UNIVERSIDADE DE PASSO FUNDO

PROGRAMA DE PÓS-GRADUAÇÃO

EM PROJETO E PROCESSOS DE FABRICAÇÃO

Área de concentração: Projeto e Processos de Fabricação

Dissertação de Mestrado

DESENVOLVIMENTO DE UMA APLICAÇÃO PARA AUTOMAÇÃO DE PROJETOS 3D: UM ESTUDO DE CASO NA INDÚSTRIA DE ÔNIBUS

Roberto Busetto

Passo Fundo

2024

UPF

CIP – Catalogação na Publicação

B977d Busetto, Roberto
Desenvolvimento de uma aplicação para automação de projetos 3D [recurso eletrônico] : um estudo de caso na indústria de ônibus / Roberto Busetto. – 2017.
6.5 MB ; PDF.

Orientador: Prof. Dr. Márcio Walber.
Dissertação (Mestrado em Projeto e Processos de Fabricação) – Universidade de Passo Fundo, 2017.

1. Processos de fabricação. 2. Automação Industrial. 3. Ônibus - Indústria. I. Walber, Márcio, orientador. II. Título.

CDU: 681.5

Catalogação: Bibliotecária Juliana Langaro Silveira - CRB 10/2427

Roberto Busetto

DESENVOLVIMENTO DE UMA APLICAÇÃO PARA AUTOMAÇÃO DE PROJETOS 3D: UM ESTUDO DE CASO NA INDÚSTRIA DE ÔNIBUS

Dissertação apresentada ao Programa de Pós-graduação em Projeto e Processos de Fabricação da Universidade de Passo Fundo, como requisito para obtenção do grau de Mestre em Projeto e Processos de Fabricação.

Data de aprovação: 25 Maio de 2017.

Os componentes da Banca examinadora abaixo aprovaram a Dissertação:

Professor Doutor Márcio Walber
Orientador

Professor Doutor Alexandre Baroni
Universidade de Caxias do Sul

Professor Doutor Nilson Luis Maziero
Universidade de Passo Fundo

Professor Doutor José Antônio Portella
Universidade de Passo Fundo

AGRADECIMENTOS

Agradeço ao meu orientador Prof. Dr. Márcio Walber por ter acreditado em mim e ter me ajudado e apoiado constantemente na produção deste trabalho. Agradeço também a minha banca qualificadora Prof. Dr. Nilson Luis Maziero e Prof. Dr. José Antônio Portella pelo direcionamento e instruções a certa da minha dissertação.

Sou muito grato aos antigos colegas de trabalho: Carlos Frederico Viero, Marlos Link, Oséias Esmelindro, André Petry, Márcio Paviani, Carlos Molozzi, Thiago Zilio e Diego Zilio por terem despertado em mim o interesse pelo tema e pelas ricas discussões. Gostaria de agradecer também a consultoria da empresa Produttare, na pessoa do Prof. Dr. Alexandre Baroni.

Agradeço também a empresa Siemens PLM pelo apoio financeiro e aos momentos aos quais tive que me ausentar. Aos meus atuais colegas de trabalho Rudimar Luis Ronsoni Jr. e Ricardo Medeiros pelas contribuições a esta dissertação.

Dedico esta dissertação a minha mãe Inésia Coan Busetto, meu irmão Renato Busetto e minha esposa Gabriela Grezzana e minha filha Clara Grezzana que me deram total apoio para que eu pudesse desenvolver este trabalho.

Em memória a Antônio Luiz Busetto, meu pai.

“Não é o mais forte que sobrevive, nem o mais inteligente, mas o que melhor se adapta às mudanças”.

Charles Darwin

RESUMO

A produção de carrocerias e ônibus é complexa e onerosa, com inúmeros opcionais de venda, orçamentos e vendas, trabalhos administrativos, fornecedores e inúmeras ordens de produção. Essas condições, durante o ciclo de vida do produto, geram produtos exclusivos para cada cliente, com elevado número de horas de engenharia. Tal conhecimento de engenharia acaba, muitas vezes, sendo perdido em anotações e arquivos pessoais, gerando atrasos em entregas de projetos, tempo de projeto desnecessário, retrabalho e erros em projeto. Neste contexto, o presente trabalho visa desenvolver uma aplicação computacional, integrada ao *software* Siemens NX para automação de projetos 3D, combinando abordagens de configuração de produto com técnicas de automação de projeto, a fim de apoiar as atividades de projeto e desenvolvimento de produtos. A abordagem é baseada em entidades inteligentes tridimensionais (*templates*) que encapsulam regras de configuração de produto e a lógica de criação de projetos, absorvendo as diversas características e variações escolhidas pelo cliente, unificando conhecimento de produto, geometrias, estrutura de produto, reuso de projeto e configuração de venda. Esta abordagem é aplicada por meio de um estudo de caso em uma encarroçadora de ônibus, em que foram utilizados dois projetos distintos. Ao final, concluiu-se que os objetivos foram alcançados e que a aplicação, além de ser estável, possui escalabilidade para ser empregada em toda a linha de produto da empresa e aplicada diretamente nos setores de Engenharia de Desenvolvimento e Engenharia de Produto.

Palavras-chave: Automação de Projetos; Modularização; Configuração de Produto; Modelagem Paramétrica; Ônibus

ABSTRACT

The production of bus bodies is complex and costly, with countless sale's optional, budgets and sales, administrative works, providers and innumerable production orders. These conditions, during the product's life cycle, generate exclusive products to each client with many hours of engineering work. Such knowledge of engineering ends, many times, getting lost in notes and personals archives, generating projects delivery delays, unnecessary time of work, rework and project errors. In this context, the present work aims to develop a computational application, integrated with *software* Siemens NX to automation of 3D projects, combining approaches of product configuration with technics of project automation, to support activities of project and development of products. The approach is based in intelligent tridimensional entities (*templates*) that encapsulate project configuration rules and the logical of project creation, absorbing many characteristics and variations, chosen by the client, unifying product knowledge, geometries, product structure, reuse of the project and sales configuration. This approach is applied through a study of case in a bus factory, where were used two different projects. In the end, it was concluded that the objectives has been achieved and that the application, beyond being stable, has scalability to be applied in every product line of the factory bus, and directly applied in development engineering and product engineering sectors.

Keywords: Project Automation; Modularization; Product Configuration; Parametric Design; Bus

LISTA DE FIGURAS

Figura A. 1 - Interface do *software* Teamcenter 128
Figura A. 2 - Interface do *software* NX 129
Figura A. 3 - Exemplo de utilização de expressão 130
Figura A. 4 - Interface para criação de expressões 130
Figura A. 5 - Exemplo de *part family* 131
Figura A. 6 - Fluxo de *workflow* para itens produzidos 132
Figura A. 7 - Fluxo de *workflow* para itens comprados 133
Figura A. 8 - Exemplo de tarefa de venda 134
Figura A. 9 - Interface do *software* configurador de produto 134
Figura A. 10 - Configuração de produto 135

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

3D	*Three Dimensional*
CAD	*Computer Aided Design*
CAE	*Computer Aided Engineering*
CAID	*Computer Aided Industrial Design*
CAM	*Computer Aided Manufacturing*
CAPE	*Computer Aided Production Engineering*
CAPP	*Computer Aided Process Planning*
CASE	*Computer Aided Software Engineering*
CIM	*Computer Integrated Manufacturing*
CRM	*Customer Relationship Management*
DECM	*Digital Engineering Content Management*
DFMA	*Design for Manufacture and Assembly*
DSR	*Design Science Research*
ECM	*Enterprise Content Management*
EDA	*Electronic Design Automation*
EDI	*Electronic Data Interchange*
EDM	*Engineering Data Management*
EDM	*Engineering Data Management*
ERP	*Enterprise Resource Planning*
FABUS	Associação Nacional dos Fabricantes de Ônibus
FEA	*Finite Element Analysis*
IPM	*Intellectual Property Management*
KBS	*Knowledge Based Systems*
LCA	*Life Cycle Analysis*
MRP	*Manufacturing Resource Planning*
NC	*Numerical Control*
PDM	*Product Data Management*
PM	*Project Management*
RP	*Rapid Prototyping*
SCM	*Software Configuration Management*
SCM	*Supply Chain Management*
SE	*Systems Engineering*
TDM	*Technical Document Management*
VR	*Virtual Reality*

SUMÁRIO

1 INTRODUÇÃO

O contexto competitivo atual, enfrentado pelas empresas encarroçadoras de ônibus, é de aumento na variedade de configurações – pela necessidade de agregar valor ao produto –, diminuição do ciclo de vida do produto, manutenção dos custos e qualidade.

Essa variedade de configurações aumenta os custos de desenvolvimento de produto, aumenta os prazos de fabricação e diminui o retorno econômico para a organização. Um ônibus pode sofrer variações de diversas maneiras, como, por exemplo, o comprimento do veículo, que está ligado diretamente à capacidade de passageiros desejada. Essas variações acabam criando produtos exclusivos.

A produção de carrocerias de ônibus é bastante complexa e trabalhosa, exigindo muita engenharia, vendas, trabalho administrativo, contratos e pagamentos de fornecedores, ou seja, uma infinidade de ordens de produção que precedem o início da produção, condições estas que geram um sistema sob medida de fabricação, com variedades de processos e *part numbers* gerando um produto customizado, mas funcionalmente idêntico ao ônibus seguinte, na linha de produção (NAPPER, 2014).

Tabela 1 - Produção de carrocerias de ônibus no mercado brasileiro

Ano	Mercado Interno (un.)	Mercado Externo (un.)	Total (un.)
2015	17.157	3.802	20.959
2014	27.967	3.440	31.407
2013	28.596	4.097	32.693
2012	28.319	4.229	32.548
2011	31.554	3.977	35.531
2010	28.035	4.563	32.598
2009	20.990	3.903	24.893

Fonte: Associação Nacional dos Fabricantes de Ônibus (FABUS)

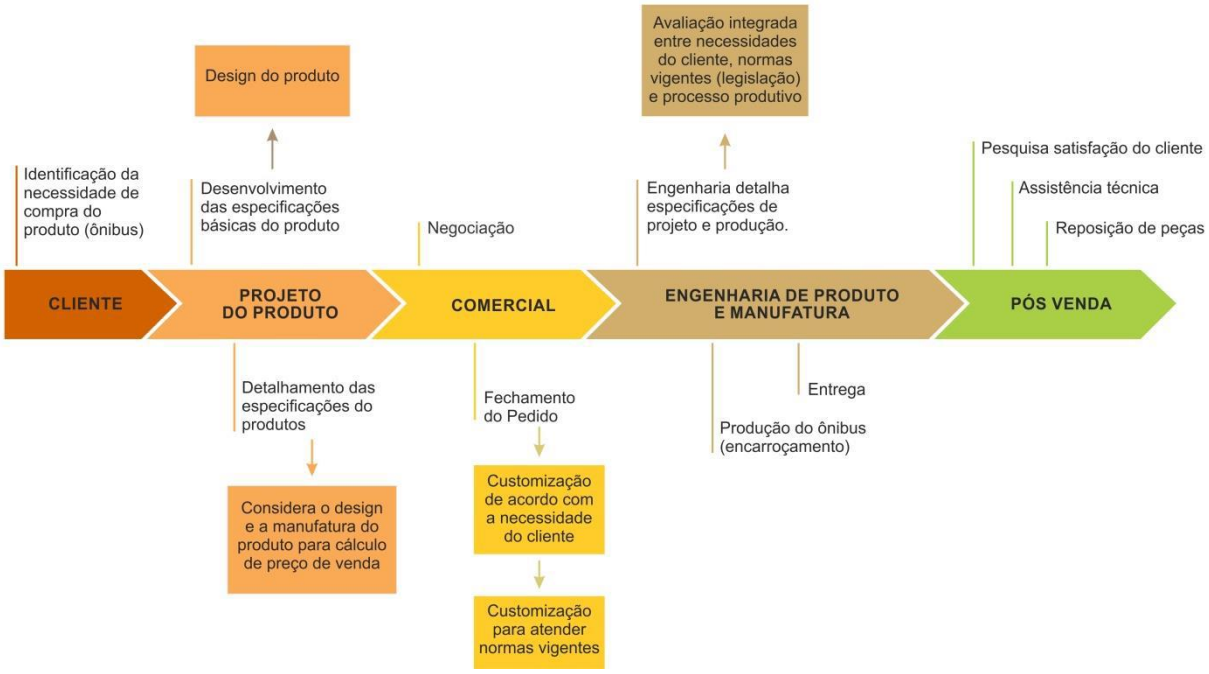
De acordo com os dados pesquisados na Associação Nacional dos Fabricantes de Ônibus (FABUS), a produção de carrocerias está dividida em quatro segmentos: Urbano, Rodoviário, Intermunicipal e Micro. Participam da produção nacional de ônibus as empresas: Marcopolo, Ciferal, Comil, Induscar, Irizar, Neobus e Mascarelo. Na Tabela 1 são apresentados os dados referentes à produção de carrocerias de ônibus no Brasil entre os anos de 2009 a 2015, segundo a FABUS. Na comparação entre os anos de 2014 e 2015, por exemplo, percebe-se uma retração de 33,3% na produção de carrocerias de ônibus.

1.1 Contexto

Em seu estudo, Napper (2014) apresenta um modelo de produção de carrocerias de ônibus. Conforme a Figura 1, o primeiro momento é o detalhamento e especificação de produto, atendendo as necessidades do cliente e entregando informações de venda para o departamento comercial gerar o fechamento do pedido junto ao cliente. Após a venda, engenharia de produto e manufatura trabalham no detalhamento e especificações de projeto e produção, para liberação do pedido para a fábrica.

Uma resposta rápida a este contexto competitivo é a adoção da personalização do produto. Este processo, sem uma metodologia de desenvolvimento adequada, não é indicado, pois gera projetos de produtos exclusivos para clientes, leva a altos níveis de estoque, uma grande variedade que gera variabilidade dos produtos em linha de produção (SALVADOR; FORZA; RUNGTUSANATHAM, 2002), bem como cria problemas para sequenciamento de produto em linha de produção (PALENCIA, A. E. R.; DELGADILLO, G. E. M, 2012), entre outros.

Figura 1 - Processo genérico das indústrias fabricantes de ônibus



Fonte: Adaptado de Napper (2014)

As empresas encarroçadoras de ônibus enfrentam esses problemas e estão desenvolvendo estruturas modulares para auxiliar na redução de problemas causados pela

diversidade de especificações de projeto e fabricação do veículo. Com um pequeno estoque de módulos, pode-se oferecer um amplo portfólio de produtos finais aos clientes (CUNHA, C.; AGARD, B.; KUSIAK, A., 2010; NAPPER, 2014; VIERO, 2013). A modularização de produto possibilita a reutilização de configuração de venda, gerando novas combinações de produtos, aumentando assim a variedade de produtos ofertados (PATEL; JAYARAM, 2014). Com isso é gerada uma grande massa de regras de variações de projeto e utilização de módulos e blocos intercambiáveis por cada produto produzido pela empresa, coexistindo então um dilema entre variedade de produtos e desempenho operacional.

As atividades de projetos mecânicos estão apoiadas em sistemas CAD 3D com recursos paramétricos. Um projeto paramétrico é aquele que possui a capacidade de se atualizar e reconstruir sua geometria a partir de uma mudança em um de seus parâmetros, como, por exemplo: espessura, comprimento, raio, altura, entre outros. As tecnologias CAD não são capazes de processar os resultados dos sistemas de configuração de produto, tendo como consequência o desenvolvimento manual de projeto, gerando desperdício de tempo em tarefas repetitivas (RAFFAELI, R.; MENGONI, M.; GERMANI, M., 2013).

Os projetistas estão familiarizados com ferramentas CAD 3D para representar suas soluções e pode-se aproximar os projetistas das ferramentas de automação mantendo a abordagem junto de suas atividades diárias. Essa aproximação dos projetistas pode ser benéfica para aquisição de conhecimentos nos modelos 3D. Com a disponibilidade de linguagens de *script*, a automação de modelos virtuais é cada vez mais viável, embora conhecimentos técnicos, incluindo programação de *software*, sejam necessários (ZORRIASSATIME et al., 2003).

Uma das respostas para estas necessidades é a introdução de *templates* de produto, ou protótipos virtuais configuráveis, uma extensão do conceito de produto paramétrico. Os *templates* são elementos tridimensionais em que o conhecimento (implícito e explícito) é armazenado, incluindo informações sobre a estrutura de produto, *layout*, restrições de montagem, normas, entre outras. Eles podem ser definidos como blocos inteligentes que encapsulam regras de configuração e a lógica de criação, absorvendo as diversas características e variações escolhidas pelo cliente (RAFFAELI, R.; MENGONI, M.; GERMANI, M., 2013).

O presente trabalho está inserido na concepção de uma solução para criação e automação de *templates* para projetos variantes.

1.2 Justificativa

As pesquisas em projeto de produto customizado incidem sobre a conversão de peças existentes em um conjunto válido de estruturas de produto. Com a escala e complexidade dos produtos configuráveis aumentando, as interdependências entre as demandas dos clientes e estruturas do produto tendem a crescer. Como resultado, surgem as estruturas do produto que não satisfazem as necessidades individuais do cliente e, portanto, são necessárias variantes do produto (REN, B.; QIU, L.; ZHANG, S.; TAN, J.; CHENG, J. 2013).

A empresa estudada possui uma divisão entre desenvolvimento de produtos e manutenção de produtos, sendo, respectivamente, os setores de Engenharia de Desenvolvimento e Engenharia de Produto. Após o lançamento de um novo modelo de carroceria, os projetos de produto são reprojetados pela Engenharia de Produto para atender a uma nova configuração, fazendo-se necessário projetar novas peças fora do conjunto de módulos padronizados.

Junto com as adaptações da carroceria para novas configurações de clientes, existem outras manutenções durante o ciclo de vida do produto, como, por exemplo, atualizações de *design*, novos opcionais de venda, mudanças nos processos de produção e alterações em normas reguladoras.

Toda essa massa de informação, durante o clico de vida do produto, acaba sendo perdida em anotações e arquivos pessoais ou até mesmo sendo algumas ações executadas para parte dos projetos, devido à falta de comunicação entre equipes e/ou uma metodologia robusta de captura do conhecimento, gerando atrasos em entrega de projetos, tempo de projeto desnecessário, retrabalho e erros em projeto.

Diante disso, a presente pesquisa descreve o desenvolvimento de uma aplicação para apoiar o processo de projeto de produto, desde a fase de configuração até a fase de criação de geometrias. A abordagem baseia-se na definição de *templates* de produto. Neste contexto, este trabalho visa combinar configuração de produto (conjunto de opcionais) com abordagens e técnicas de automação de projetos, a fim de apoiar as atividades de projeto de produtos para cumprir requisitos gerais e específicos.

Ainda é possível apoiar-se no estudo de Viero (2013), em que o autor cita, como possibilidades para trabalhos futuros, a necessidade de um processo de automação de projetos de produtos e um apoio a gestão de conhecimento.

1.3 Objetivos

A seguir serão apresentados o objetivo geral e os objetivos específicos da presente pesquisa.

1.3.1 Objetivo geral

O objetivo principal deste trabalho é desenvolver uma aplicação que permita a automação de projetos 3D.

1.3.2 Objetivos específicos

Em linhas especificas, pretende-se:

- Desenvolver a aplicação integrada ao *software* comercial Siemens NX;
- Utilizar o conceito de *template* de produto;
- Fornecer um ambiente informacional que sustente a captura de conhecimentos de engenharia sobre projeto de produto;
- Desenvolver um estudo de caso analisando a aplicabilidade da aplicação desenvolvida na automação de projetos finais de montagem e fabricação em uma empresa fabricante de ônibus.

1.4 Metodologia de pesquisa

Este estudo utiliza a metodologia de pesquisa *Design Science Research (DSR)*. Tal método é aplicado em Tecnologia da Informação (TI), sendo que a preocupação final é a criação de artefatos com aplicação nas organizações e na sociedade em geral. Projeto de *software* é, ao mesmo tempo, um processo iterativo e um produto resultante (artefato), não só resolvendo um problema humano, mas também o fazendo de uma maneira eficaz (HEVNER, A.; CHATTERJEE, S., 2010).

Tendo em vista os resultados que se pretende alcançar, a *Design Science Research* é a metodologia mais adequada ao processo de investigação proposto neste trabalho, sendo,

fundamentalmente, um paradigma orientado à resolução de problemas. Desta forma, a metodologia deverá considerar as seis fases a seguir:

Fase 1 – Identificação do problema e motivação. Este trabalho resultou da observação do problema na engenharia de produto da empresa encarroçadora de ônibus. Além dessa observação, baseou-se na seção de pesquisas futuras, sugeridas na dissertação de Viero (2013). A definição do problema resume-se na oportunidade gerada pela modularização de automatizar projetos na engenharia de produto. Nas encarroçadoras de ônibus, após o lançamento de um novo modelo de carroceria, os projetos de produto são reprojetados para atender a uma nova configuração, seja por adição ou remoção de algumas funcionalidades, fazendo-se necessário criar novas peças, fora do conjunto de módulos padronizados. Posteriormente, define-se a estrutura conceitual-teórica da presente pesquisa. Essa etapa constitui-se de uma revisão da literatura, consultando livros sobre os temas de interesse. Também estão contemplados artigos, dissertações e teses de bancos de dados nacionais e internacionais. Na pesquisa bibliográfica buscou-se identificar trabalhos sobre modularização, desenvolvimento de *software* e apresentação do projeto de carroceria de ônibus.

Fase 2 – Definição dos objetivos para a solução. Os objetivos podem ser encontrados na seção 1.3 desta dissertação.

Fase 3 – Projeto e desenvolvimento. Para o desenvolvimento/implementação do artefato, foram utilizadas as linguagens de programação C# (CSHARP) e JAVA. O banco de dados utilizado foi o MYSQL. O projeto e desenvolvimento do artefato são temas do quarto capítulo.

Fase 4 – Demonstração. Para demonstrar a eficácia do artefato foi utilizado o estudo de caso, apresentado no capítulo 5.

Fase 5 – Avaliação. Para observar e medir os resultados foi utilizado o método de estudo de caso, pois o artefato foi implementado em uma situação do mundo real corporativo, apresentada no capítulo 5.

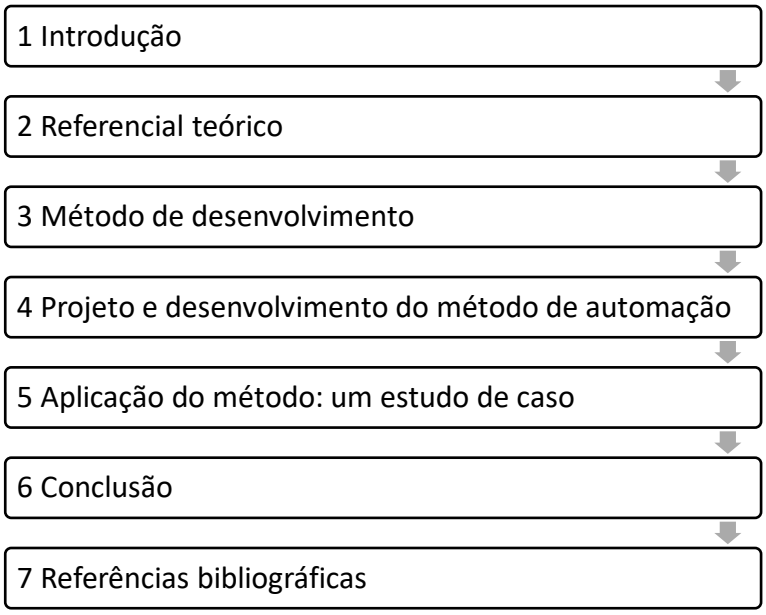
Fase 6 – Comunicação (Publicação). A pesquisa aplicada do presente trabalho tem como objetivo a produção de uma Dissertação de Mestrado Profissional. Este processo de comunicação se dará pela produção do presente trabalho e na apresentação e defesa do trabalho efetuado.

Nesta dissertação, classifica-se a natureza da pesquisa como aplicada, de abordagem qualitativa, com objetivo exploratório. O método utilizado para responder à questão de pesquisa foi *Design Science Research*.

