando-os como se fossem instâncias da classe “animal”.

Entretanto, apesar de uma subclasse ter obrigatoriamente que apresentar as mesmas propriedades e métodos das suas superclasses, os comportamentos desses métodos podem ser diferentes. No caso, tanto a classe “avestruz” quanto a classe “pato” podem botar ovos como qualquer ave: porém, o método “BotarOvos” foi sobrescrito na subclasse “avestruz”, para que o resultado desse método fosse diferente do das demais aves (afinal de contas, o ovo de avestruz é mesmo enorme). Nesse cenário, caso o programador deseje fazer uma listagem para forçar todas as instâncias de “aves” botarem ovos, todas as instâncias botariam ovos de “ave”, incluindo aqui as instâncias de “ornitorrinco”; porém as instâncias de “avestruz” se comportariam como uma “ave” qualquer, com exceção que essas botariam um ovo enorme, conforme definido na classe “avestruz”. Esse exemplo define o conceito polimórfico de sobreposição (*overriding*).

Já a sobrecarga pode ser implementada quando a linguagem de programação permite que um método diferente possa ser chamado pelo mesmo nome de outro método já

existente, diferindo um do outro somente pelos parâmetros de entrada definidos para esses métodos. Um exemplo de sobreposição, usando-se novamente a definição de classes mostrada na Figura 111, seria a implementação de um método “BotarOvos()” e a implementação de outro método homônimo, mas com parâmetros diferentes, como por exemplo “BotarOvos(quantidade)”. O programador que estivesse usando essa classe saberia claramente que ambos os métodos servem para sua “ave” botar ovos, mas a depender da situação, ele poderia chamar um ou outro método conforme fosse necessário.

INSTITUTO DE PESQUISAS ENERGÉTICAS E NUCLEARES
Diretoria de Pesquisa, Desenvolvimento e Ensino
Av. Prof. Lineu Prestes, 2242 – Cidade Universitária CEP: 05508-000
Fone/Fax(0XX11) 3133-8908
SÃO PAULO – São Paulo – Brasil
http://www.ipen.br

O IPEN é uma Autarquia vinculada à Secretaria de Desenvolvimento, associada à Universidade de São Paulo e gerida técnica e administrativamente pela Comissão Nacional de Energia Nuclear, órgão do Ministério da Ciência, Tecnologia, Inovações e Comunicações.