1.5 Estrutura do trabalho

Este trabalho está organizado em seis capítulos, além as referências bibliográficas. Conforme ilustrado na Figura 2, os capítulos estão definidos como: introdução, referencial teórico, método de desenvolvimento, projeto e desenvolvimento do método de automação, aplicação do método: um estudo de caso, conclusão e referencial teórico.

Figura 2 - Estrutura do trabalho



Fonte: Autor, 2017

Inicialmente, o primeiro capítulo apresenta a introdução da pesquisa, a justificativa em torno do tema escolhido, objetivo geral e específico e metodologia definida para o trabalho.

Já o segundo capítulo aborda a revisão bibliográfica em torno dos temas: modularização, projeto de carroceria de ônibus, engenharia de *software* e trabalhos relacionados.

O terceiro capítulo, por sua vez, apresenta o método de desenvolvimento da aplicação proposta e a explanação sobre os diferentes tipos de projeto CAD e suas configurações.

O quarto capítulo apresenta a arquitetura geral do sistema e o projeto, bem como o desenvolvimento das aplicações Automator, CustomNX e Servidora de Regras.

Na sequência, o quinto capítulo aborda o estudo de caso com a aplicação do método proposto sobre dois diferentes tipos de projeto.

Por fim, o sexto capítulo é dedicado às conclusões e considerações finais acerca do trabalho desenvolvido, bem como sugestões para trabalhos futuros.

2 REFERENCIAL TEÓRICO

A seguir apresenta-se o referencial teórico sobre os principais conceitos referentes a modularização, projeto de carroceria e ônibus, *Product Lifecycle Management* (PLM), engenharia de *software* e trabalhos relacionados ao tema.

2.1 Modularização

Na última década, o conceito de modularização tem atraído a atenção de engenheiros, pesquisadores e estrategistas corporativos em muitas indústrias.

Quando um produto ou processo é "modular", os elementos de seu *design* são divididos e atribuídos aos módulos de acordo com uma arquitetura formal ou plano. Do ponto de vista de engenharia, a modularização geralmente tem três objetivos: tornar a complexidade administrável, possibilitar o trabalho paralelo e organizar as futuras incertezas (BALDWIN; CLARK, 2000; BALDWIN; CLARK, 2004).

Baldwin e Clark (2000) são os responsáveis pela forte introdução do conceito de modularização em projeto como retorno econômico para as empresas; por outro lado, esse trabalho tem sido largamente discutido nas áreas de sistemas de informação e *design* de produto, porém apenas de forma qualitativa (FUSARI; GAMBA, 2009).

As empresas que desejam aumentar a variedade de produtos ofertados a seus clientes, sem um impacto negativo substancial sobre seu desempenho operacional, podem utilizar a modularização. A ideia básica por trás da modularização é projetar, desenvolver e produzir peças que podem ser combinadas com várias outras, reforçando assim a compatibilidade entre variedade de produtos e desempenho operacional para os produtos discretos (SALVADOR; FORZA; RUNGTUSANATHAM, 2002).

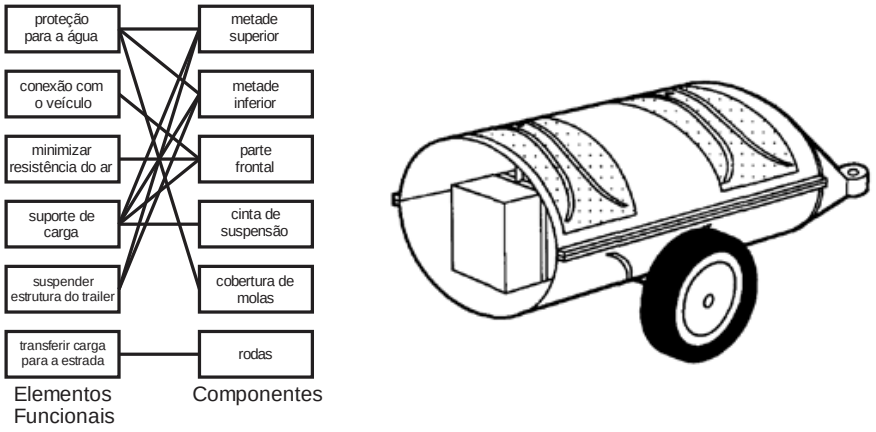
2.1.1 Arquitetura de produtos

Para Ulrich (1995), a escolha da arquitetura de produto tem implicações sobre o desempenho da manufatura na empresa. Quanto a sua arquitetura, um produto é dividido em: arquitetura modular e arquitetura integral. A arquitetura modular faz um relacionamento simples (um para um) entre os requisitos funcionais ao componente físico do produto, desassociando as interfaces dos

componentes. Já a arquitetura integral possui um mapeamento complexo (não um para um) entre função e componente físico, com interfaces acopladas ao componente.

A Figura 3 representa a arquitetura integral, com relacionamento complexo – do tipo muitos para muitos – entre os elementos funcionais e os componentes físicos.

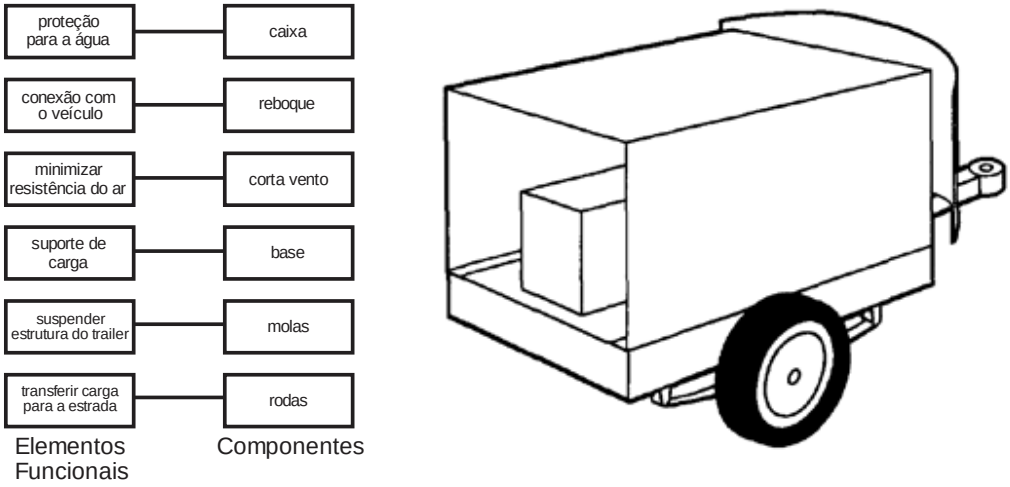
Figura 3 - Arquitetura Integral



Fonte: Adaptado de Ulrich, 1995

A Figura 4 representa a arquitetura modular, com relacionamento um pra um entre os elementos funcionais e os componentes.

Figura 4 - Arquitetura Modular



Fonte: Adaptado de Ulrich, 1995

Baldwin e Clark (2000) descrevem que para um sistema tornar-se modular é necessário definir sua arquitetura, isto é, o que seus módulos são e como os módulos irão interagir para desempenhar seu trabalho.

2.1.1.1 Módulo e interface

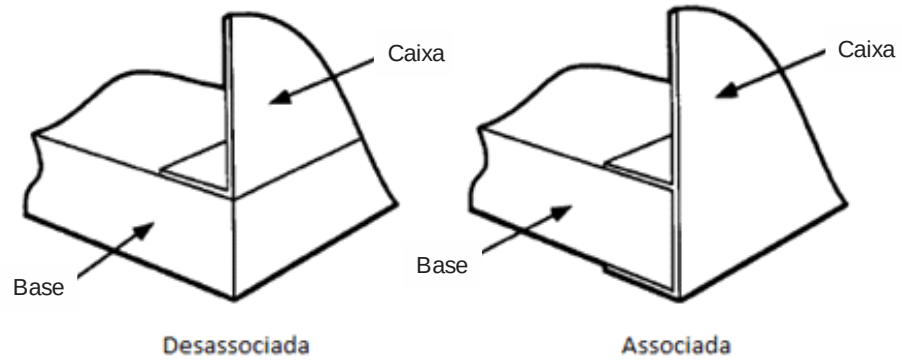
Segundo Baldwin e Clark (2004), módulos podem ser entendidos como unidades em um complexo sistema e estruturalmente independentes uns dos outros, que podem ser projetados e produzidos independentemente uns dos outros, mas que trabalham juntos. Já Fusari e Gabma (2009) descrevem módulo como sendo um grupo de parâmetros de projeto fortemente interconectados, sendo esses independentes dos parâmetros dos outros módulos existentes.

Quanto às interfaces, Baldwin e Clark (2004) descrevem que são como os módulos interagem entre si, como os módulos irão trabalhar em conjunto e como cada módulo executa o seu trabalho. Segundo Ulrich (1995), as interfaces podem ser físicas, havendo conexões geométricas entre dois componentes, ou não físicas, com ligações não tangíveis. As especificações de interface garantem a padronização de ligações entre os componentes.

Ulrich (1995) apresenta em sua metodologia dois tipos de interface entre os componentes: desassociadas (*de-coupling*) ou associadas (*coupling*).

A Figura 5 representa duas interfaces, uma desassociada e outra associada. Ao modo de exemplo, a interface associada exige que o projeto da caixa (B*ox*) seja alterado sempre que ocorrer uma mudança na espessura da cama (*Bed*), diferentemente do projeto desassociado.

Figura 5 - Interface desassociada e interface associada

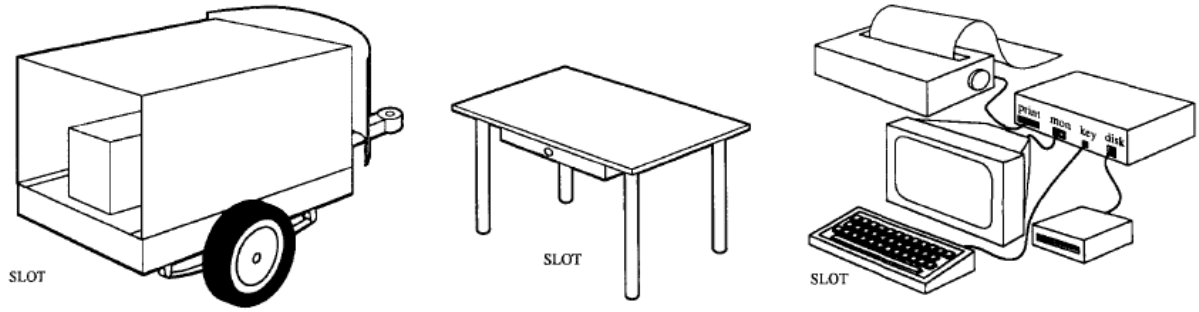


Fonte: Adaptado de Ulrich, 1995

Ulrich (1995) ainda cita em sua metodologia uma divisão no que diz respeito a interface, sendo: ranhura/entalhe/fenda (*slot*), barramento (*bus*) e de seção (*sectional*).

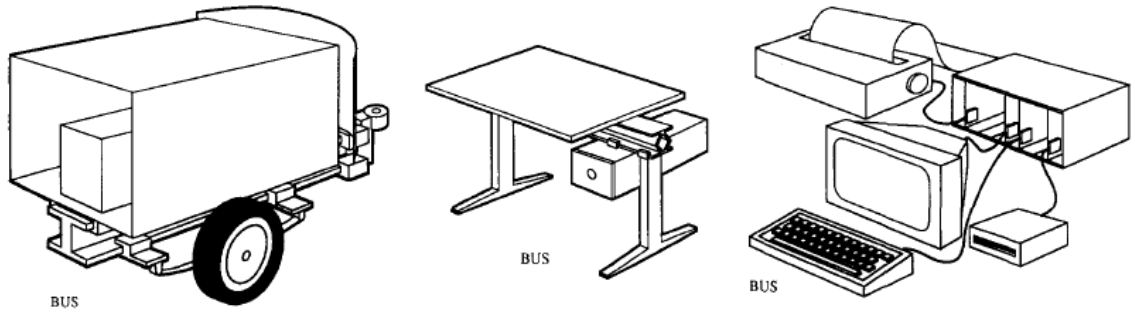
Na arquitetura modular de ranhura/entalhe/fenda (*slot*), cada uma das interfaces entre os componentes é de um tipo distinto, impossibilitando a intercambialidade entre os componentes. Na Figura 6 estão três exemplos de projetos com esse tipo de arquitetura.

Figura 6 - Arquitetura modular do tipo *Slot*



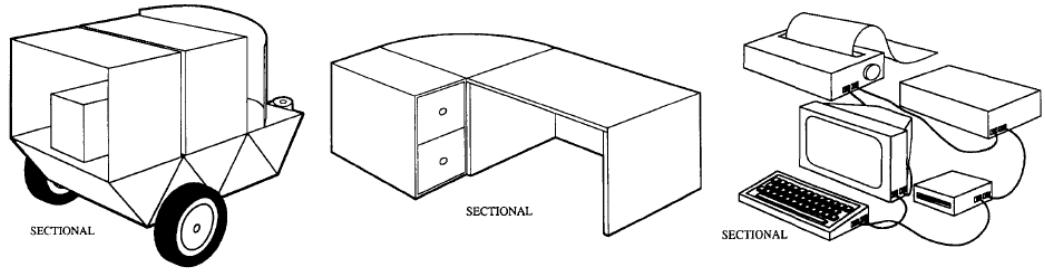
Fonte: Adaptado de Ulrich, 1995

Figura 7 - Arquitetura modular do tipo *Bus*



Fonte: Adaptado de Ulrich, 1995

Figura 8 - Arquitetura modular do tipo *Sectional*



Fonte: Ulrich, 1995

Em uma arquitetura modular de barramento (*bus*), existe um barramento padronizado para possibilitar o mesmo tipo de interface entre os componentes. A Figura 7 ilustra os projetos com esse tipo de arquitetura.

Já na arquitetura modular de seção (*sectional*), todas as interfaces são iguais, do mesmo tipo, de modo que todos os componentes podem ser associados ou anexados. Na Figura 8 estão representados três projetos utilizando esse tipo de arquitetura.

Para o autor supracitado, essa escolha entre as arquiteturas e interfaces é livre e ocorre dependendo da ótica pela qual o produto é observado, seja do nível da estrutura de montagem (*assembly)* ou do nível de peça.

2.1.2 Abordagens da modularização

Os seres humanos interagem com artefatos em três formas básicas: eles os projetam, os produzem e os usam. Assim, pode-se dizer que há três tipos de modularização: modularização de projeto, modularização de produção e modularização de uso. (BALDWIN; CLARK, 2000; BALDWIN; CLARK, 2004).

Modularização de uso apoia a personalização do sistema, fazendo com que o sistema atenda às necessidades e gostos dos usuários finais. Como exemplo, podem ser citados os consumidores de cama, colchão, lençóis, etc., vendidos por diferentes varejistas. Isso só acontece pois há um padrão de projeto normativo entre os fabricantes, possibilitando assim os componentes se encaixarem (BALDWIN; CLARK, 2004).

Na modularização de produção é possível a simplificação do processo de produção, dividindo complexas tarefas produtivas em pequenos processos. Os fabricantes de automóveis usam deste conceito para produzir componentes de um automóvel em departamentos distintos e voltam a reuni-los para uma montagem final. Esta estratégia só pode ser executada porque há uma especificação de projeto feita pela engenharia, informando como cada componente irá interagir com o veículo. Tal modularização é fundamental para a produção em massa (BALDWIN; CLARK, 2004).

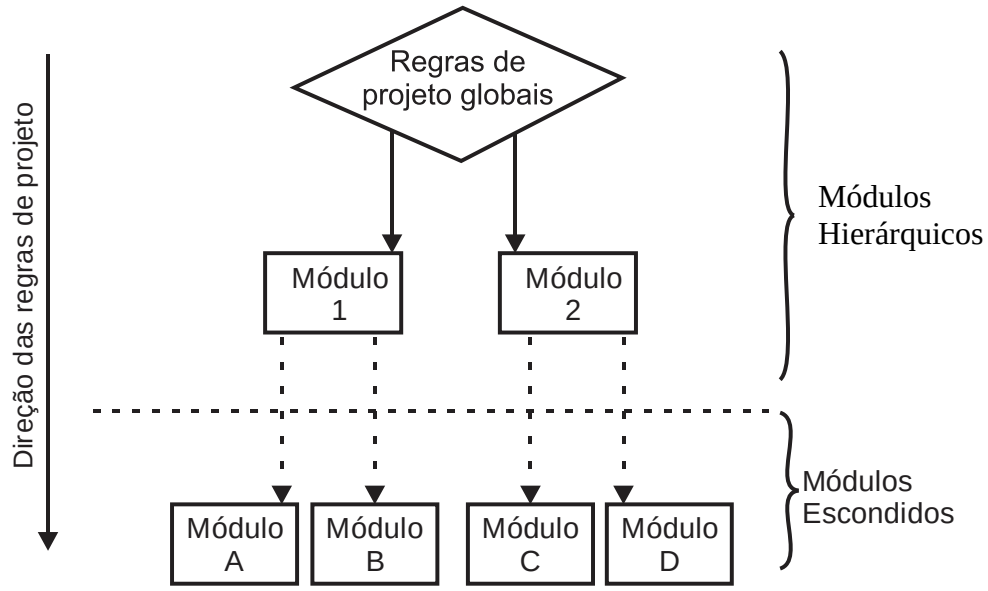
Segundo Baldwin e Clark (2004), um sistema de engenharia pode ser considerado modular em projeto se, e somente se, o processo de concepção possa ser distribuído entre módulos separados obedecendo a uma regra e interação de projeto. Os módulos devem permitir melhores respostas às necessidades dos clientes quando integrados aos requisitos de clientes na fase de concepção de produto (CUNHA, C.; AGARD, B.; KUSIAK, A., 2010). Nesse referencial teórico aprofunda-se ao conceito de modularização de projeto, visto que é o necessário para entendimento deste trabalho.

2.1.2.1 Modularização de projeto

Segundo Fusari e Gabma (2009), um projeto é uma descrição detalhada de um produto. Ele é completamente determinado por um número de parâmetros e suas interconexões. Os parâmetros são relacionados uns com os outros somente se há uma conexão, física ou lógica, ou uma dependência entre eles.

Em 1950, liderada por cientistas da computação, a modularização de projeto foi essencial para a evolução dos computadores até os dias de hoje. Devido à nova arquitetura modular, os computadores passaram a ser produzidos em larga escala, possibilitando empresas a projetarem sistemas modulares e componentes para computador, como dispositivos de armazenamento, memórias, e outras placas, aumentando o volume de negócios. (BALDWIN; CLARK, 2004).

Figura 9 - Projeto modular hierarquicamente representado



Fonte: Adaptado de Fusari e Gamba, 2009

Um projeto modular é um conjunto de módulos hierarquicamente agrupados a um específico projeto de regras, ou conjunto de regras, o qual cada módulo deve respeitar para manter a interface com os outros módulos e todo o projeto (FUSARI; GAMBA, 2009).

A Figura 9 representa a estrutura hierárquica dos módulos, sendo que, quanto maior for a posição hierárquica de um módulo na estrutura modular, mais implícitas são as restrições de projeto para os módulos inferiores conectados. Em outras palavras, as regras são ditadas pelos

módulos mais acima, e quanto mais abaixo o módulo estiver na estrutura, menos poder sobre o sistema ele vai ter.

O processo de modularização melhora a especialização no processo de desenvolvimento, porém implica em várias consequências para a gestão da operação e para a gestão de projeto. Quanto ao projeto, a modularização possibilita a independência funcional de cada módulo, dentro das regras gerais do projeto. Do ponto de vista operacional, o maior efeito da modularização é a descentralização, cada módulo poderá ser produzido por unidades distintas. Em outras palavras, cada módulo pode evoluir, dentro de determinadas regras de projeto, independentemente dos outros módulos, e cada projetista pode trabalhar no seu módulo sem preocupação de prejudicar todo o projeto. De um modo geral, a modularização nos permite gerenciar a complexidade, porque ela parte um sistema em um conjunto de elementos independentes de tamanho menor, e as regras de projeto amarram os módulos em uma estrutura hierárquica (SALVADOR; FORZA; RUNGTUSANATHAM, 2002; FUSARI; GAMBA, 2009).

2.2 Projeto de carroceria de ônibus

Tendo em vista o objetivo do presente trabalho, é importante conhecer o processo de desenvolvimento de produto, bem como os principais projetos que compõem o arranjo estrutural de uma carroceria de ônibus. Um dos projetos que irá compor o estudo de caso é o projeto da estrutura lateral, representado na Figura 14, da seção 2.2.2.2.

2.2.1 Projeto e desenvolvimento de uma carroceria de ônibus

Necessidades de clientes não são fáceis de serem obtidas. Essas necessidades podem ser obtidas a partir do histórico de vendas, tendo assim alguns padrões de compra e características de produto (CUNHA, C.; AGARD, B.; KUSIAK, A., 2010).

Segundo a pesquisa de Quevedo (2014), as informações – ou configuração de produto – que regem o projeto de uma carroceria estão contidas no pedido de vendas elaborado pelos clientes junto à área comercial. Nessa configuração de produto estão informações como: modelo da carroceria e chassi (conforme ilustrado na Figura 10), comprimento da carroceria, modelos de poltrona, ar-condicionado, janelas, banheiro, entre outros opcionais ofertados pela

empresa. Em seu estudo, mostrou-se como um grande fator de ineficiência de projeto, a conexão direta do projeto com características de venda e o alto número de opcionais ofertados pela empresa. Como o índice de aproveitamento de projetos é baixo, surge uma grande variedade de conjuntos que desempenham a mesma função global, aumentando assim o tempo de projeto e limitando os processos de produção da carroceria.

A linha de montagem de uma indústria de ônibus é muito semelhante a sistemas de montagem de automóveis, todas as carrocerias passam pelos mesmos postos de trabalho, equilibrando as cargas de trabalho para evitar tempo ocioso. Uma das grandes diferenças na linha de montagem de carrocerias é que nem todos os ônibus exigem as mesmas operações nos postos de trabalho, eles são altamente customizados, tornando diverso o *mix* de produção. As operações de montagem são manuais e complexas para serem automatizadas, fazendo os recursos se moverem ao longo da linha de montagem (PALENCIA, A. E. R.; DELGADILLO, G. E. M, 2012).

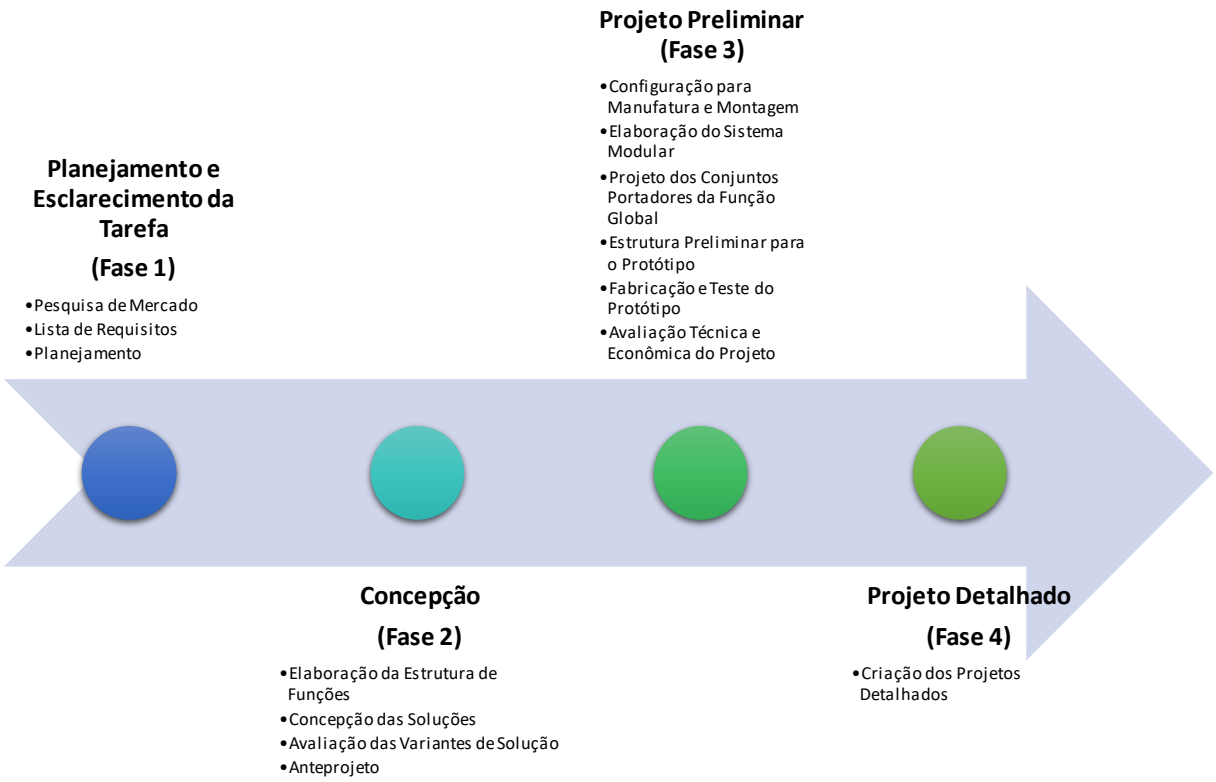
Figura 10 - Exemplo de chassi para ônibus

Fonte: SCANIA, 2016

O projeto de uma carroceria de ônibus, no setor de Engenharia de Produto, parte de um conceito pré-definido pelos setores de Engenharia de Desenvolvimento e *Design*, sendo categorizados em três níveis, em sua complexidade de configuração: Projeto Determinado, Projeto Customizado e Projeto Especial. No Projeto Determinado, usa-se um projeto de referência já existente (projeto já produzido pela empresa) podendo ter ou não pequenas alterações para adequar-se à configuração escolhida pelo cliente. Sua complexidade de projeto

é baixa. Os Projetos Customizados possuem uma complexidade média, exigindo do projetista maior conhecimento de produto. Este projeto também parte de um projeto anterior, de referência, porém com customização parcial. Já os Projetos Especiais possuem uma alta complexidade para sua determinação, exigindo do projetista, além do alto conhecimento de produto, conhecimentos em chassis, normas, manuais de encarroçamento e desenvolvimento de novos conceitos de produto (QUEVEDO; 2014, VIERO; 2013).

Figura 11 - Fluxo de desenvolvimento de projeto de uma nova carroceria



Fonte: Adaptado de Quevedo, 2014

Quevedo (2014) descreve que o fluxo de desenvolvimento de projeto de uma carroceria de ônibus, na engenharia de desenvolvimento, está definido em quatro fases: Planejamento e Esclarecimento da Tarefa, Concepção, Projeto Preliminar e Projeto Detalhado. Essas fases são posteriores à aprovação de conceito de *design* da carroceria. A Figura 11 retrata o fluxo de desenvolvimento com as subetapas.

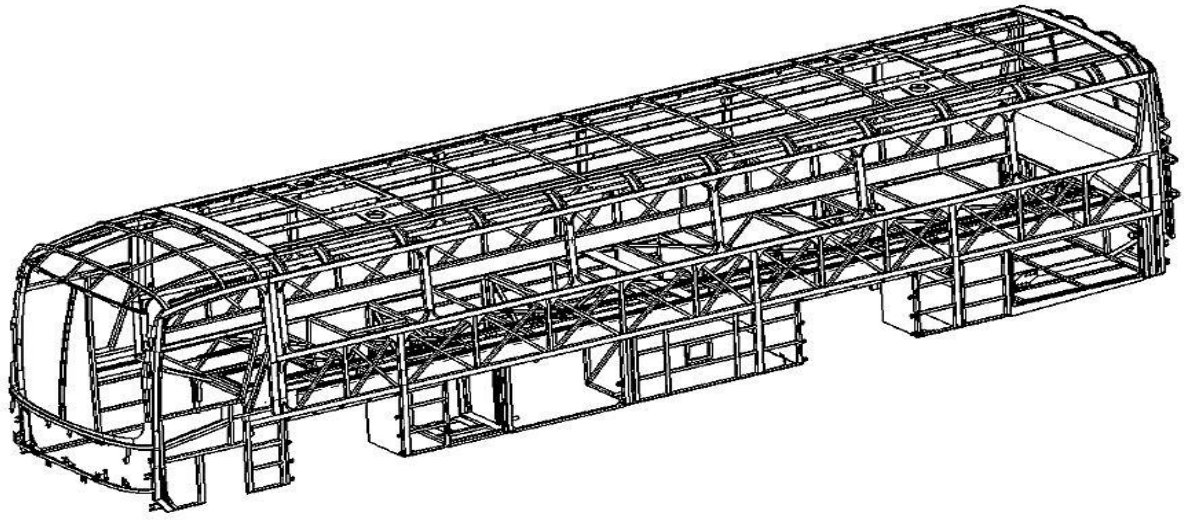
- As tarefas que compõem a Fase 1 (Planejamento e Esclarecimento da Tarefa) são definidas como: Pesquisa de Requisitos, Lista de Requisitos e Planejamento.

- As tarefas que compõem a Fase 2 (Concepção), são definidas como: Elaboração da Estrutura de Funções, Concepção das Soluções, Avaliação das Variantes de Solução e Anteprojeto.
- A Fase 3 (Projeto Preliminar) está subdividida em: Configuração para Manufatura e Montagem, Elaboração do Sistema Modular, Projeto dos Conjuntos Portadores da Função Global, Estrutura Preliminar para o Protótipo, Fabricação e Teste do Protótipo, Avaliação Técnica e Econômica do Projeto.
- Na Fase 4, encerra-se o desenvolvimento com a criação dos Projetos Detalhados para a Produção.

2.2.2 Arranjo estrutural de uma carroceria de ônibus

Um ônibus é formado pelo acoplamento de uma estrutura, denominada casulo, sobre um chassi. No Brasil, são chamadas montadoras as empresas que fornecem o chassi para as encarroçadoras, que são as responsáveis pela fabricação e acoplamento da carroceria sobre o chassi, formando assim o ônibus (CIAPPARINI, 2012; WALBER, 2009).

Figura 12 - Casulo de uma carroceria de ônibus



Fonte: VIERO, 2013

O arranjo estrutural de uma carroceria de ônibus, denominado casulo, é composto por sete componentes estruturais. Esses conjuntos são definidos como: frente, traseira, lateral

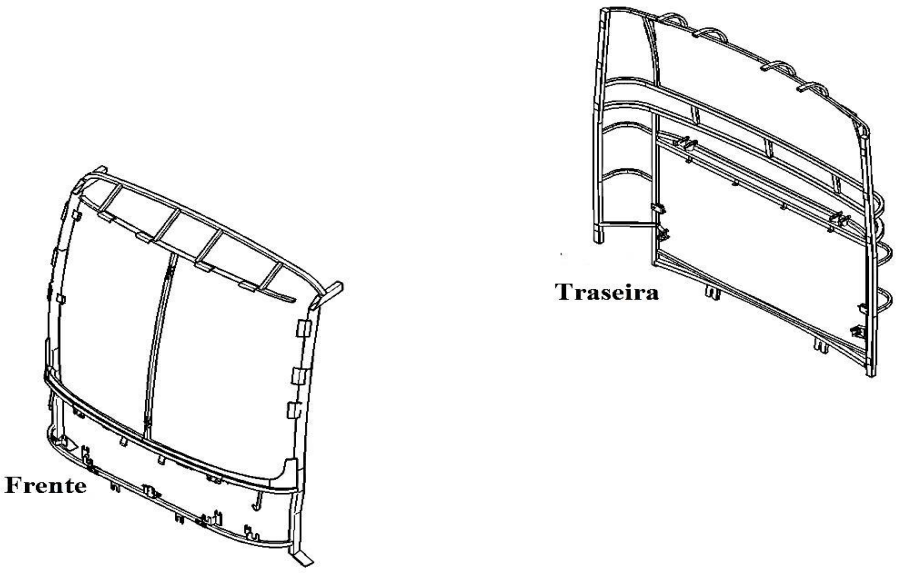
esquerda, lateral direita, base superior, base inferior, teto (QUEVEDO, 2014; VIERO, 2013; CIAPPARINI, 2012; WALBER, 2009).

Estes conjuntos, cujos principais materiais são uniões soldadas de tubos e chapas, são fabricados em gabaritos separados, e em seguida são transportados até um gabarito de montagem, onde são unidos pelo processo de soldagem. A principal função do casulo é dar forma e rigidez à carroceria do ônibus, suportando todas as solicitações e esforços (VIERO, 2013; CIAPPARINI, 2012; WALBER, 2009). A Figura 12 ilustra o projeto de um casulo antes de seu acoplamento ao chassi.

2.2.2.1 Estrutura da frente e traseira

Ambas as estruturas são compostas por tubos cortados e conformados, acompanhando o *design* da carroceria. Na estrutura dianteira são fixados componentes como: faróis, para-choque, mecanismos de articulação de tampa e limpadores, fibra frontal, entre outros. Já na estrutura traseira, são fixados: fibra interna e externa, acabamentos externos e internos, portinhola do motor, vigia e para-choque.

Figura 13 - Frente e Traseira de uma carroceria de ônibus



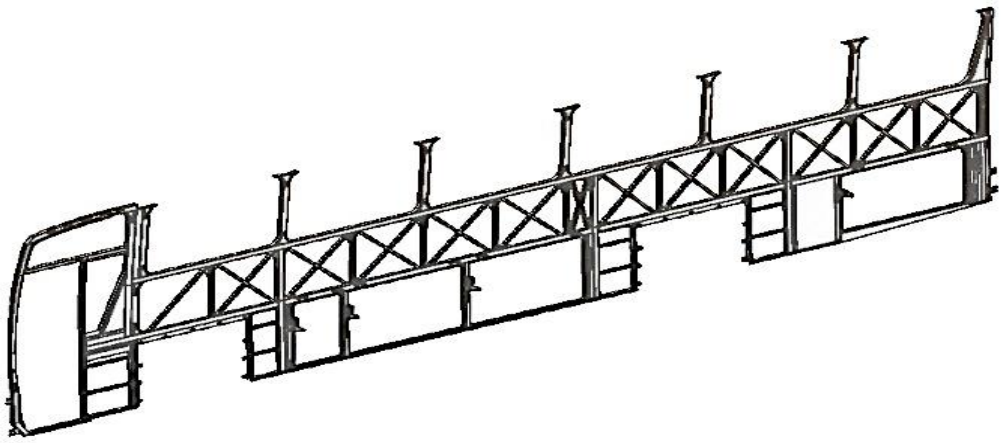
Fonte: VIERO, 2013

A Figura 13 exibe um projeto de estrutura dianteira e traseira (CIAPPARINI, 2012; VIERO, 2013; WALBER, 2009).

2.2.2.2 Estruturas laterais

A definição em relação aos lados, esquerdo e direito, dos componentes utilizados em projeto de carrocerias de ônibus é feita colocando-se uma pessoa sentada na poltrona do motorista, determinando, assim, ambo os lados (WALBER, 2009).

Figura 14 - Projeto de uma lateral



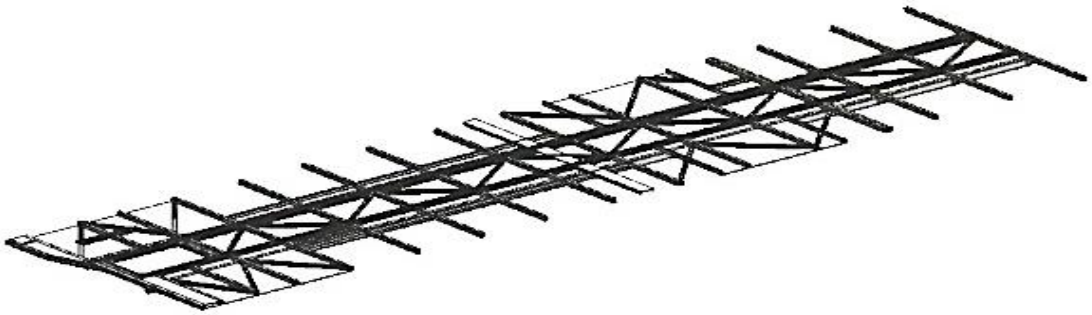
Fonte: VIERO, 2013

A estrutura lateral direita e a estrutura lateral esquerda são formadas por tubos conformados, conforme a seção transversal do veículo. As uniões são geralmente soldadas. Nas laterais são fixadas as janelas, portinholas dos bagageiros, chapeamentos externos, bem como toda a parte de acabamento interno e externo da carroceria. As estruturas laterais são sujeitas a cargas originadas pela ação do arranque, aceleração/frenagem, curva e torção. Os elementos que garantem a rigidez da estrutura lateral são denominados contraventos (componente x) (CIAPPARINI, 2012; VIERO, 2013; WALBER, 2009). A Figura 14 representa uma estrutura lateral.

2.2.2.3 Estrutura de base

A estrutura da base é formada por tubos retangulares e quadrados, geralmente unidos pelo processo de soldagem. Elemento de um arranjo estrutural, a base é essencialmente horizontal, fazendo a ligação da estrutura da lateral direita e da estrutura da lateral esquerda, servindo para a sustentação do assoalho e interface entre o chassi e a carroceria (VIERO, 2013; CIAPPARINI, 2012; WALBER, 2009).

Figura 15 - Projeto de base para uma carroceria de ônibus



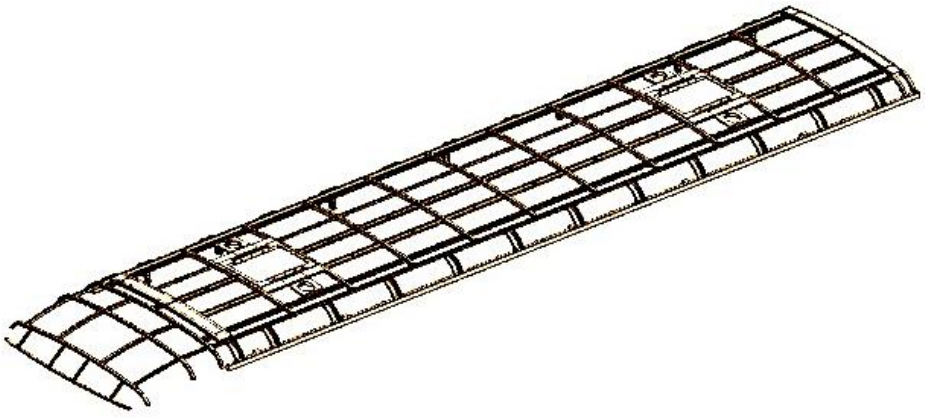
Fonte: VIERO, 2014

Pode ser visto um projeto de base para uma carroceria de ônibus na Figura 16.

2.2.2.4 Estrutura de teto

Fechando o casulo, tem-se a estrutura do teto composta por tubos, chapas e perfis, sua união é sobre as duas laterais, servindo para fixação de diversos componentes como: monitor de TV, porta-pacotes, acabamentos internos, fibra interna e externa, ar-condicionado, entre outros (VIERO, 2013; CIAPPARINI, 2012; WALBER, 2009). A Figura 16 ilustra o projeto do teto de uma carroceria de ônibus.

Figura 16 - Projeto de teto de uma carroceria de ônibus



Fonte: VIERO, 2013

2.3 *Product Lifecycle Management* (PLM)

O Gerenciamento de Clico de Vida do Produto (*Product Lifecycle Management* – PLM) é a atividade eficaz de gerir os produtos por todo seu ciclo de vida, iniciando com os primeiros

requisitos de desenvolvimento até ele ser retirado do mercado. O objetivo do PLM é aumentar as receitas de produtos, reduzir os custos relacionados com o produto, maximizar o valor do portfólio de produtos e maximizar o valor dos produtos atuais e futuros, tanto para clientes quanto para acionistas (STARK, 2015).

Conforme Siemens PLM *Software* (2016), define-se um sistema PLM como:

> O Gerenciamento de ciclo de vida do produto (PLM) é um sistema de gerenciamento de informações que pode integrar dados, processos, sistemas de negócio e, em último caso, pessoas em uma empresa estendida. O *software* PLM permite que você gerencie estas informações ao longo de todo o ciclo de vida inteiro de um produto, de maneira eficiente e econômica, desde a idealização, projeto e manufatura, através do serviço e disposição.

Segundo CIMdata (2016), PLM é uma abordagem estratégica de negócio que utiliza um conjunto de soluções para suportar a gestão colaborativa de informações de produto, abrangendo desde o conceito inicial até o fim da vida de um produto, integrando pessoas, processos, informações e sistemas.

Para Stark (2015), PLM é a estratégia para as empresas aumentarem seus lucros por meio da redução de tempo para lançamento de novos produtos e dando excelente suporte aos produtos correntes. Como os custos de produto estão diretamente relacionados à fase inicial do processo de desenvolvimento, o PLM fornece ferramentas e conhecimentos necessários para minimizar os custos do produto, além dos custos de assistências técnicas, garantias e reciclagem. O PLM fornece suporte ao uso de produtos pelos clientes. Com informações precisas sobre os produtos, o PLM permite que o valor do produto seja maximizado por todo seu ciclo de vida, além de estender o tempo de geração de receita. O PLM mantém o ciclo de vida do produto sob controle dos gerentes.

Para Rozenfeld (2007), existem duas classes de definição do PLM: PLM como uma ferramenta e PLM como uma estratégia. A primeira definição refere-se ao PLM como sendo um conjunto de *softwares* capaz de integrar todos os dados do produto durante seu ciclo de vida. Na segunda definição, o PLM, como uma estratégia, visa obter desempenho por meio da gestão integrada das informações de todo o ciclo de vida do produto.

2.3.1 Modelo de ciclo de vida de produto no PLM

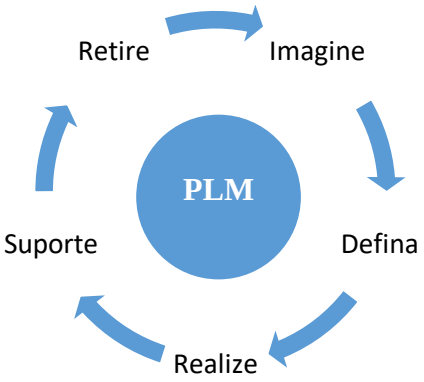
Grieves (2005) desenvolveu um modelo visual, descrito na Figura 17, em que é possível compreender o ciclo de informações do produto por diferentes disciplinas. Em torno do núcleo

de informação estão as principais áreas que compõem o clico de vida de um produto: planejamento, projeto, fabricação, suporte e descarte.

Figura 17 - Modelo de ciclo de vida de produto sugerido por Grieves

Fonte: Adaptado de Grieves, 2005

Figura 18 - Modelo de ciclo de vida de produto sugerido por Stark



Fonte: Adaptado de Stark, 2015

No Planejamento, fase inicial do desenvolvimento de qualquer produto, analisa-se e planeja-se os requisitos. Logo após, na fase de projeto, os requisitos são transformados em protótipos pela Engenharia de Desenvolvimento e, logo depois, o produto é detalhado e especificado pela Engenharia de Produto. Após o produto ser especificado, a Engenharia de Manufatura transforma a especificação do produto em especificação de fabricação (lista de processos e roteiros de fabricação) e ocorre a fabricação do produto. O departamento de Vendas

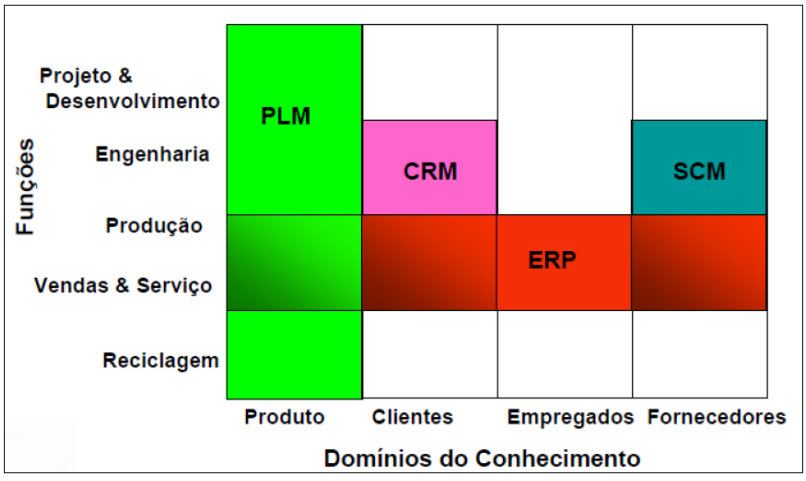
e Distribuição utiliza as informações do produto para informar aos clientes sobre seu uso e conformidade, além de eventuais suportes técnicos. Manutenção, Descarte e Reciclagem fecham o ciclo de vida do produto, sendo que, novamente, as informações de produto são necessárias para uma eficiente retirada do produto do mercado.

Stark (2015) sugere um ciclo de vida do produto "*from cradle to grave*" (do berço ao túmulo) em cinco fases: Imagine (*imagine*), Defina (*define*), Realize (*realise*), Suporte (*suport*) e Retire (*retire/dispose*), conforme exemplificado na Figura 18. Em cada uma das cinco fases, o produto está em um estágio diferente. Na fase de imaginação, o produto é somente uma ideia. Durante a fase de definição, as ideias são convertidas em descrição detalhada. No final da fase de realização, o produto existe na sua forma final (por exemplo, um automóvel). Na fase de suporte e retirada, o produto está em cliente e pode chegar a uma fase em que não é mais útil, então é aposentado pela empresa, e descartado pelo cliente.

2.3.2 Disciplinas formadoras do PLM

Para Grieves (2005), apesar do PLM ser um conceito do século 21, reúne elementos conceituais e tecnológicos já existentes. Os elementos são: *Computer-Aided Design* (CAD), *Engineering Data Management* (EDM), *Product Data Management* (PDM), *Computer Integrated Manufacturing* (CIM) e recentemente Grieves (2012) adicionou mais um elemento: *Systems Engineering* (SE).

Figura 19 - Matriz de domínio dos sistemas



Fonte: Adaptado de Grieves, 2005

Segundo Stark (2015), existem mais de cinquenta tipos de aplicações envolvendo os conceitos de PLM. Podendo destacar-se:

Tabela 2 - Elementos que ajudaram na origem do PLM

Acrônimo	Significado	Acrônimo	Significado
CAD	*Computer Aided Design*	EDM	*Engineering Data Management*
CAE	*Computer Aided Engineering*	FEA	*Finite Element Analysis*
CAID	*Computer Aided Industrial Design*	IPM	*Intellectual Property Management*
CAM	*Computer Aided Manufacturing*	KBS	*Knowledge Based Systems*
CAPE	*Computer Aided Production Engineering*	LCA	*Life Cycle Analysis*
CAPP	*Computer Aided Process Planning*	MRP	*Manufacturing Resource Planning*
CASE	*Computer Aided Software Engineering*	NC	*Numerical Control*
CIM	*Computer Integrated Manufacturing*	PDM	*Product Data Management*
DECM	*Digital Engineering Content Management*	PM	*Project Management*
EDI	*Electronic Data Interchange*	RP	*Rapid Prototyping*
ECM	*Enterprise Content Management*	SCM	*Software Configuration Management*
EDA	*Electronic Design Automation*	TDM	*Technical Document Management*
		VR	*Virtual Reality*

Fonte: Adaptado de Stark, 2015

Muitas vezes acontece a comparação entre PLM e o *software* empresarial ERP (*Enterprise Resource Planning*). Os conhecimentos mais comuns em uma empresa são de: produto, cliente, funcionários e fornecedores. Conforme representado na Figura 19, pode-se representar estes domínios em uma matriz, em que, na horizontal, estão as funções, na vertical, estão os domínios do conhecimento e, ao centro, as áreas de atuação do PLM, CRM (*Customer Relationship Management*), ERP (*Enterprise Resource Planning*) e SCM (*Supply Chain Management*). Claramente visível que o PLM coincide com o domínio do conhecimento sobre o produto e abrange todas as áreas funcionais da organização (GRIEVES, 2005).

2.3.3 Características do PLM

Segundo Grieves (2005), as características fundamentais do PLM devem ser as seguintes:

- Singularidade: Essa é a característica mais importante do PLM, pois a falta de singularidade dos dados é a maior fonte de desperdício de tempo, energia e material.

Decorrente da facilidade de alteração e multiplicação de dados. Sendo assim, as aplicações de *softwares* que implementem o PLM devem ter como característica fundamental a singularidade dos dados do produto, de modo que não haja questionamentos.

- Correspondência: Característica-chave do PLM, refere-se à ligação entre um objeto real e os seus dados e informações. Esses dados devem ser fidedignos ao objeto (tamanho, peso, cor e etc.), pois evita desperdício de tempo caso alguma peça tenha que ser modificada ou substituída.
- Coesão: Existem diferentes representações ou vistas de informação de um produto. Esse sempre terá uma vista geométrica, que reflete a sua estrutura, mas também pode ser representado de forma tridimensional, que é a maneira como o percebemos no espaço. Além dessas vistas, o produto pode ter vistas, como por exemplo, de elétrica e hidráulica. Para que o projeto seja coeso, deve existir uma ligação entre essas vistas, para que quando haja uma alteração no produto, todas as outras vistas sejam ajustadas automaticamente. Isso é coesão.
- Rastreabilidade: Rastreabilidade é a capacidade do *software* de armazenar as informações sobre o produto, enquanto ele está sendo desenvolvido, para que fique documentado o caminho percorrido até obter-se o produto final. Dessa maneira as informações sobre o produto ficam documentadas e com fácil acesso.
- Reflexividade: A Reflexividade é quando o estado de um produto é modificado no espaço físico e imediatamente documentado no espaço virtual. A vantagem da reflexividade no *software* vai ser percebida quando for necessário rastrear um produto "*as built*", economizando tempo e diminuindo a probabilidade de erros, pois as alterações ou melhorias feitas no produto físico foram efetuadas também no produto virtual.
- Disponibilidade: A disponibilidade no PLM é o inverso da reflexividade, ou seja, está relacionada à transição da informação do espaço real para o virtual.

2.3.4 Visão e estratégia

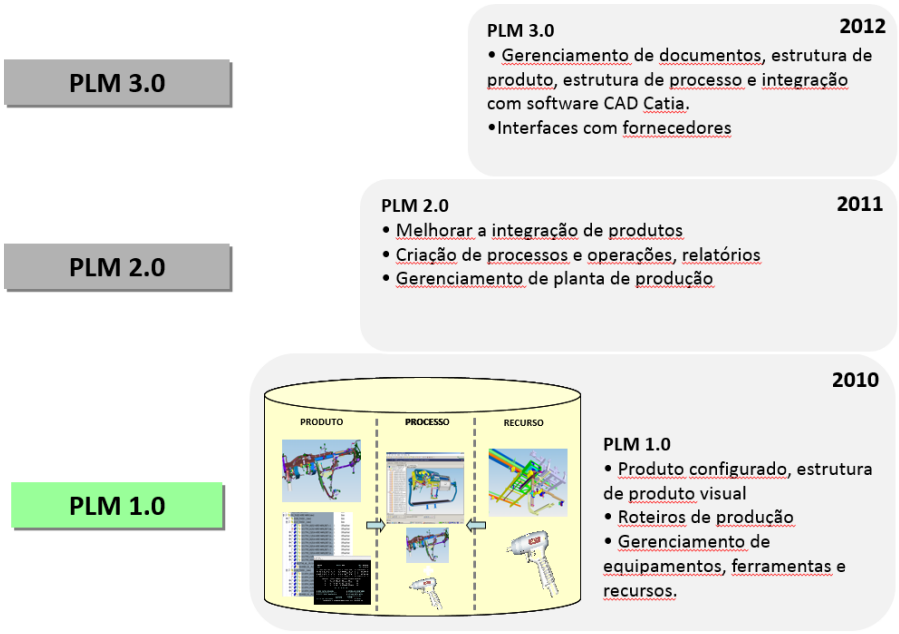
Stark (2011) comenta a necessidade de descrever uma visão futura de aproximadamente cinco anos, representando o ponto em que a empresa deseja chegar com a utilização do PLM.

Essa visão deve ser clara para que a empresa não perca o foco da implementação e as pessoas possam trabalhar para atingir os resultados, além de ser o ponto de partida para desenvolver uma estratégia para o PLM. A Figura 20 representa a visão do PLM na empresa Honda, em que é possível observar uma visão de três anos sobre o PLM.

Segundo Stark (2011), para alcançar a visão PLM, duas estratégias devem ser desenvolvidas: a estratégia PLM e a estratégia de implementação.

A Estratégia PLM mostra como os recursos da equipe PLM serão organizados no futuro. A estratégia descreve uma maneira de alcançar os objetivos utilizando escopo e recursos. As estratégias não são genéricas, são específicas para cada organização. A estratégia deve ser documentada e comunicada a todas as pessoas envolvidas. Os principais benefícios desta estratégia são: mostrar como atingir os objetivos do PLM, mostrar como os recursos serão utilizados e mostrar as políticas que regem o gerenciamento e uso dos recursos do PLM.

Figura 20 - Exemplo de visão do PLM na empresa Honda



Fonte: Adaptado de PLM World, 2010

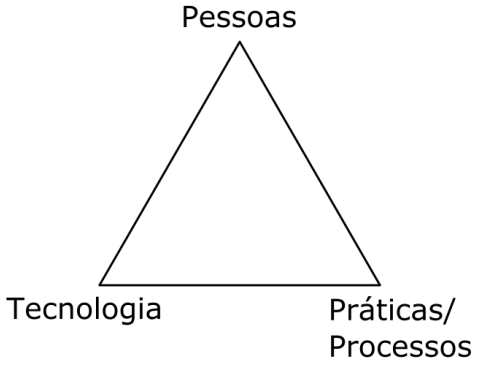
A estratégia de implementação mostra como os recursos serão organizados para tornar o ambiente atual no ambiente PLM do futuro. A estratégia de implementação é o ponto de partida para o desenvolvimento do plano de implementação, no qual a estratégia será detalhada com a utilização de recursos, geralmente por um período de um ano. Os principais benefícios desta estratégia são: maiores chances de alcançar a visão do PLM e a futura estratégia PLM,

assegura que os recursos serão utilizados ao máximo durante a implementação, comunica a todos sobre o andamento da implementação, garante com que todos trabalhem no escopo, permite maior assertividade em decisões.

2.3.5 Elementos do PLM

Segundo Grieves (2005), as pessoas, processos/práticas e tecnologia são partes fundamentais de uma perspectiva PLM. Esses três elementos formam um triângulo e estão representados na Figura 21, sendo que as pessoas estão no topo do triângulo, indicando seu papel dominante.

Figura 21 - Elementos do PLM



Fonte: Adaptado de Grieves, 2005

2.3.5.1 Pessoas

Para Grieves (2005), as pessoas impactam no sucesso ou fracasso do PLM dentro de uma organização, sendo pela capacidade limitada ou pela capacidade robusta. Indiferentemente do tipo de capacidade, elas são determinadas pelos seguintes aspectos: experiência, educação e treinamento e suporte.

- Experiência: a acumulação de informações e conhecimentos sobre diferentes situações. Com relação ao PLM, se pudermos capturar a experiência, temos a oportunidade de transmiti-la para outros indivíduos.
- Educação e treinamento: com os treinamentos as pessoas são ensinadas sobre o que fazer e com a educação ensina-se porque o fazem. A educação está ligada à prática enquanto o treinamento está ligado aos processos. O PLM pode servir

com informações para simular situações do mundo real, aumentando assim a produtividade.

- Suporte: sendo uma continuação da educação e treinamento, o suporte é necessário durante a execução das tarefas.

Grieves (2005) ainda cita outro fator importante referente às pessoas, o fator capacidade de mudança. A capacidade de mudança impacta no sucesso ou fracasso de um PLM.

2.3.5.2 Processos / Práticas

Os processos possuem um comportamento e uma definição muito maior do que a prática. Os processos podem ser representados, por exemplo, por fluxogramas e algoritmos dos quais espera-se uma saída determinada. Já a prática é uma caixa preta, em que entende-se a entrada, mas nem sempre pode-se esperar determinada saída. A eficiência esperada está diretamente ligada à definição e esclarecimento de processos (GRIEVES, 2005).

2.3.5.3 Tecnologia

Para Grieves (2005), o PLM é fortemente dependente de *softwares* e qualquer discussão nesta área é datada, pois estas tecnologias sofrem mudanças contínuas. Porém, existem questões que são independentes de qualquer versão de *software*. Essas questões dizem respeito ao ambiente que o PLM opera:

- PLM necessita de uma adequada infraestrutura de tecnologia;
- Aplicações de PLM precisam ser abertas e harmonizar-se com outras aplicações;
- Aplicações de PLM devem ser configuráveis e não customizáveis;
- Aplicações de PLM devem ser utilizáveis e incorporadas;
- Aplicações de PLM devem ser utilizadas de acordo com o seu desempenho.

2.4 Engenharia de *software*

Softwares são abstratos e intangíveis. São programas de computador e podem ser desenvolvidos para um cliente específico ou para o mercado em geral (SOMMERVILLE, 2011). O *software* compreende o conjunto de programas, procedimentos, dados e documentação. A engenharia de *software* é a disciplina cujo foco é o processo de produção do *software*, desde as etapas iniciais até a entrega e manutenção (PFLEEGER, S. L, 2004).

Projeto (*design*) pode ser definido como instruções, baseadas em conhecimentos, que transformam as “coisas” em valor. Disciplinas como arquitetura, engenharia, ciência da computação, engenharia de *software*, *design* e sistemas de informação tornam o projeto um elemento central, porém com diferentes pontos de vista. Projeto de engenharia é a transformação e avaliação sistemática de requisitos para artefatos. Projeto de *software* é, ao mesmo tempo, um processo iterativo e um produto resultante (artefato), não só resolvendo um problema humano, mas também fazendo de uma maneira eficaz (HEVNER, A.; CHATTERJEE, S., 2010).

No desenvolvimento de *software* existem quatro variáveis de controle: custo, tempo, qualidade e escopo. Quanto ao custo, muito dinheiro no início do projeto pode gerar mais problemas do que solução; por outro lado, um projeto com pouco dinheiro pode não resolver o problema do cliente. Sobre o tempo, maior prazo de entrega pode melhorar a qualidade do *software* e aumentar o escopo. Tempo reduzido diminui qualidade, escopo e aumenta os custos. Com respeito à qualidade, o fato de haver dificuldade de controlar uma variável não quantitativa, faz da qualidade a pior variável para controle, sendo que, geralmente, os custos são altos. Por fim, quanto menor o escopo, maior será a qualidade, além de permitir fazer maiores entregas em curtos prazos e com baixo custo (BECK, K.; FOWLER, M., 2000).

Os processos de desenvolvimento de *software* são complexos e dependem de pessoas para que as decisões sejam tomadas. Existem diferentes tipos de processos, mas não existe um processo ideal, levando as empresas a desenvolverem e adaptarem seus próprios processos (SOMMERVILLE, 2011).

As organizações multinacionais possuem diversos grupos de desenvolvimento de *software* espalhados por vários países ao redor do mundo, porém, trabalhar com equipes que estão geograficamente separadas é um desafio e deve ser controlado com utilização dos processos de desenvolvimento de *software* (PAPADOPOULOS, G.; 2015).

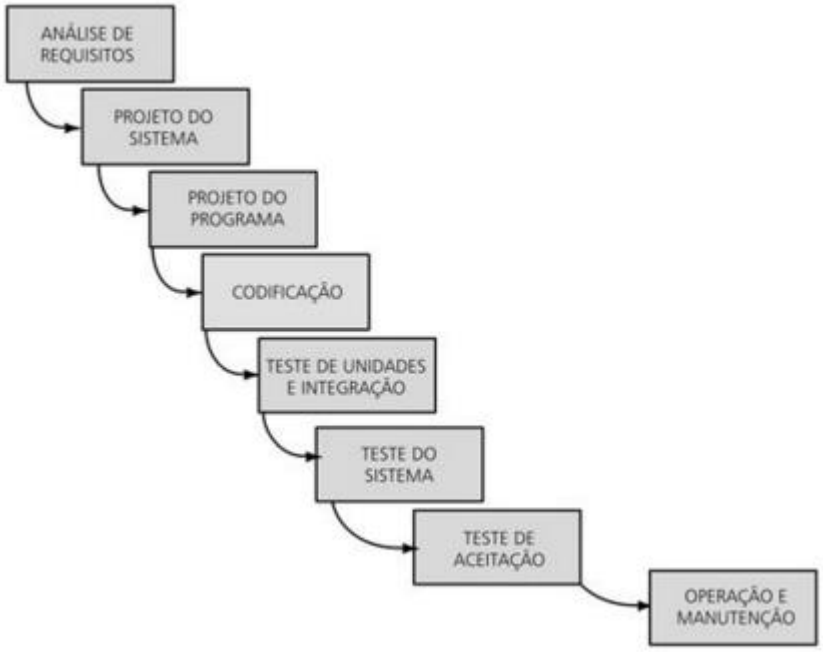
Os processos de *software* são categorizados como: dirigidos a planos ou processos ágeis. Os dirigidos a planos são conhecidos como tradicionais ou convencionais, quando todas as atividades são planejadas com antecedência e a validação final é feita com o plano inicial. Já os processos ágeis, possuem o desenvolvimento gradativo, sendo adaptativo às mudanças que surgem no decorrer do desenvolvimento (SOMMERVILLE, 2011).

O modelo dirigido a planos mais tradicional é o modelo em cascata. Este processo tem o princípio de planejar e programar todas as atividades do desenvolvimento antes de começar a

trabalhar nelas. Sua transição é de fácil adaptação, pois fornece uma estrutura clara para controlar as atividades durante todo o ciclo de desenvolvimento de *software*. (SOMMERVILLE, 2011; PAPADOPOULOS, G.; 2015). Este modelo foi utilizado por muitos anos como padrão de desenvolvimento dos *softwares* entregues ao Departamento de Defesa dos Estados Unidos (PFLEEGER, S. L, 2004).

Na Figura 22 é possível analisar o modelo em cascata, observa-se que o desenvolvimento de um estágio deve terminar antes de iniciar o próximo.

Figura 22 - Exemplo de modelo em cascata



(Fonte: PFLEEGER, S. L, 2004)

Este modelo é muito útil para representação dos fluxos de produção de *software* para clientes, devido à sua simplicidade; porém, vários problemas foram atribuídos a ele na literatura, como, por exemplo, a falta de conectividade entre as fases, não sendo possível interação entre determinados estágios. (PFLEEGER, S. L, 2004).

Outros modelos tradicionais são: modelo em V, prototipação, modelo em espiral, desenvolvimento incremental, orientado a reuso, RUP, entre outros (PFLEEGER, S. L, 2004; SOMMERVILLE, 2011).

2.4.1 Metodologias ágeis de desenvolvimento

Gomes (2013) descreve, em seu livro, que o Manifesto Ágil é a base para a criação de todos os métodos ágeis. Essas metodologias ágeis são compostas por vários métodos de desenvolvimento de *software*, a maioria deles com ciclos curtos de desenvolvimento. Esses ciclos são conhecidos como iteração. Uma iteração contém todas as etapas necessárias para desenvolvimento: planejamento, análise, *design*, codificação, testes e documentação. Nos ciclos de desenvolvimentos curtos, os *feedbacks* são frequentes e as respostas às alterações são rápidas.

Em seu trabalho, Papadopoulos (2015) apresentou um estudo sobre a migração de uma metodologia tradicional para metodologias ágeis em grandes e distribuídos projetos de *software*. Os resultados foram satisfatórios, aumentando a qualidade, permitindo mudanças de requisitos ao longo do desenvolvimento e satisfação dos desenvolvedores. Esses resultados originaram-se da comparação das atividades de duas equipes, uma com o uso tradicional de metodologias e a outra utilizando metodologias ágeis, no mesmo período de tempo.

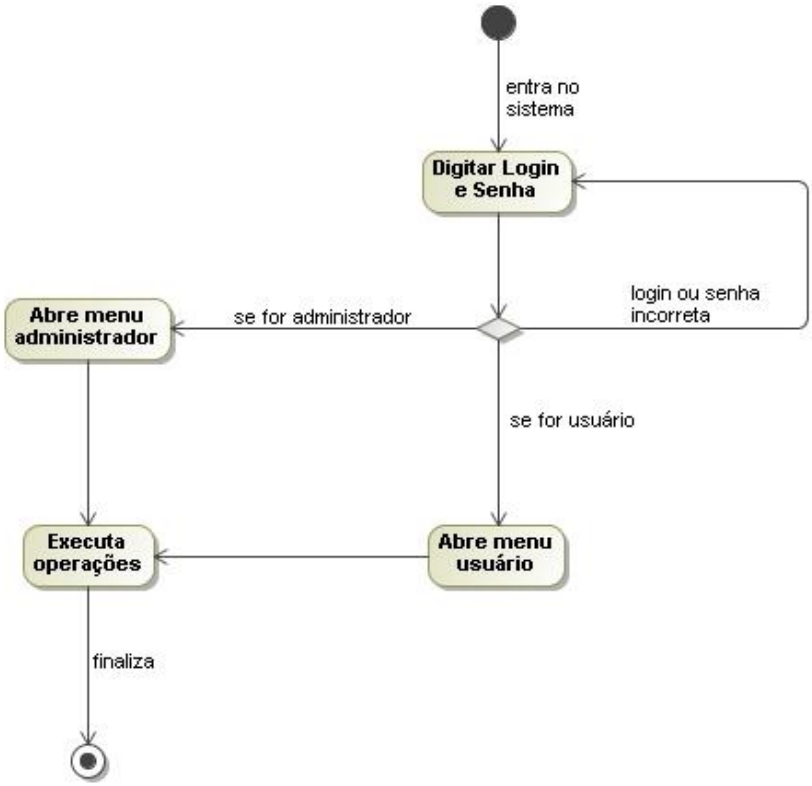
Tais princípios geraram práticas agregando maior valor aos clientes, criando equipes auto-organizadas, sustentando a criatividade e a produtividade, além de lidar com mudanças nos requisitos em qualquer fase do processo de desenvolvimento. Os clientes participam ativamente no processo de desenvolvimento, facilitando o *feedback*. Os princípios são diretrizes para a entrega de *software* de alta qualidade de forma ágil (DINGSØYR, T.; NERUS, S.; BALIJEPALLY, V.; MOE, N. N., 2012).

2.4.2 Unified Modeling Language (UML)

UML é o acrônimo de *Unified Modeling Language* (Linguagem de Modelagem Unificada), sendo uma linguagem de representação gráfica, baseada em diagramas, que permite representar e projetar sistemas de *software* padronizando a comunicação entre duas partes, abstraindo uma linguagem de computador. Esses diagramas são: atividade, caso de uso, classe, objetos, sequência, comunicação, estado, pacotes, componentes, implantação, interação, *timing*, *composite structure* (MEDEIROS, E. 2014). Para o presente trabalho, foram utilizados os diagramas de atividade, caso de uso e diagrama de classe, sendo detalhados a seguir.

O diagrama de atividade descreve o fluxo de informações resultante do processamento da ação do usuário. A Figura 23 exemplifica um diagrama de atividade, em que o principal objetivo é demonstrar visualmente o sistema de identificação de usuário, caso seja um administrador de sistema ou um usuário.

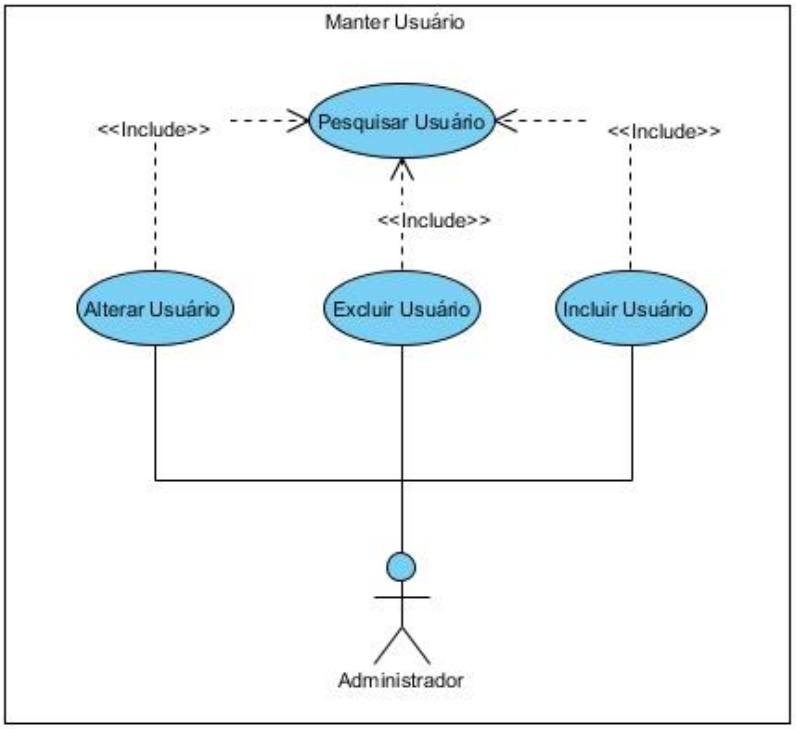
Figura 23 - Exemplo de diagrama de atividade



Fonte: Autor, 2017

O diagrama de casos de uso é utilizado para representar de forma modular o sistema e sua interação com os usuários, representado por atores. A Figura 24 exemplifica um diagrama de caso de uso para o módulo de gerenciamento de usuários, em que o administrador do sistema possui acesso para alterar, excluir e/ou criar um usuário.

Figura 24 - Exemplo de diagrama de caso de uso



Fonte: Autor, 2017

O diagrama de classes é uma representação da estrutura e relações das classes, que servem de modelo para objetos.

2.5 Trabalhos relacionados ao tema

No artigo desenvolvido por Aleksić, Janković e Stoimenov (2011) é apresentado um estudo de caso de modelagem de janelas e portas customizadas para uma empresa de customização em massa e com produção *one-of-a-kind* (OKP). No estudo de caso é utilizado conceito de modelagem orientada a objetos (OO), metodologia muito comum para concepção de *software*, para representar as classes e atributos do produto. O objetivo da pesquisa foi desenvolver um *framework* de modelagem de família de produto utilizando a metodologia MDA (Arquitetura Dirigida pelo Modelo). O resultado do estudo de caso foi positivo, com altos níveis de personalização de produtos e encurtamento do prazo de execução. O *software* desenvolvido para customização dos produtos não tem integração com sistemas CAD, porém gera saídas do tipo STEP (*Standard for The Exchange of Product Data*), que podem ser importados em sistemas CAD, sendo gerada, também, a estrutura de produto (BOM).

Athanasopoulos, Ugail e Castro (2009) apresentam em seu estudo uma ferramenta de geração de superfícies genéricas para projeto e construção de aeronaves. As complexas geometrias podem ser criadas e modificadas pelo usuário em tempo de execução. As parametrizações das superfícies utilizam o método PDE (*Partial Differential Equations*). Os objetivos da pesquisa foram discutir a utilização do método PDE na modelagem de aviões, além de demostrar que a técnica pode gerar um modelo de avião sem qualquer conhecimento prévio pelo usuário. Os resultados foram satisfatórios.

O estudo de Ren, Qiu, Zhang, Tan e Cheng (2013) é referente à criação de um método para construir uma ligação entre as demandas dos clientes e estrutura de produto, agilizando assim as respostas aos clientes. A geração da estrutura de produto utiliza o método *fuzzy* de processo de análise hierárquica (AHP) para a tomada de decisão e tem como entrada as demandas dos clientes, normas técnicas e o modelo de estrutura do produto a ser configurado.

Frutos (2006) apresenta, em sua Tese de Doutorado, um modelo de customização de produtos facilitando o projeto e interação com o cliente na seleção e configuração de um produto. O modelo proposto utiliza conceitos de modelagem orientada a objetos, análise de decisão e programação linear. Um sistema foi desenvolvido, automatizando o método, e foi aplicado em forma de estudo de caso na construção civil. O sistema não é compatível com *softwares* CAD.

No artigo de Mavridou, Kehagias, Tzovaras e Hassapis (2013) é apresentado um *framework* para indústria automotiva. O *software* desenvolvido interage com os usuários, a fim de recomendar uma configuração de produto inicial com base em suas necessidades pessoais. O *software* desenvolvido utiliza metodologias ágeis de desenvolvimento e não possui integrações com outros *softwares* CAD.

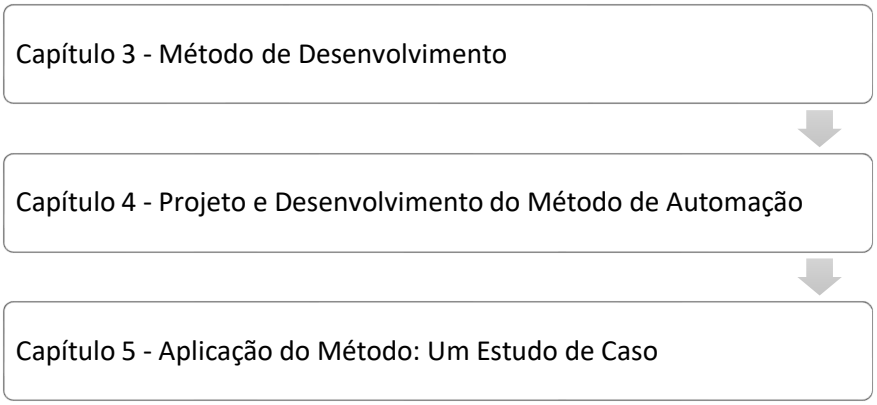
Myung e Han (2001) apresentaram em seu artigo um *framework* de modelagem paramétrica, em nível de peças e conjuntos, de uma máquina de usinagem utilizando base de conhecimento (*knowledge-base*). O *software* especialista desenvolvido tem interface com sistemas CAD através de API (Interface de Programação de Aplicações), além de uma interface de configuração de projeto para máquinas de usinagem. Seus resultados foram satisfatórios, reduzindo tempos em modificações de projetos.

A pesquisa, publicada por Raffaeli, Mengoni e Germani (2013), tem como foco ferramentas e métodos para produção automática de modelos geométricos, combinando configuração de produto com técnicas de automação de projeto, apoiando as atividades de

projeto de produto. Utiliza-se uma abordagem chamada de protótipos virtuais configuráveis (CVP), ligados a três domínios: especificação do produto, dados geométricos e conhecimento do produto. Um estudo de caso, com desenvolvimento do *software*, foi desenvolvido para representar o método proposto.

Como a utilização e entendimento das aplicações é extensa e complexa, optou-se por dividir o estudo de caso em três capítulos: método de desenvolvimento, projeto e desenvolvimento do método de automação e aplicação do método, conforme representado na Figura 25.

Figura 25 - Fluxo do estudo de caso



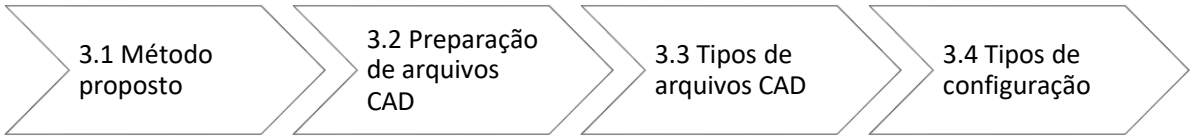
Fonte: Autor, 2017

O método de desenvolvimento descreve os conceitos por trás dos *templates* de produto, sendo essenciais para o entendimento do restante do trabalho. O projeto e desenvolvimento do método de automação descreve detalhadamente o projeto das aplicações propostas. Por fim, o Capítulo 5 – Aplicação do Método: um Estudo de Caso – detalha como foi solucionado o problema proposto, utilizando como estudo de caso o projeto de uma lateral esquerda e um projeto de distribuição de janelas.

3 MÉTODO DE DESENVOLVIMENTO

Na empresa estudada, o projeto de carroceria de ônibus é categorizado em três níveis: Projeto Determinado, Projeto Customizado e Projeto Especial. O método proposto permeia as três categorias, porém sua eficiência é maior nos projetos customizados e projetos especiais, nos quais o nível de complexidade é maior. Sendo assim, o capítulo está dividido em quatro sessões: método proposto, preparação de arquivos CAD, tipos de arquivos CAD e tipos de configuração, conforme ilustra a Figura 26.

Figura 26 - Sessões do Capítulo 3



Fonte: Autor, 2017

A seção método proposto introduz a metodologia de *templates* de produto para solucionar o problema proposto, unificando cinco áreas: conhecimento de produto, geometrias, estrutura de produto, reuso de projeto e configuração de venda. Para gerir essas áreas introduz-se a arquitetura das três aplicações desenvolvidas: Automator, CustomNX e Aplicação Servidora de Regras.

A seção preparação de arquivos CAD introduz os diferentes tipos de arquivos CAD (projetos) e configurações necessárias para as duas últimas sessões. Nesta seção é apresentada uma matriz que identifica quais configurações são necessárias para cada tipo de arquivo CAD. Os arquivos CAD foram divididos em Família de Peças, Família de Conjuntos, Itens Não Família e *Templates* e possuem configurações dos tipos *Expressions, Part Family* e *Assembly*.

Conforme apresentado no APÊNDICE A, as *expressions* são entidades presentes no *software* CAD NX que armazenam valores e/ou fórmulas que podem definir o comportamento dos objetos.

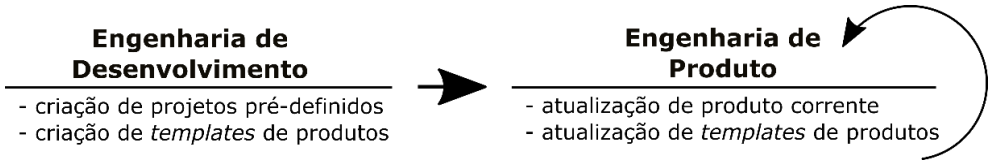
A seção tipos de arquivos CAD detalha cada um dos quatro tipos de arquivos CAD. Por fim, a seção de tipos de configurações detalha os três tipos de configurações de projeto.

Posteriormente, no Capítulo 4 é detalhado o projeto e desenvolvimento das três aplicações propostas neste método.

3.1 Método proposto

Como a Engenharia de Produto parte de um conceito predefinido pelos setores de Engenharia de Desenvolvimento e *Design*, este método também inicia-se na Engenharia de Desenvolvimento, exemplificando, a Engenharia de Produto receberá, junto com o projeto da Lateral Esquerda, o *template* da Lateral Esquerda. A Figura 27 exemplifica este fluxo, a Engenharia de Produto receberá os *templates* predefinidos pela Engenharia de Desenvolvimento, fazendo com que a Engenharia de Produto de continuidade no desenvolvimento dos *templates*, atualizando-os para cada nova melhoria de projeto ou mercado.

Figura 27 - Fluxo de manutenção dos *templates*



Fonte: Autor, 2017

Dessa forma o método proposto está baseado em *template* de produto. Os *templates* são elementos tridimensionais, desenvolvidos no *software* CAD NX, em que o conhecimento implícito e explicito é armazenado.

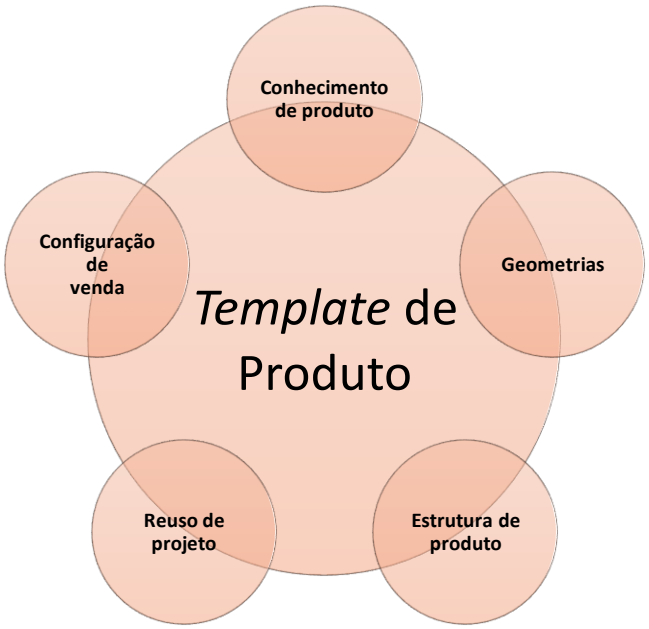
Eles podem ser definidos como blocos inteligentes que encapsulam regras de configuração e a lógica de criação, absorvendo as diversas características e variações escolhidas pelo cliente (RAFFAELI, R.; MENGONI, M.; GERMANI, M., 2013).

O método proposto unifica cinco áreas: conhecimento de produto, geometrias, estrutura de produto, reuso de projeto e configuração de venda, conforme representado na Figura 28.

- Conhecimento de produto: absorção do conhecimento dos engenheiros e projetistas nas fases de concepção e manutenção de produtos, tornando o conhecimento de construção do ônibus um ativo da empresa.
- Geometrias: o método tem acesso a toda a base de projetos da engenharia, podendo fazer escolhas de utilização de projetos baseada em regras definidas pelo engenheiro. Caso essa procura não satisfaça a regra, o método tem a capacidade de criar o projeto inexistente.
- Estrutura de produto: após a conclusão do *template*, o usuário tem disponível a estrutura de produto completa, conforme projetada por regras pelo engenheiro.

- Reuso de projeto: os *templates* são baseados em regras, ou seja, o método nunca irá utilizar um projeto diferente do que foi estipulado na regra, ou, criar um projeto novo, duplicando projetos no PLM.
- Configuração de venda: o *template* é totalmente baseado no principal agente externo de um projeto de ônibus, as configurações de carrocerias feitas pelos clientes.

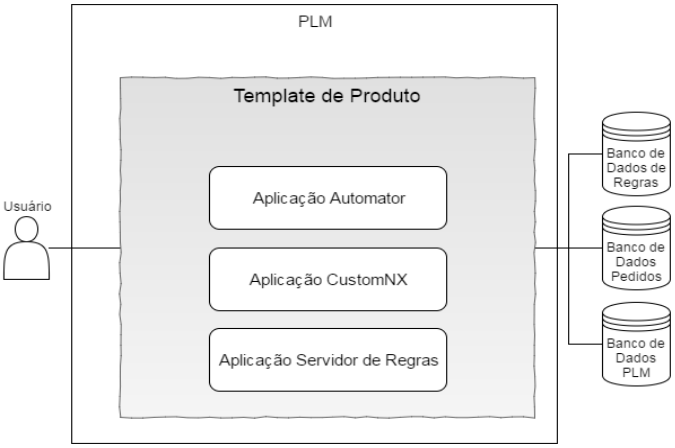
Figura 28 - Método proposto



Fonte: Autor, 2017

Para gerir essas cinco áreas foram desenvolvidas três aplicações: Automator, CustomNX e Aplicação Servidora de Regras, conforme ilustrado na Figura 29. A aplicação Automator é responsável pela gestão de regras de cada *template*, ou seja, é a aplicação em que o usuário irá criar, alterar e deletar regras para cada *template* desenvolvido, retendo conhecimento de produto e efetuando reuso de projeto. A Aplicação Servidora de Regras é responsável por armazenar essas regras em banco de dados e efetuar os cálculos quando solicitado. Por fim, a aplicação CustomNX faz o elo de ligação entre a configuração de venda e as regras desenvolvidas (aplicação servidora de regras) para cada *template*, tendo como resultado a estrutura de produto. A aplicação CustomNX é responsável também pela criação de projetos, quando necessário.

Figura 29 - Visão geral da arquitetura



Fonte: Autor, 2017

No Capítulo 4, a arquitetura do sistema e as aplicações terão seu desenvolvimento detalhado.

3.2 Preparação dos arquivos CAD

Para que os arquivos CAD sejam utilizados neste método de desenvolvimento, faz-se necessário um *setup* rigoroso de configurações.

As configurações estão divididas em três categorias: ***Expressions***, ***Part Family*** e ***Assembly***, conforme ilustrado na Figura 30. Configuração do tipo *Expressions* são conjuntos de expressões obrigatórias para que o arquivo tenha integração com a aplicação desenvolvida. Já a configuração de *Part Family* é somente de disposição das colunas do Excel. Por fim, a configuração do tipo *Assembly* refere-se à nomenclatura interna que o item possui na estrutura.

Figura 30 - Separação de configurações de arquivos CAD

Expressions	*Part Family*	*Assembly*
•conjuntos de expressões obrigatórias.	•configuração das colunas do Excel.	•configuração de nomenclatura interna do item na estrutura.

Fonte: Autor, 2017

Para facilitar o entendimento de cada configuração, dividiram-se os arquivos CAD em quatro categorias: **Família de Peças**, **Família de Conjuntos**, **Itens Não Família** e ***Templates***.

A Tabela 3 relaciona os tipos de configurações e os tipos de arquivos CAD, sendo que o hífen (-) indica que **não há configuração** e o caractere (X) indica a **obrigatoriedade de configuração**. É possível identificar que:

- Família de peça necessita somente de configuração do tipo *Part Family;*
- Família de conjunto necessita dos três tipos de configuração;
- Itens Não Família não necessita de nenhuma configuração; e
- *Template* necessita somente da configuração do tipo *Expressions*.

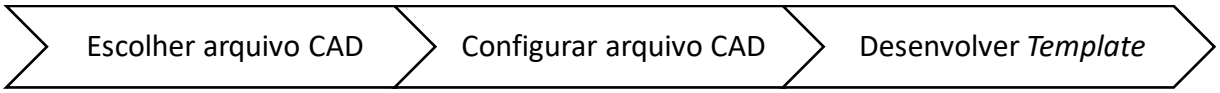
Tabela 3 - Configurações necessárias em cada tipo de arquivo CAD

		Configuração		
		Expressions	***Part Family***	***Assembly***
Arquivo CAD	Família de Peças	**-**	**X**	**-**
	Família de Conjuntos	**X**	**X**	**X**
	Itens Não Família	**-**	**-**	**-**
	Template	**X**	**-**	**-**

Fonte: Autor, 2017

Desta forma, o fluxo lógico de configuração do *template* segue conforme ilustrado na Figura 31. Primeiro, define-se o tipo de arquivo CAD; em seguida aplicam-se as configurações dependendo do tipo de arquivo CAD escolhido; e, por fim, desenvolve-se o *template*.

Figura 31 - Fluxo lógico de configuração de *template*



Fonte: Autor, 2017

As próximas duas sessões serão dedicadas a detalhar cada tipo de arquivo CAD e cada tipo de configuração.

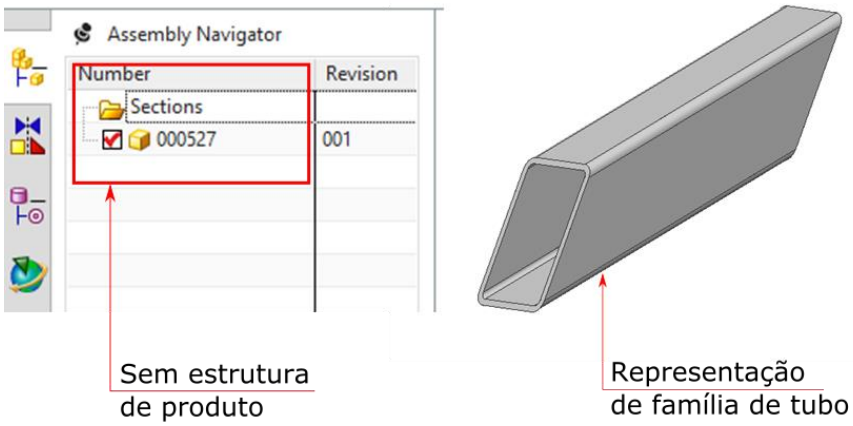
3.3 Tipos de arquivos CAD

Arquivos CAD são arquivos que contêm informações de projetos, neste trabalho associados ao *software* Siemens NX. Esta seção exemplifica os quatro tipos de projetos: **Família de Peças**, **Família de Conjuntos**, **Itens Não Família** e ***Templates***.

3.3.1 Família de peças

Famílias de peças caracterizam-se por não possuírem estrutura de produto e possuir comportamento paramétrico. A Figura 32 ilustra uma família de tubo retangular projetada no *software* NX.

Figura 32 - Exemplo de Família de Peça

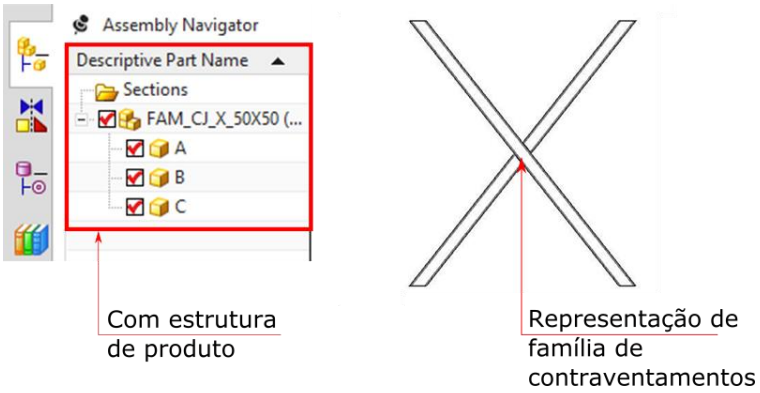


Fonte: Autor, 2017

3.3.2 Família de conjuntos

Famílias de conjuntos caracterizam-se por possuírem estrutura de produto e possuir comportamento paramétrico. A Figura 33 exemplifica uma família de contraventamentos, projetada no *software* NX, com os componentes “A”, “B” e “C” em sua estrutura.

Figura 33 - Exemplo de família de conjuntos



Fonte: Autor, 2017

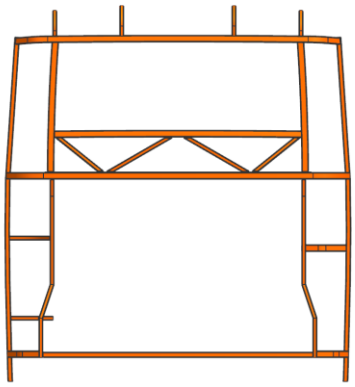
Os componentes “A”, “B” e “C”, que foram citados acima, podem ser do tipo de arquivo CAD: família de peças, família de conjuntos ou itens não família. Cada um deles possui uma configuração distinta e estão sendo explicados neste capítulo.

Como foi possível identificar, o tipo de arquivo Família de Conjunto possui os três tipos de configurações e necessita, exclusivamente, da criação de um projeto na aplicação Automator. Este procedimento é descrito na seção 5.5.7.

3.3.3 Itens não família

Itens não família são todas as peças/conjuntos que não fazem parte do grupo de família de peça e família de conjunto. A estrutura traseira, representada na Figura 34, é um exemplo deste tipo de arquivo.

Figura 34 - Exemplo de item não família



Fonte: Autor, 2017

3.3.4 Templates

Por fim, os *Templates* são os arquivos responsáveis pela criação das fórmulas ou matemáticas, definindo o comportamento parcial/total de peças e conjuntos conforme um grupo de regras. Esses arquivos não possuem modelagem.

Este tipo de arquivo é o principal objeto de estudo desta dissertação e será detalhado no Capítulo 5.

3.4 Tipos de configurações

A seguir são detalhados os quatro tipos de configurações.

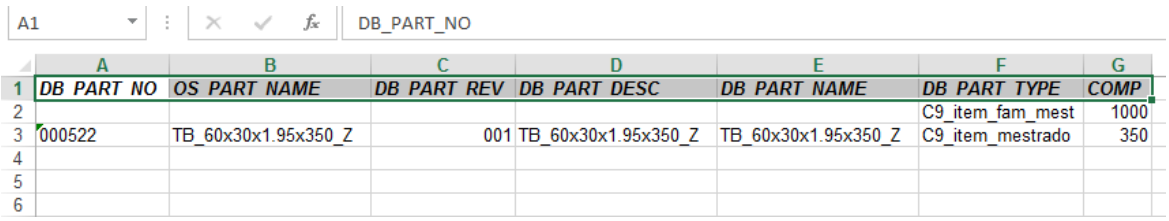
3.4.1 Configuração do tipo part family

Quanto à configuração do tipo *Part Family*, conforme linha 1 da Figura 35, é obrigatório que as colunas do Excel estejam na seguinte ordem:

- DB_PART_NO
- OS_PART_NAME
- DB_PART_REV
- DB_PART_DESC
- DB_PART_NAME
- DB_PART_TYPE

Quanto ao restante das colunas, exclusivas a cada projeto, não se faz necessária uma ordem especial. As medidas informadas são em milímetros.

Figura 35 - Ordem das colunas do *Part Family*



	A	B	C	D	E	F	G
1	DB_PART_NO	OS_PART_NAME	DB_PART_REV	DB_PART_DESC	DB_PART_NAME	DB_PART_TYPE	COMP
2						C9_item_fam_mest	1000
3	000522	TB_60x30x1.95x350_Z	001	TB_60x30x1.95x350_Z	TB_60x30x1.95x350_Z	C9_item_mestrado	350
4							
5							
6							

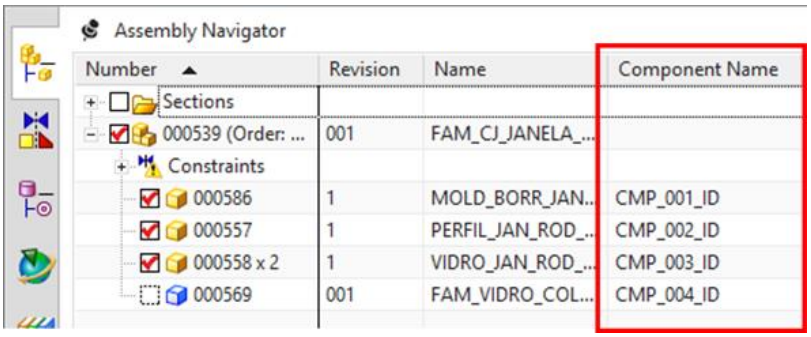
Fonte: Autor, 2017

3.4.2 Configuração do tipo Assembly

A configuração do tipo *Assembly* consiste em renomear o componente dentro da montagem.

A estrutura de nomenclatura “CMP_xxx_ID” deve ser mantida obrigatoriamente, alterando o padrão “xxx” por valor numérico sequencial, como representado na coluna “*Component Name*” da Figura 36, em que o primeiro componente, obrigatoriamente, recebe o nome de CMP_001_ID e os demais assumem o valor sequencial “CMP_002_ID”, “CMP_003_ID” e “CMP_004_ID”.

Figura 36 - Exemplo de configuração de *Assembly*



Fonte: Autor, 2017

A configuração do tipo *Assembly* necessita da criação de um projeto na aplicação Automator. A aplicação é objeto de desenvolvimento deste estudo de caso e é apresentada no capitulo 4. A configuração será explicada no capítulo 5.

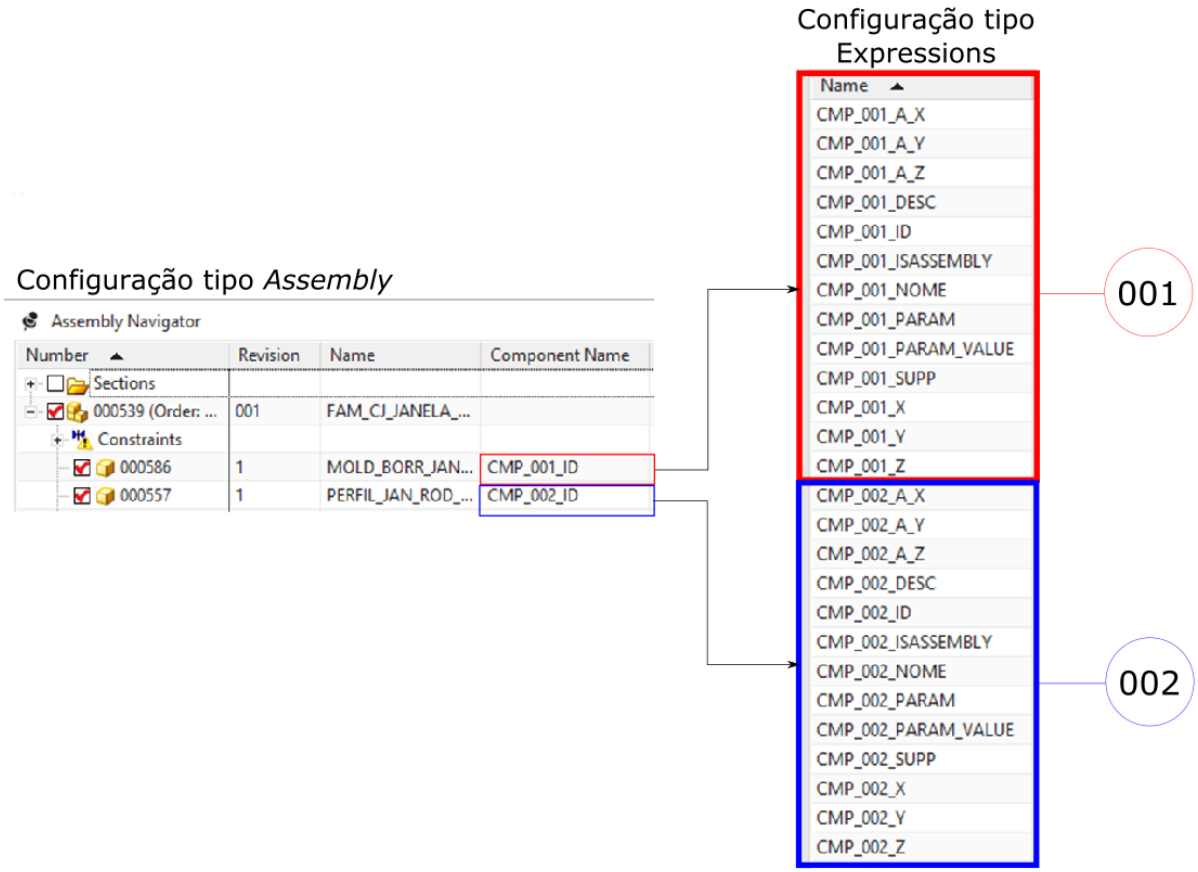
3.4.3 Configuração do tipo Expressions

Usualmente, este tipo de configuração é utilizado somente nos arquivos CAD do tipo *Template* e Família de peças, pois estas expressões são responsáveis por criar e adicionar peças ao espaço 3D.

Quanto à configuração do tipo *Expressions*, deve-se criar, para cada componente da estrutura, um conjunto de treze expressões: CMP_xxx_X, CMP_xxx_Y, CMP_xxx_Z, CMP_xxx_A_X, CMP_xxx_A_Y, CMP_xxx_A_Z, CMP_xxx_ISASSEMBLY, CMP_xxx_NOME, CMP_xxx_DESC, CMP_xxx_PARAM, CMP_xxx_PARAM_VALUE, CMP_xxx_SUPP e CMP_xxx_ID. Deve-se manter a estrutura de nomenclatura, alterando o padrão “xxx” por valores numéricos sequenciais.

A Figura 37 exemplifica a criação de dois conjuntos de expressões criadas no *software* Siemens NX. Parte-se da configuração do tipo *Assembly* e criam-se as expressões, na quais o padrão “xxx” foi alterado para “001” e “002”, sequencialmente.

Figura 37 - Exemplo de dois conjuntos que utilizam a configuração do tipo *Expressions*



Fonte: Autor, 2017

Para criação de uma expressão no *software* Siemens NX é necessário informar dois valores: nome da expressão (*Name*) e tipo de dado (*Type*). A seguir, serão detalhadas cada uma das expressões, já alterando o padrão “xxx” para “001”, facilitando assim o entendimento:

- CMP_001_X: Tipo de dados *number*, refere-se à posição geométrica do componente no eixo X. Exemplo: CMP_001_X = 0.
- CMP_001_Y: Tipo de dados *number*, refere-se à posição geométrica do componente no eixo Y. Exemplo: CMP_001_Y = 0.
- CMP_001_Z: Tipo de dados *number*, refere-se à posição geométrica do componente no eixo Z. Exemplo: CMP_001_Z = 0.
- CMP_001_A_X: Tipo de dados *number*, refere-se à rotação, em graus, do componente no eixo X. Exemplo: CMP_001_A_X = 45.
- CMP_001_A_Y: Tipo de dados *number*, refere-se à rotação, em graus, do componente no eixo Y. Exemplo: CMP_001_A_Y = 90.

- CMP_001_A_Z: Tipo de dados *number*, refere-se à rotação, em graus, geométrica do componente no eixo Z. Exemplo: CMP_001_A_Z = -45.
- CMP_001_ISASSEMBLY: Tipo de dados *number*, 0 (zero) se o componente for uma montagem (duas ou mais peças) ou 1 (um) se o componente for uma peça. Exemplo: CMP_001_ISASSEMBLY = 1.
- CMP_001_NOME: Tipo de dados *string*, nomenclatura final que será utilizada para criar o componente. Exemplo: CMP_001_NOME = "CJ_JAN_CORR_ROD_250x1320".
- CMP_001_DESC: Tipo de dados *string*, descrição final que será utilizada para criar o componente. Geralmente utiliza-se o mesmo que a nomenclatura. Exemplo: CMP_001_DESC = CMP_001_NOME.
- CMP_001_PARAM: Tipo de dados *string*, parâmetros para criação automática do componente caso ele não exista no banco de dados. Deve-se seguir a seguinte lógica: "FAM|PARAMETRO1|PARAMETRO2|ETC". O primeiro parâmetro é referente ao código da família de produto, campo fixo. Os demais atributos referem-se ao título das colunas do *Part Family*, campos variáveis. A Figura 38 exemplifica a utilização dos parâmetros: CMP_001_PARAM = "FAM|ALTURA_VAO|LARGURA_VAO". As medidas informadas são em milímetros.
- CMP_001_PARAM_VALUE: Tipo de dados *string*, valores para cada parâmetro informado na expressão CMP_001_PARAM. Os valores são tabelados por '|' (*pipie*). Exemplo: CMP_001_PARAM_VALUE = "000539|500~0|500~5", em que o primeiro parâmetro refere-se ao código da família de produto (FAM); o segundo refere-se ao parâmetro ALTURA_VAO; e o terceiro parâmetro refere-se ao parâmetro LARGURA_VAO. Ainda é possível adicionar uma tolerância para a busca e criação do item. A tolerância deve ser informada sempre após o valor desejado, separando-o pelo caractere '~' (til). Conforme o exemplo dado, o sistema tentará encontrar um projeto com largura do vão igual a 500 mm. Caso não seja encontrado, o sistema tentará diminuir 1 milímetro do valor estipulado, fazendo buscas até o limite de tolerância informado, neste caso 495 mm. Caso nenhum item seja encontrado, o sistema irá criar o item com o valor nominal de 500 mm.

- CMP_001_SUPP: Tipo de dados *number*, 0 (zero) se o componente ficará invisível (*hide*) na montagem final ou 1 (um) se o componente ficará visível (*visible*) na montagem final.
- CMP_001_ID: Tipo de dados *number*, código do projeto atribuído ao componente.

Figura 38 - Exemplo de utilização de parâmetros

E	F	G	H
DB_PART_DESC	***DB_PART_TYPE***	***ALTURA_VAO***	***LARGURA_VAO***
FAM_CJ_JANELA_CORR_RODOVIARIA	C9_item_fam_mest	1000	1600
CJ_JAN_CORR_ROD_1000x1600	C9_item_mestrado	1000	1600
CJ_JAN_CORR_ROD_920.0x920	C9_item_mestrado	920	920
CJ_JAN_CORR_ROD_920.0x1720.0	C9_item_mestrado	920	1720

Fonte: Autor, 2017

Conclui-se este capítulo com o entendimento do funcionamento do método aplicado aos *templates*. Apresentou-se os quatro tipos de projetos e as três diferentes configurações possíveis nivelando a compreensão da construção dos *templates*, que acontecem no Capítulo 4 e no Capítulo 5.

O capítulo subsequente é destinado à compreensão do projeto e desenvolvimento das aplicações CustomNX, Automator e Servidora de Regras.

4 PROJETO E DESENVOLVIMENTO DO MÉTODO DE AUTOMAÇÃO

Para a utilização do método proposto no Capítulo 3 serão desenvolvidas e apresentadas três aplicações, conforme representado na Figura 39, denominadas neste trabalho como:

a) **CustomNX**: A função principal desta aplicação é criar uma variação de produto para um determinado pedido de venda. Em modo funcional, a aplicação consulta configurações de vendas na base de dados, efetua os cálculos das regras desenvolvidas para o determinado *template* e adiciona os componentes no espaço 3D do CAD, formando assim a variação do produto final.
b) **Automator**: A função principal desta aplicação é a criação e manutenção de regras para o *template*. Cada *template* possui um conjunto de regras específico.
c) **Servidor de Regras**: Por fim, o servidor de regras tem a função principal de gerenciar a utilização e criação das regras pelas aplicações CustomNX e Automator.

A organização deste capítulo é feita em cinco sessões: arquitetura conceitual do sistema, aplicação servidor de regras, aplicação Automator, aplicação CustomNX e alterações em outros sistemas, conforme ilustra a Figura 39.

Figura 39 - Sessões do Capítulo 3

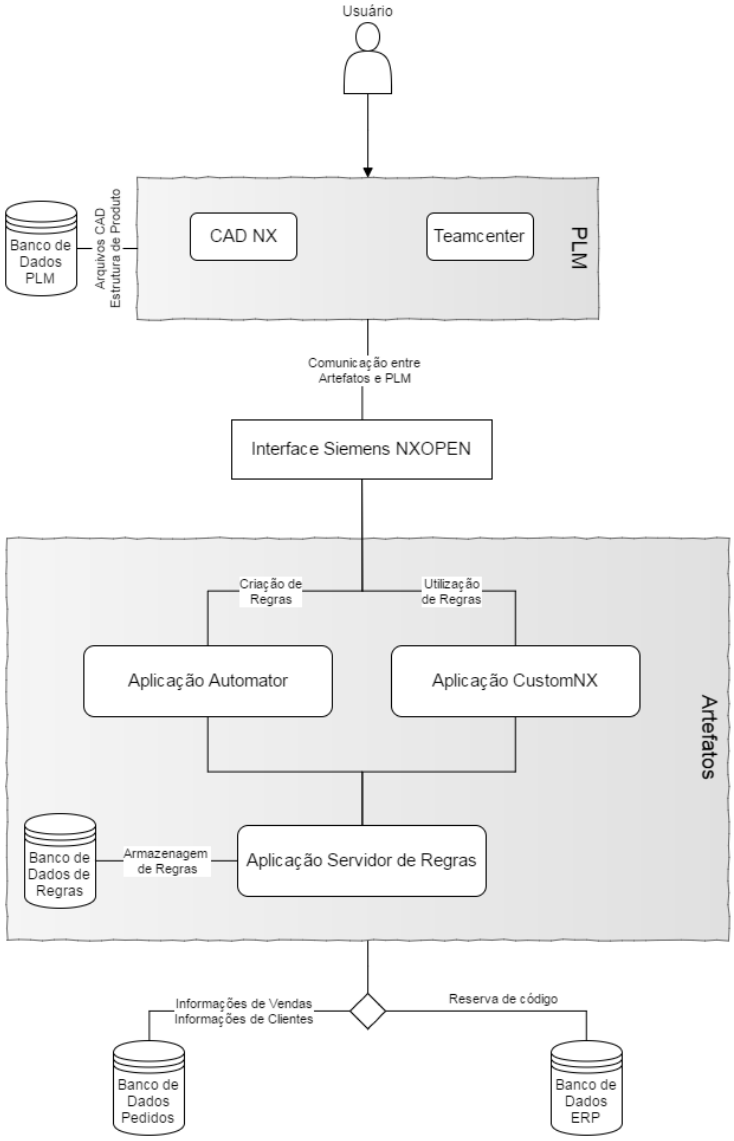
Fonte: Autor, 2017

A seção arquitetura conceitual do sistema detalha a estrutura funcional do sistema proposto, ilustrando as interações entre aplicações e usuário. As sessões aplicação servidor de regras, aplicação Automator, aplicação CustomNX e alterações em outros sistemas referem-se ao projeto e desenvolvimento das aplicações propostas. Estas aplicações serão detalhadas abordando o processo de desenvolvimento de modelos abstratos do sistema, para que cada modelo apresente uma perspectiva diferente do sistema.

4.1 Arquitetura conceitual do sistema

A estrutura fundamental do sistema é definida pela arquitetura representada na Figura 40. Em termos de usuário, foram desenvolvidas duas aplicações: aplicação para criação de regras – definida como **Automator** – e aplicação para utilização das regras, definida como **CustomNX**. Para gerenciar as regras foi desenvolvida a aplicação denominada **Aplicação Servidor de Regras**, juntamente com seu banco de dados. O servidor de regras ainda possui transações com o banco de dados ERP.

Figura 40 - Arquitetura do sistema



Fonte: Autor, 2017

As aplicações CustomNX e Automator são executadas em conjunto com o *software* CAD Siemens NX, utilizando, para isso, um conjunto de APIs denominado NX Open. A troca de mensagens entre cliente e servidor, por meio de protocolo HTTP, refere-se à troca de arquivos CAD 3D, cálculo de regras e criações de códigos no ERP.

4.1.1 Configuração atual do ambiente

Os aplicativos foram desenvolvidos utilizando-se notebook DELL, processador Intel Core I7-4900MQ 2.80 GHz, 500 GB de Hard Drive, 32 GB de memória e placa de vídeo NVIDIA Quadro K2100M, com sistema operacional Windows Enterprise 7 64 bits. O ambiente de desenvolvimento e teste foi em máquina virtual, utilizando o *software* de virtualização VMware Workstation 12 Player, versão 12.1.1, utilizando o sistema operacional Windows Server 2012 R2.

Para plataforma PLM foram utilizados os *softwares* Siemens NX 9 e Teamcenter 9. Foi utilizado o banco de dados MySQL com gerenciador XAMPP, na versão 3.2.1. Para modelagem de banco de dados utilizou-se a ferramenta visual Oracle MySQL Workbench 6.3 Community e para a modelagem do sistema utilizou-se o *software* Visual Paradigm Community 13.1.

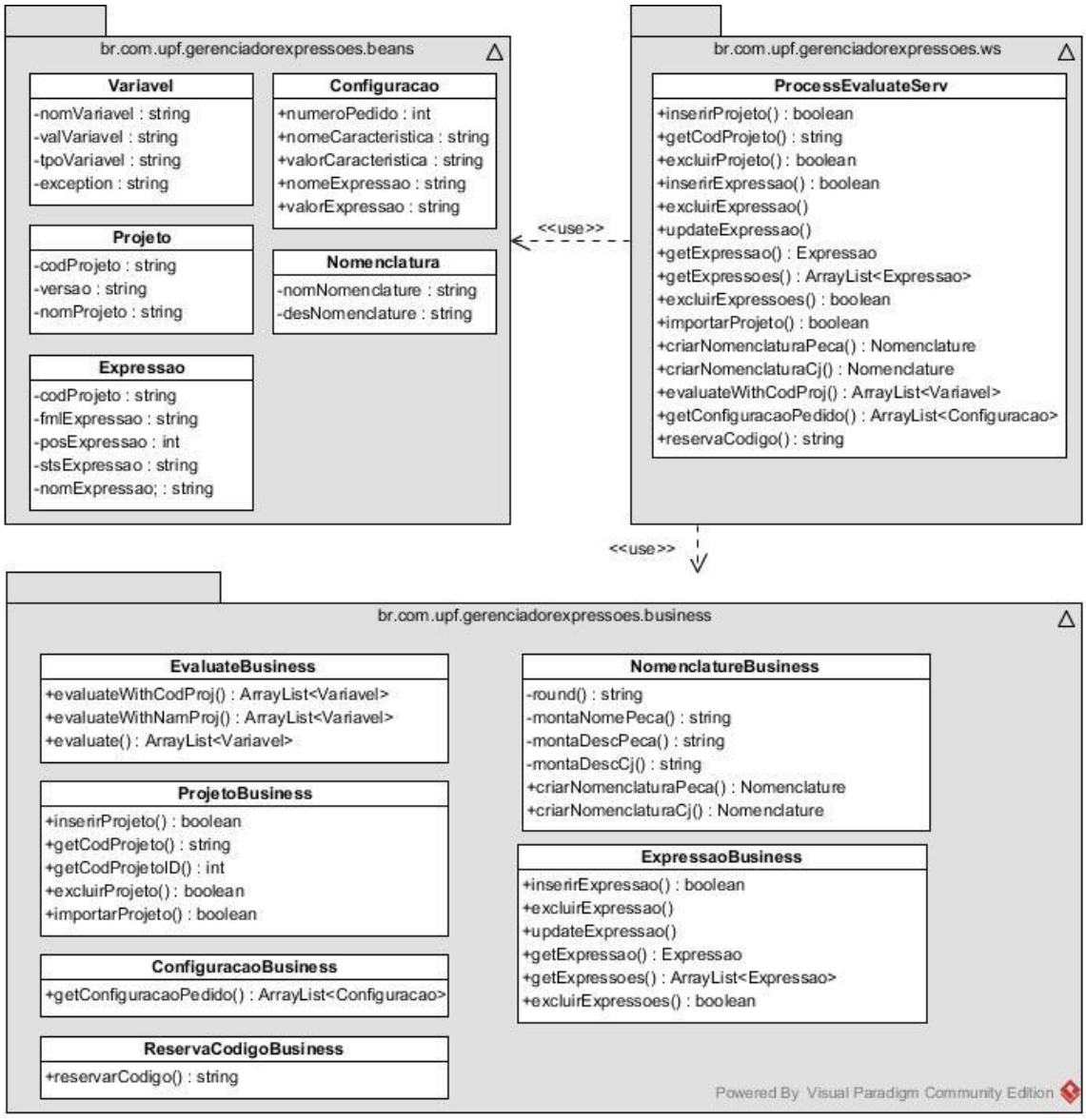
A plataforma (IDE) de desenvolvimento dos *webservices* foi NetBeans 8.1, com servidor Glassfish 4.1.1 e linguagem de programação JAVA. A IDE de desenvolvimento dos aplicativos *desktops* foi o Microsoft Visual Studio 2015 com linguagem de programação C# (CSharp).

4.2 Aplicação Servidor de Regras

O diagrama de classes que serve de modelo para os objetos da aplicação de regras está representado na Figura 41. Foram desenvolvidos três pacotes de classes: *beans*, *ws* e *business*. Para o pacote *beans* (classe básica/primária) foram desenvolvidas cinco classes: Variável, Projeto, Expressão, Configuração e Nomenclatura. Para o pacote *business* (classe de negócio, responsável pelas lógicas e transações com banco de dados) foram desenvolvidas seis classes: EvaluateBusiness, ProjetoBusiness, ConfiguracaoBusiness, ReservaCodigoBusiness, NomenclaturaBusiness e Expressão Business. Por fim, o pacote de classes *ws* (classe *webservice*) possui uma classe: ProcessEvaluateServ. Esta classe *ws* é responsável pela integração entre aplicação cliente e servidor, tendo disponíveis quinze métodos: inserirProjeto,

getCodProjeto, excluirProjeto, inserirProjeto, inserirExpressão, excluirExpressao, updateExpressao, getExpressao, getExpressoes, excluirExpressao, importarProjeto, criarNomenclaturaPeca, criarNomenclaturaCj, evaluateWithCodProj, getConfiguracaoPedido e reservaCodigo.

Figura 41 - Diagrama de classes Aplicação de Regras

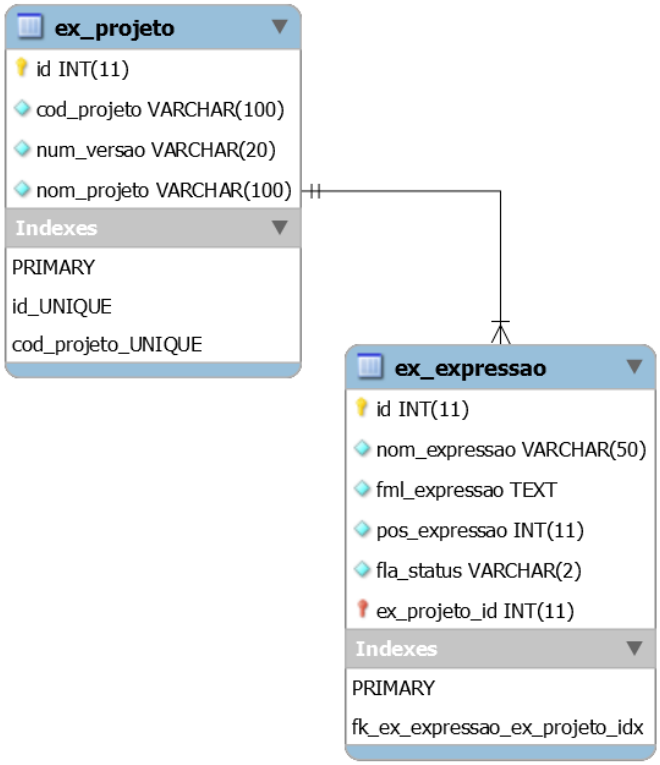


Fonte: Autor, 2017

O Modelo Entidade-Relacionamento (MER, Modelo ER) se baseia no princípio em que todos os dados estão guardados em tabelas e, por meio da modelagem de dados, é possível a representação, de forma clara, de toda a base de informação necessária para o determinado sistema. O modelo ER da aplicação Servidor de Regras está representado na Figura 42. A

aplicação possui duas instâncias: ex_projeto e ex_expressao. O relacionamento entre as duas tabelas é de 1:N (um projeto para muitas expressões).

Figura 42 - Modelo ER da aplicação servidor de regras

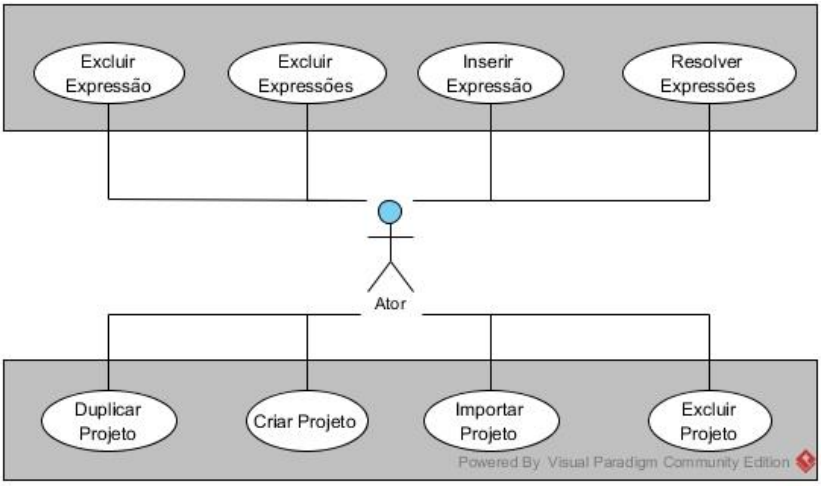


Fonte: Autor, 2017

4.3 Aplicação Automator

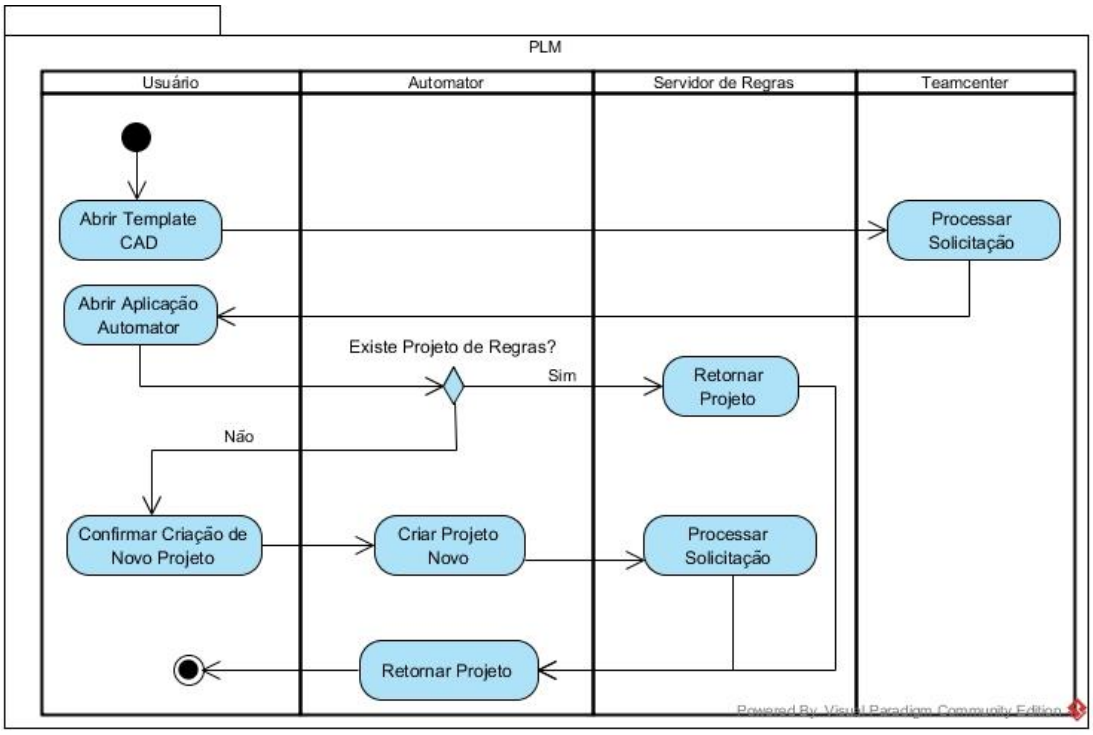
Diagrama de Casos de uso é utilizado para representar de forma modular o sistema e sua interação com os usuários. Neste diagrama geral do sistema, apresentado na Figura 43, é descrito um cenário com oito principais funcionalidades do ponto de vista do usuário: criar projeto, duplicar projeto, importar projeto, excluir projeto, resolver expressões, excluir expressões, excluir projeto e inserir expressão.

Figura 43 - Caso de uso manter *Automator*



Fonte: Autor, 2017

Figura 44 - Diagrama de atividade abrir projeto de regras

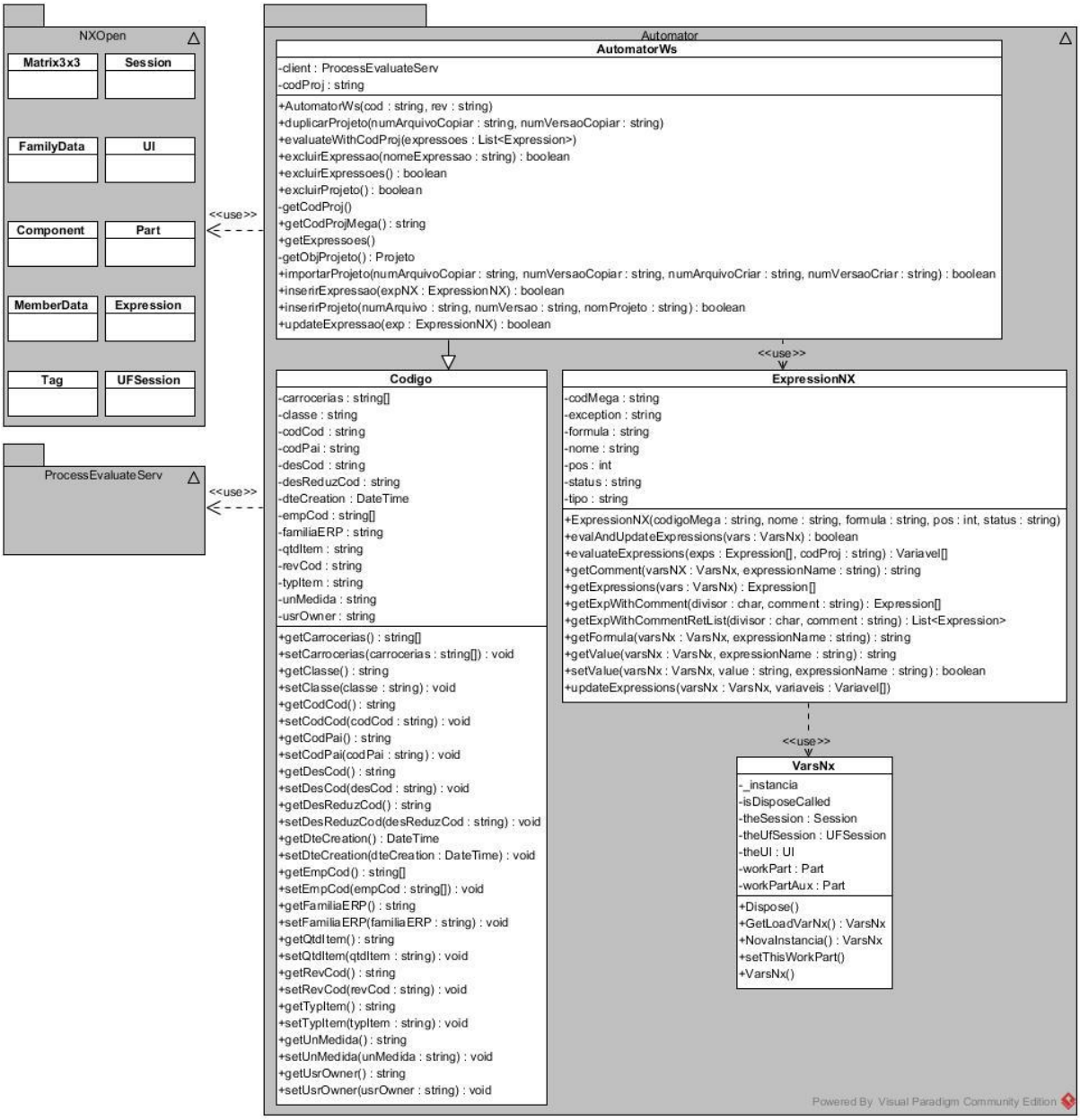


Fonte: Autor, 2017

O diagrama de classes que serve de modelo para os objetos da aplicação Automator está representado na Figura 45. Foi desenvolvido um pacote de classes denominado Automator, contendo quatro classes: AutomatorWs, Codigo, ExpressionNX e VarsNx. Este pacote de

classes se relaciona com o pacote de classes externa NX Open e também com a aplicação servidor de regras ProcessEvaluateServ.

Figura 45 - Diagrama de classes aplicação *Automator*

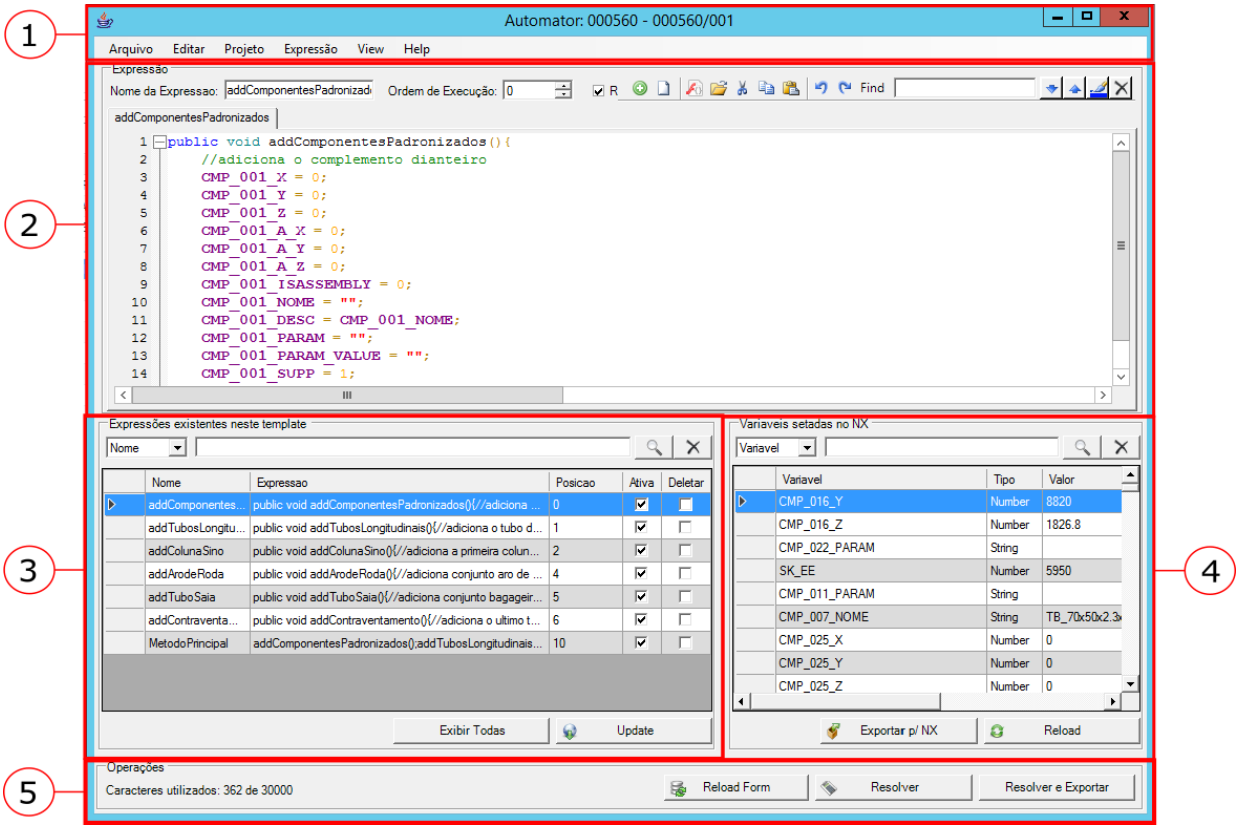


Fonte: Autor, 2017

4.3.1 Tela aplicação Automator

A *interface* da aplicação *Automator*, representada na Figura 46, é dividida em seis módulos: (1) menus, (2) área para criação dos cálculos, (3) área contendo todas as expressões existentes no projeto, (4) área com todas as expressões disponíveis no NX e (5) área com botões de operações.

Figura 46 - Interface aplicação *Automator*



Fonte: Autor, 2017

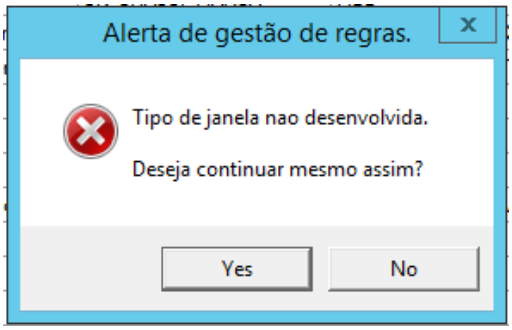
4.3.2 Recursos especiais da aplicação

A aplicação Automator possui dois recursos importantes para o projeto e desenvolvimento de *templates*. O recurso de reportar ao usuário um alerta durante a execução do *template* e outro recurso para armazenar funções especiais ou de uso comum. Estes recursos são detalhados a seguir.

4.3.2.1 Gerenciamento de alertas ao usuário

Este recurso foi desenvolvido a fim de alertar o usuário final de que a configuração que ele está aplicando ao *template* possui alguma característica especial e o resultado final poderá ficar inconsistente.

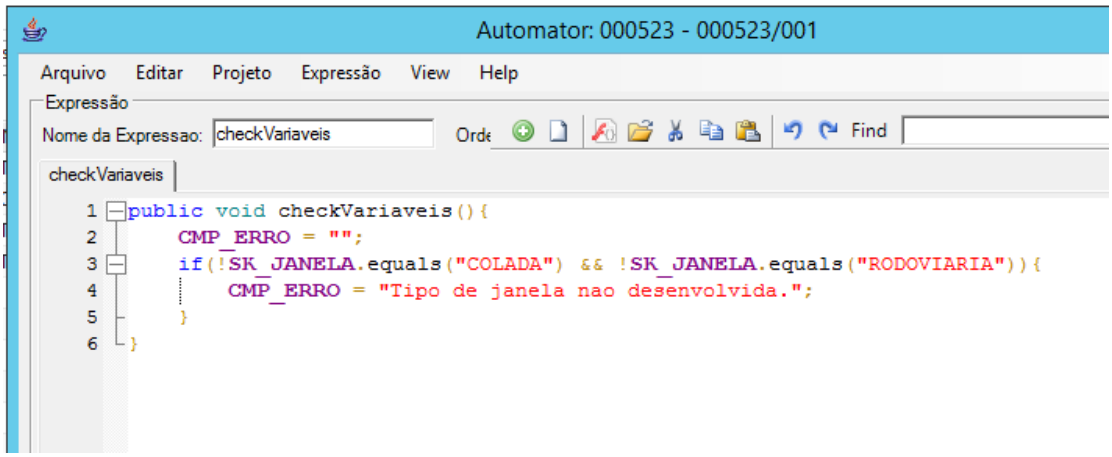
Figura 47 - Alerta de gestão de regras



Fonte: Autor, 2017

A ativação deste recurso se faz abastecendo uma mensagem em uma variável denominada CMP_ERRO, sendo que, posteriormente, a mensagem é apresentada ao usuário, conforme representado na Figura 47.

Figura 48 - Verificação de variáveis para o *template*



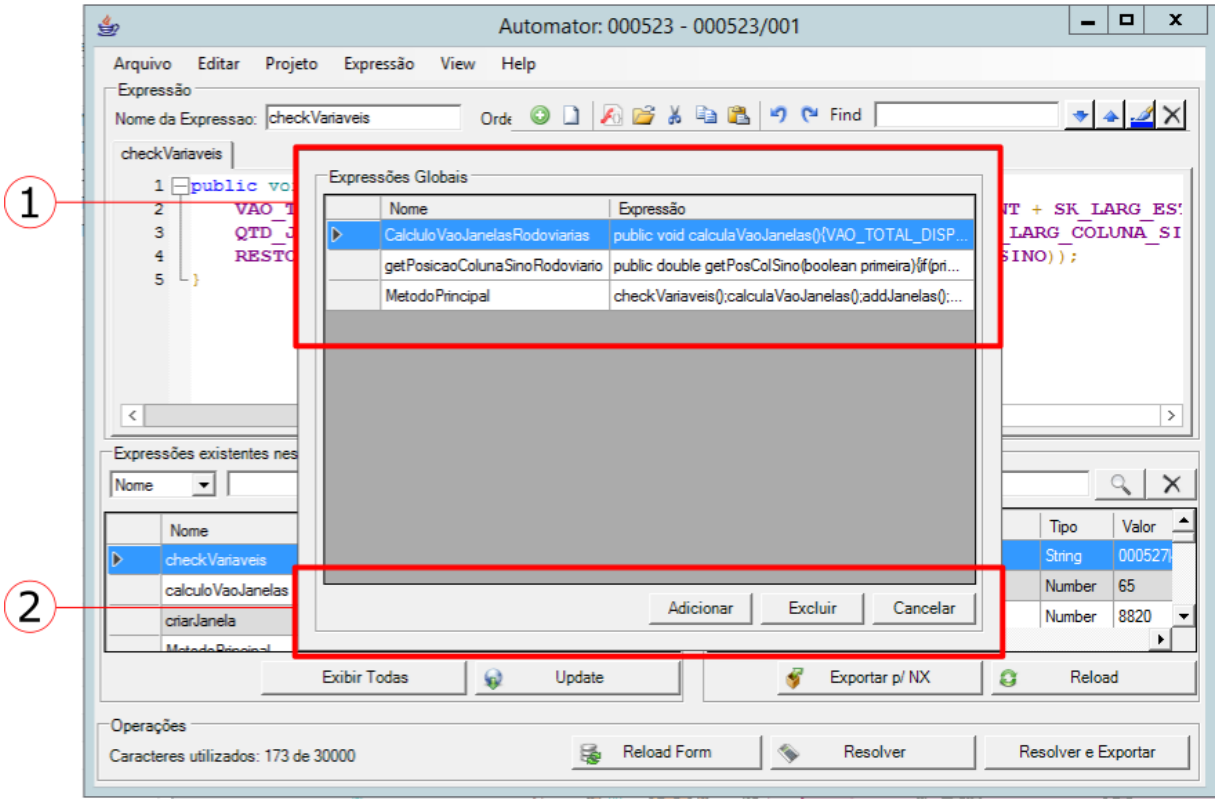
Fonte: Autor, 2017

Na Figura 48 é exemplificada a utilização deste recurso, em que o método é responsável por verificar se a configuração aplicada no *template* está dentro do escopo de desenvolvimento; em outras palavras, verifica se o *template* possui as regras necessárias para gerar um produto a partir das características de venda. O método “checkVariaveis” indica que o *template* está desenvolvido somente para janelas do tipo rodoviária e do tipo colada.

4.3.2.2 Gerenciamento de funções globais

O recurso para gerenciamento de funções globais foi desenvolvido a fim de fazer reuso de métodos completos ou frações de códigos.

Figura 49 - Gerenciamento de funções globais



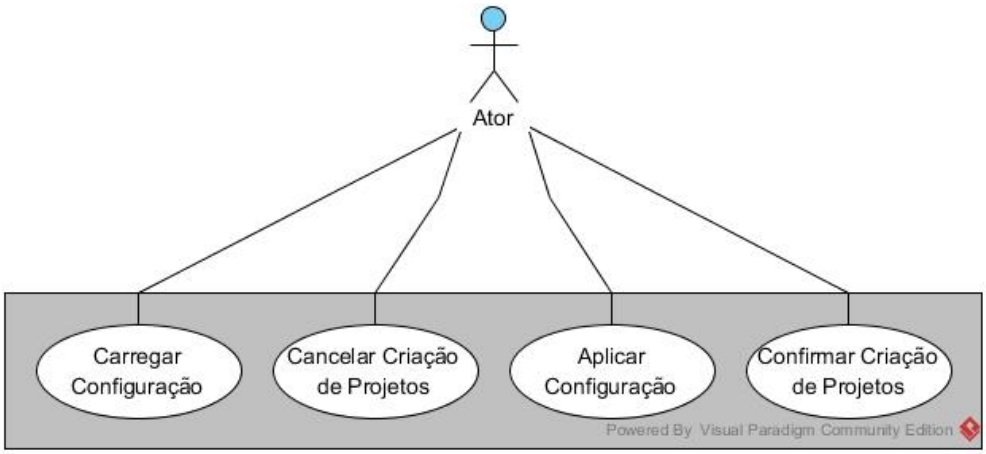
Fonte: Autor, 2017

Conforme ilustrado na Figura 49, as expressões globais estão disponíveis a qualquer usuário e *template*. O projetista poderá escolher a expressão (1) desejada e adicioná-la ao *template* (2).

4.4 Aplicação CustomNX

Conforme representado na Figura 50, foi desenvolvido um diagrama de caso de uso para auxiliar no entendimento de quais ações o usuário terá sobre o sistema. Neste diagrama é descrito um cenário com quatro principais funcionalidades do sistema do ponto de vista do usuário (ator): Carregar Configuração, Aplicar Configuração, Criar Projetos e Cancelar Criação.

Figura 50 - Caso de uso Manter CustomNX



Fonte: Autor, 2017

A Figura 51 descreve o fluxo de informação resultante do processamento da ação do usuário, iniciando com a abertura do arquivo CAD *template* até a finalização do projeto e processamento do sistema. Neste cenário é possível observar as aplicações CustomNX e Servidor de Regras trocando informações com o ambiente PLM. Em tarefas sequenciais, o usuário abre o *template* CAD, em sequência abre a aplicação CustomNX e informa apenas o número do pedido como parâmetro. Com a configuração de venda na tela, o usuário inicia o processo aplicando a configuração.

A partir deste momento a aplicação CustomNX solicita a resolução de regras ao Servidor de Regras, enviando, para isso, o conjunto de informações da configuração de venda (pedido). Em seguida, o Servidor de Regras retorna, para a aplicação CustomNX, um conjunto de informações resultantes do cálculo das regras. Regras estas que são específicas para o determinado *template*.

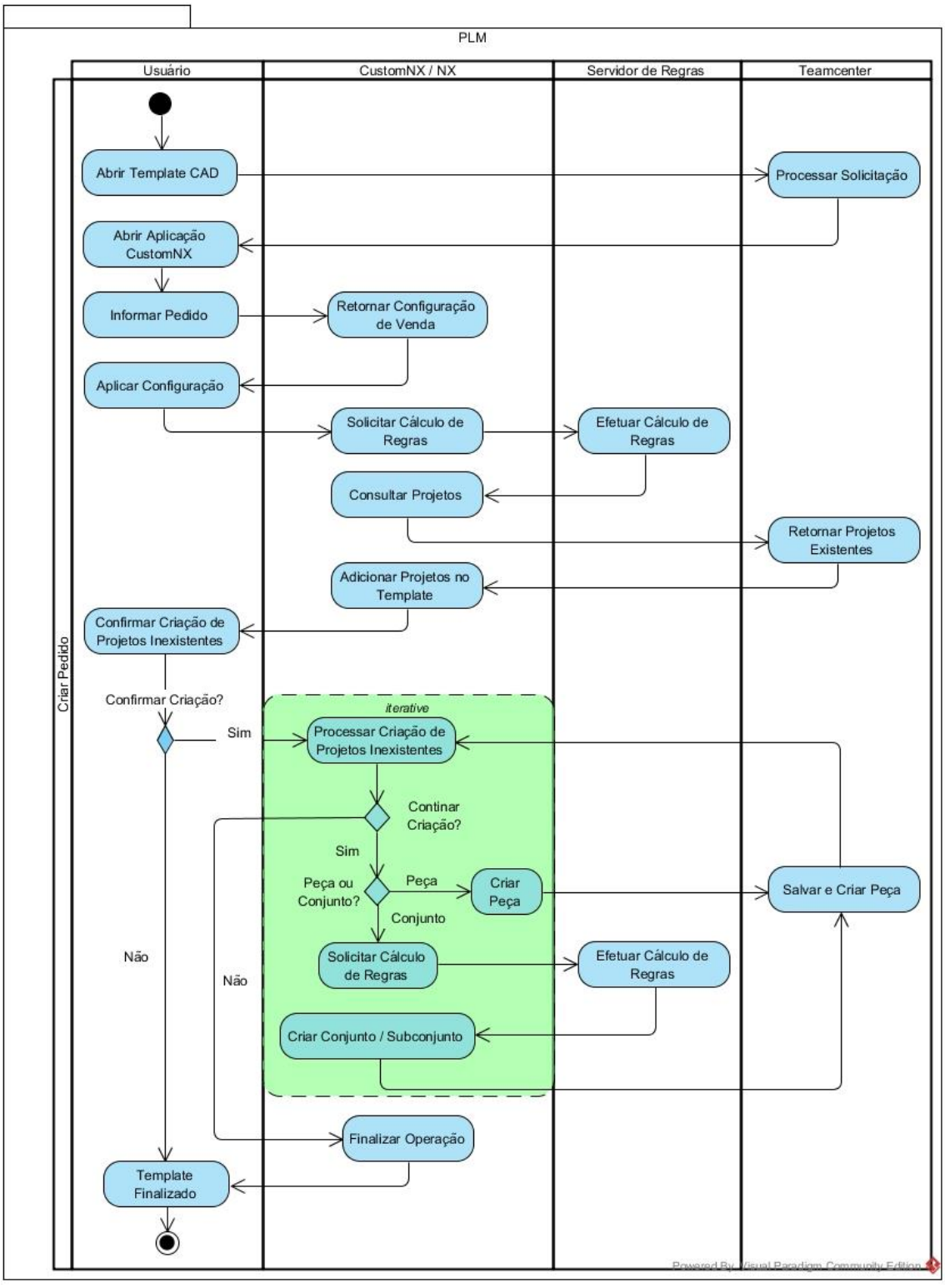
Com o cálculo das regras efetuado, a aplicação CustomNX faz a separação dos projetos existentes dos projetos inexistentes no ambiente PLM. Por meio das informações calculadas pelo Servidor de Regras, os projetos existentes são alocados no espaço 3D. Finalizando esta etapa, o sistema solicita intervenção do usuário, caso exista algum projeto inexistente, para que dê permissão para iniciar a criação dos projetos faltantes.

Caso a permissão seja negada, a operação é finalizada; caso contrário, a aplicação CustomNX abre todos arquivos CAD necessários e faz a separação distinta entre arquivos do tipo Peça e arquivos do tipo Conjunto. O processo de criação começa pelos arquivos do tipo Peça, sendo que nenhum cálculo de regras é necessário. Posteriormente, finaliza-se o processo

efetuando-se a criação dos arquivos do tipo Conjunto, sendo que se faz necessário recálculo de regras no Servidor de Regras.

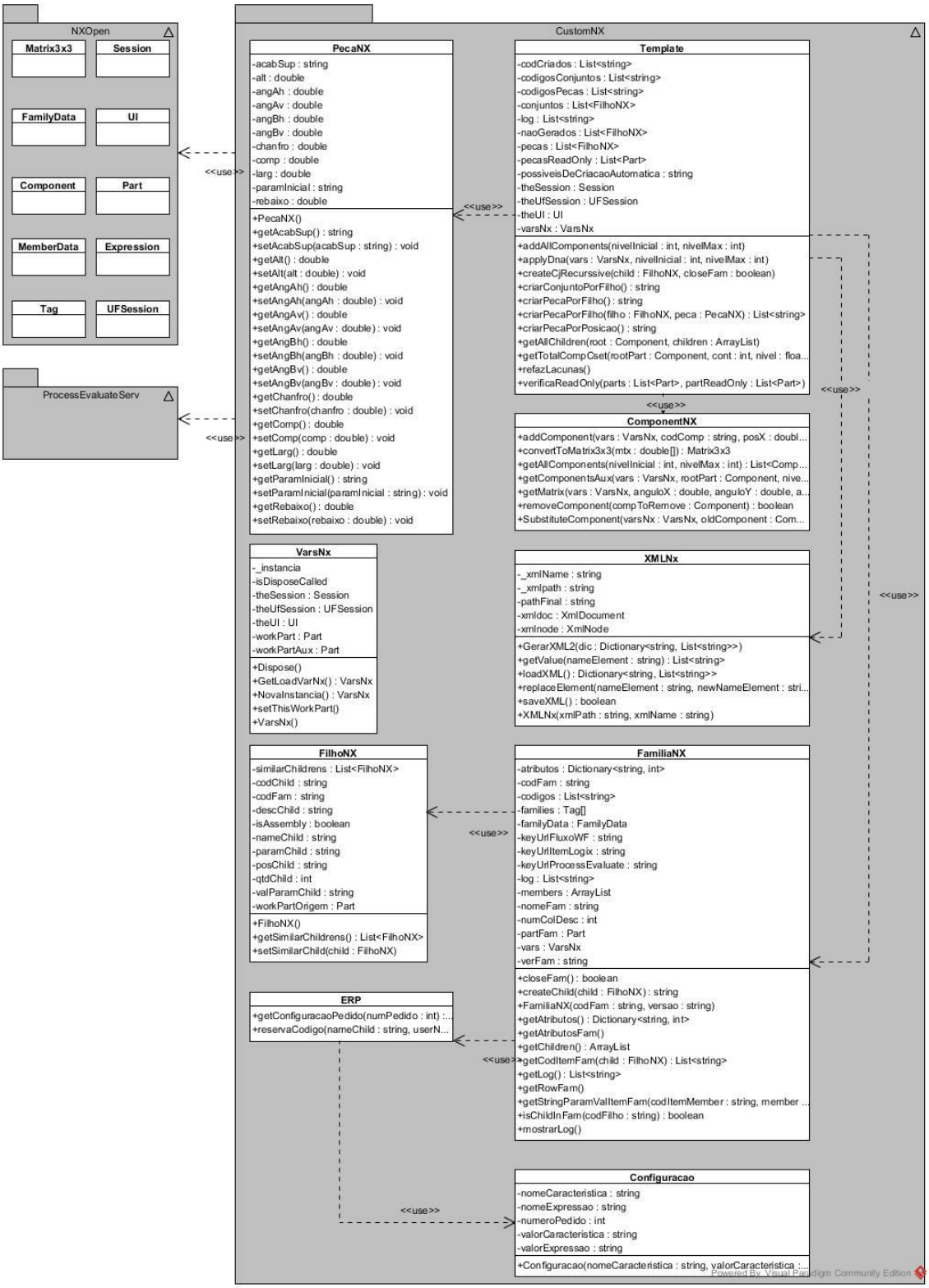
O diagrama de classes que serve de modelo para os objetos da aplicação CustomNX está representado na Figura 52. Foi desenvolvido um pacote de classes denominado CustomNX, contendo nove classes: PecaNX, Template, VarsNX, ComponentNX, XMLNx, FilhoNX, FamiliaNX, ERP e Configuracao. Este pacote de classes se relaciona com o pacote de classes externa NXOpen e também com a aplicação servidor de regras ProcessEvaluateServ.

Figura 51 - Diagrama de atividade aplicar configuração



Fonte: Autor, 2017

Figura 52 - Diagrama de classes aplicação CustomNX

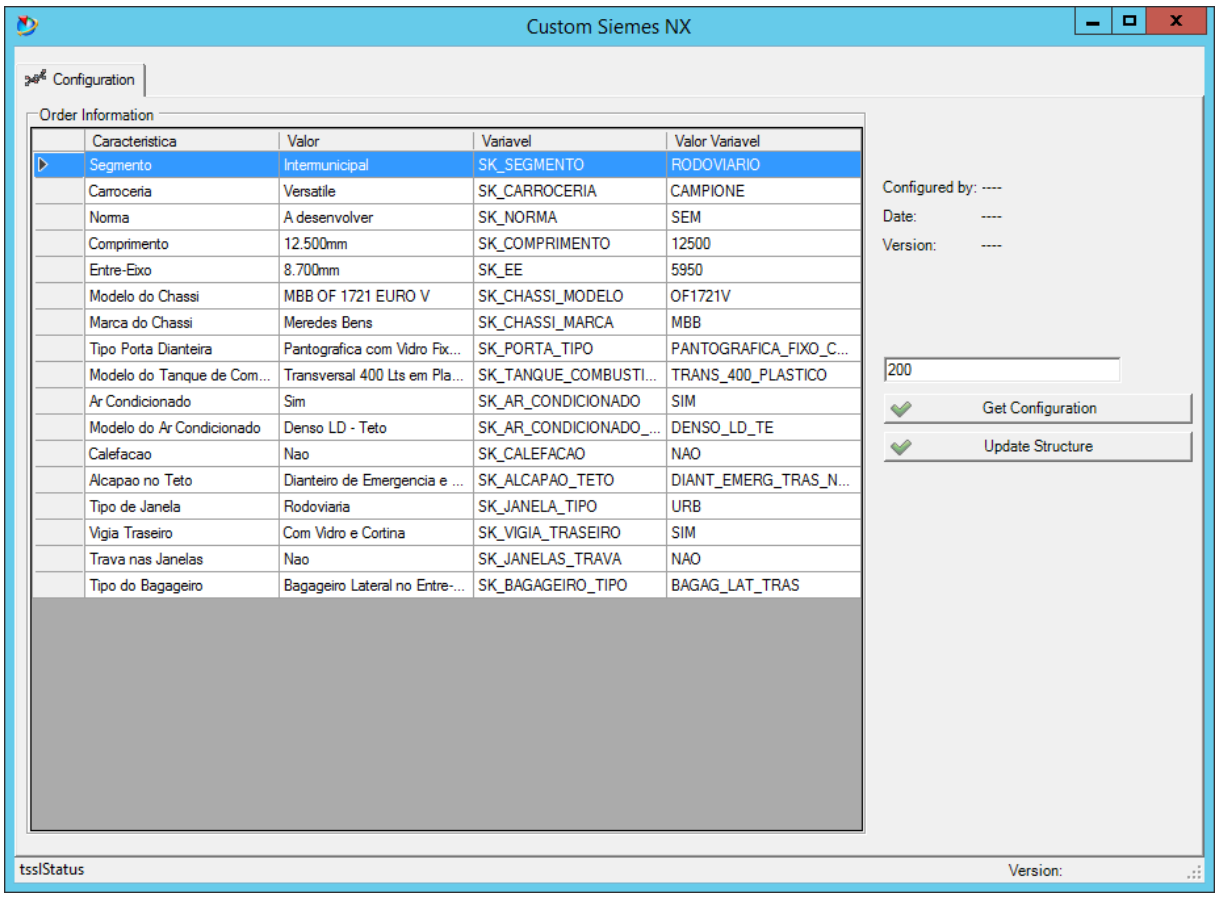


Fonte: Autor, 2017

4.4.1 Tela aplicação CustomNX

A *interface* da aplicação CustomNX, representada na Figura 53, possui apenas um módulo com as opções de carregar pedido e aplicar configuração.

Figura 53 - Interface aplicação CustomNX



Fonte: Autor, 2017

4.5 Alterações em outros sistemas

A principal fonte de informações para as aplicações desenvolvidas nessa dissertação são as tarefas de vendas, ou seja, o *software* configurador de produto.

Para facilitar o desenvolvimento das regras no *software* desenvolvido neste capítulo, o Automator, criou-se dois novos atributos no *software* configurador de produto: “Variável” e “Valor Variável”, conforme ilustrado na Figura 54. Em ambos os campos padronizou-se uma nomenclatura sem espaços e/ou caracteres especiais para os valores das características e opções.

Como estes valores são únicos no *software* configurador de produto, não existirá outro cadastro de característica com o nome “SK_SEGMENTO”, por exemplo. O prefixo SK remete a palavra *skeleton*.

Figura 54 - Alteração do configurador de produto

Caracteristica	Valor	Variavel	Valor Variavel
Segmento	Intermunicipal	SK_SEGMENTO	RODOVIARIO
Carroceria		SK_CARROCERIA	
Norma	A desenvolver	SK_NORMA	SEM
Comprimento	12.500mm	SK_COMPRIMENTO	12500
Entre-Eixo	8.700mm	SK_EE	5950
Modelo do Chassi	MBB OF 1721 EURO V	SK_CHASSI_MODELO	OF1721V
Marca do Chassi	Meredes Bens	SK_CHASSI_MARCA	MBB
Tipo Porta Dianteira	Pantografica com Vidro Fixo Colado	SK_PORTA_TIPO	PANTOGRAFICA_FIXO_COLADO
Modelo do Tanque de Combustivel	Transversal 400 Lts em Plastico	SK_TANQUE_COMBUSTIVEL	TRANS_400_PLASTICO
Ar Condicionado	Sim	SK_AR_CONDICIONADO	SIM
Modelo do Ar Condicionado	Denso LD - Teto	SK_AR_CONDICIONADO_MODELO	DENSO_LD_TE
Calefacao	Nao	SK_CALEFACAO	NAO
Alcapao no Teto	Dianteiro de Emergencia e Traseiro Normal	SK_ALCAPAO_TETO	DIANT_EMERG_TRAS_NORM
Tipo de Janela	Rodoviaria	SK_JANELA_TIPO	URB
Vigia Traseiro	Com Vidro e Cortina	SK_VIGIA_TRASEIRO	SIM
Trava nas Janelas	Nao	SK_JANELAS_TRAVA	NAO
Tipo do Bagageiro	Bagageiro Lateral no Entre-Eixo e Passante na	SK_BAGAGEIRO_TIPO	BAGAG_LAT_TRAS

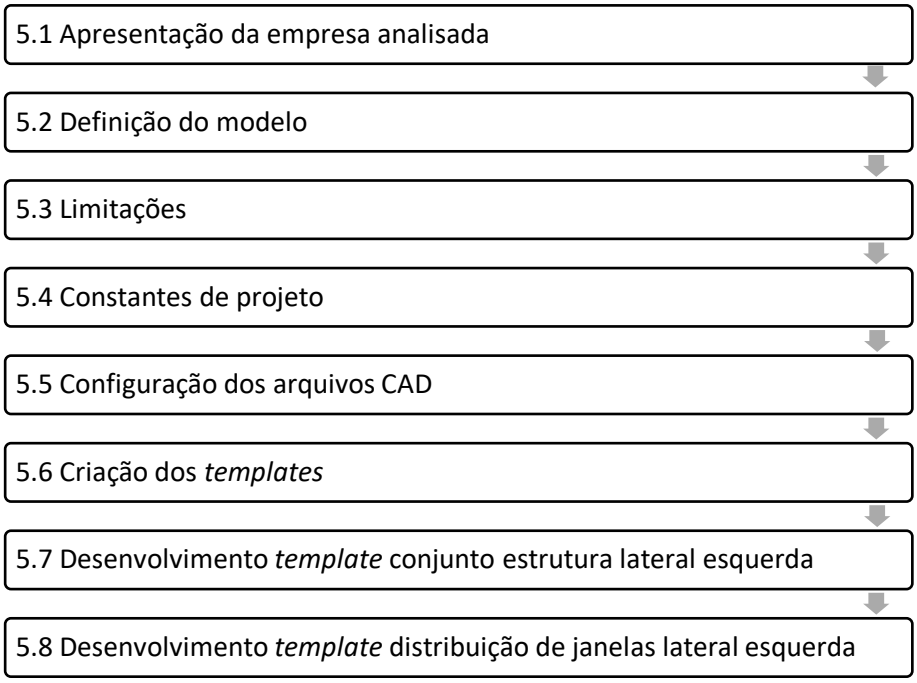
Fonte: Autor, 2017

Este capítulo descreveu o projeto e desenvolvimento das três aplicações responsáveis por solucionar o problema proposto. O entendimento do funcionamento e arquitetura dessas três aplicações é essencial para a compreensão do próximo capítulo, no qual o método será aplicado a solucionar um problema real.

5 APLICAÇÃO DO MÉTODO: UM ESTUDO DE CASO

Este capítulo está dividido em oito sessões, sendo: Apresentação da empresa analisada, Definição do modelo, Limitações, Constantes de projeto, Configuração dos arquivos CAD, Criação dos *templates*, Desenvolvimento *template* conjunto estrutura lateral esquerda e Desenvolvimento *template* distribuição de janelas lateral esquerda, conforme ilustrado na Figura 51.

Figura 55 - Sessões do Capítulo 5



Fonte: Autor, 2017

A seção **Apresentação da empresa analisada** traz informações referentes à empresa a qual o estudo de caso foi desenvolvido.

A segunda seção, **Definição do modelo**, refere-se à explanação acerca da escolha dos *templates* para este estudo de caso, além da introdução dos projetos utilizados nos *templates*.

A seção **Limitações** apresenta restrições ao modelo apresentado no estudo de caso.

A seção **Constantes de projeto** detalha os padrões de projetos de carroceria de ônibus, na empresa estudada. Estas constantes de projeto são essenciais para o desenvolvimento dos *templates*.

A quinta seção, **Configuração dos arquivos CAD**, traz a explanação de todos os projetos utilizados para a criação dos *templates*.

A seção **Criação dos *templates*** detalha a criação dos arquivos no *software* CAD NX além da criação das constantes de projeto e utilização da aplicação Automator.

Os *templates* iniciam seus desenvolvimentos nesta seção, **Desenvolvimento *template* conjunto estrutura lateral esquerda**. Esta seção detalha como o *template* da estrutura lateral esquerda é desenvolvido.

Para finalizar o capítulo, a seção **Desenvolvimento *template* distribuição de janelas lateral esquerda** detalha como o *template* de distribuição de janelas lateral esquerda é produzido.

5.1 Apresentação da empresa analisada

Este estudo de caso foi realizado em uma empresa fabricante de carrocerias de ônibus. Sua fundação ocorreu no ano de 1986, sendo que a organização possui sua unidade fabril localizada no estado do Rio Grande do Sul. A unidade fabril possui aproximadamente 2.400 colaboradores e tem uma produção média de produção de 10 ônibus/dias. As linhas de produto são divididas em: Rodoviário, Intermunicipal, Urbano e Micro.

Tabela 4 - Produção de carrocerias de ônibus da empresa estudada

Ano	Mercado Interno (un.)	Mercado Externo (un.)	Total (un.)
2015	2.129	592	2.721
2014	3.107	346	3453
2013	3.326	616	3942
2012	3.331	478	3809
2011	4.118	543	4661
2010	3.245	716	3961
2009	2.652	508	3160

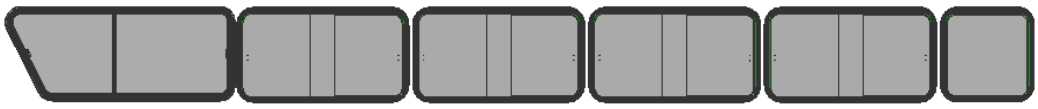
Fonte: Associação Nacional dos Fabricantes de Ônibus (FABUS)

Conforme Tabela 4, o mercado nacional é o principal mercado, porém a organização exporta seus produtos para mais de 30 países. Conforme dados de 2015, a empresa comercializou 2.721 unidades, sendo 2.129 (78,24%) unidades para o mercado brasileiro e 592 (21,76%) unidades para o mercado externo, ocupando a quarta posição entre os fabricantes de veículos coletivos (FABUS).

5.2 Definição do modelo

Para a aplicação do método, foi definido como objeto de estudo o conjunto de janelas da lateral esquerda de um ônibus do segmento rodoviário, representado na Figura 56. A escolha deste projeto se deu devido à facilidade de compreensão para qualquer leitor, visto que o conjunto de janelas é um dos objetos com que os usuários – do produto ônibus – mais interagem e possuem um comportamento visual mais fácil de ser interpretado.

Figura 56 - Conjunto janelas lateral esquerda

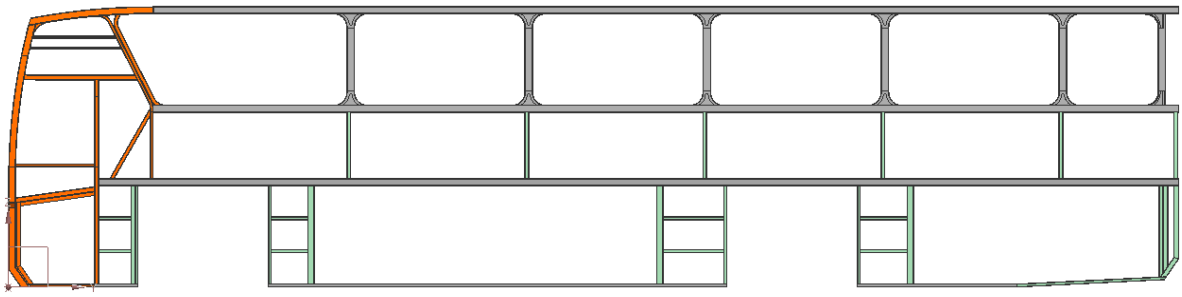


Fonte: Autor, 2017

Porém, conforme ilustrado pela Figura 56, observou-se a necessidade de contextualizar o uso do conjunto de janelas para que o leitor pudesse entender seu comportamento de uma forma visual e estruturada, como, por exemplo, não ser possível identificar a sustentação das janelas bem como seu posicionamento no ônibus.

Para solucionar este problema, foi acrescentado ao desenvolvimento o conjunto estrutural da lateral esquerda, conforme ilustrado na Figura 57, pois o comportamento das janelas é definido por suas geometrias. O projeto da estrutura lateral esquerda é um dos projetos mais manipulados dentro dos setores de engenharia de produto e engenharia de desenvolvimento, ajudando, assim, na validação dos objetivos deste trabalho.

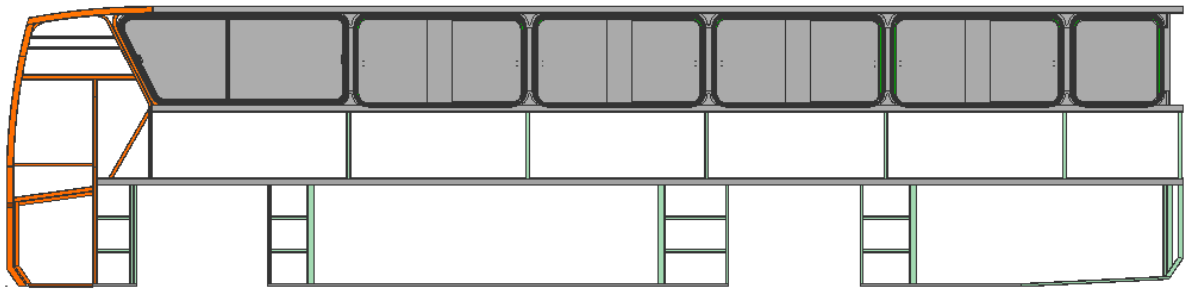
Figura 57 - Conjunto estrutura lateral esquerda



Fonte: Autor, 2017

Por fim, a Figura 58 exemplifica o resultado da aplicação do método, com a montagem do conjunto de vidros sobre o conjunto estrutural da lateral esquerda.

Figura 58 - Conjunto estrutura lateral esquerda com vidros laterais



Fonte: Autor, 2017

5.2.1 Configuração de venda utilizada

A configuração de venda utilizada para a construção dos modelos está descrita na Tabela 5, na qual a coluna “Nome” determina as características de venda que influenciam no modelo e a coluna “Valor” representa o valor determinado para a característica. Esta configuração é denominada de “**Pedido referência**” nesta dissertação.

Tabela 5 - Configuração de carroceria utilizada

Nome	Valor
Segmento	Rodoviário
Para-choque Dianteiro	330 mm
Para-choque Traseiro	390 mm
Modelo do Chassi	VW 17230
Posição do Motor	Dianteiro
Porta Recuada	Não
Comprimento	12,4 M
Tipo Balanço Dianteiro	Normal
Balanço Dianteiro	2300 mm
Tipo Entre-Eixo	Original
Entre-Eixo	5950 mm
Porta Motorista	Não
Tipo Porta Dianteira	Pantográfica
Tipo de Janela	Janela Rodoviária

Fonte: Autor, 2017

5.2.2 Tipos de arquivo CAD utilizados

Para o desenvolvimento deste estudo de caso foram necessários quinze projetos de produto. A divisão destes projetos deu-se conforme apresentado na seção 3.3. A

Tabela 6 relaciona os tipos de arquivos CAD utilizados com os tipos de configurações, sendo que o hífen (-) indica que **não há configuração** e o caractere (X) indica o **tipo de configuração**. É possível identificar que:

- 5 são famílias de peças:
  - 000537 – família de borracha para janela rodoviária,
  - 000540 – família de perfil para janela rodoviária,
  - 000544 – família de vidro para janela rodoviária,
  - 000569 – família de vidro para janela colada, e
  - 000527 – família de tubo retangular.
- 1 é família de conjuntos:
  - 000539 – família de janelas corrediça rodoviária,
- 7 são itens não família:
  - 000533 – complemento dianteiro esquerdo,
  - 000563 – conjunto coluna sino,
  - 000565 – conjunto fechamento traseiro esquerdo,
  - 000566 – conjunto aro de roda dianteiro esquerdo,
  - 000567 – conjunto aro de roda traseiro esquerdo,
  - 000568 – contraventamentos lateral esquerdo, e
  - 000584 – conjunto primeira janela rodoviária lado esquerdo.
- 2 são *templates:*
  - 000523 – *Template* Distribuição de Janelas, e
  - 000560 – *Template* Conjunto Estrutura Lateral Esquerda.

Tabela 6 - Relacionamento dos projetos utilizados *versus* configuração

		Configuração			
		Família de Peças	**Família de Conjuntos**	**Itens Não Família**	***Template***
Arquivo CAD	000537	**X**	-	-	-
	000540	**X**	-	-	-
	000544	**X**	-	-	-
	000539	-	**X**	-	-
	000569	**X**	-	-	-
	000527	**X**	-	-	-
	000533	-	-	**X**	-
	000563	-	-	**X**	-
	000565	-	-	**X**	-
	000566	-	-	**X**	-
	000567	-	-	**X**	-
	000568	-	-	**X**	-
	000584	-	-	**X**	-
	000523	-	-	-	**X**
	000560	-	-	-	**X**

Fonte: Autor, 2017

Todas os projetos de produto foram configurados conforme as orientações presentes na seção 3.2 e serão explanados na seção 5.5.

5.3 Limitações

O modelo apresentado limita-se à representação geométrica das peças e conjuntos. As geometrias não representam o projeto final do produto, visto que os projetos são de propriedade intelectual da empresa em questão.

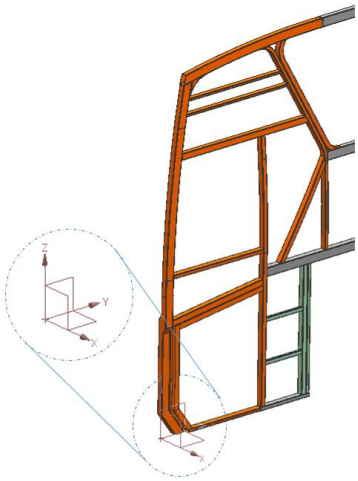
Devido à extensão dos códigos-fonte dos *templates*: conjunto estrutura lateral esquerda e distribuição de janelas lateral esquerda, os respectivos códigos não serão detalhados nesta dissertação, porém estão disponíveis no ANEXO A e no ANEXO B, respectivamente.

A escolha pela ordem de adição dos projetos no espaço 3D, por meio das funções descritas neste capítulo, deu-se pela facilidade em compreender a execução e comportamento de cada *template*.

5.4 Constantes de projeto

O posicionamento de uma geometria no espaço 3D de um *software* CAD depende dos valores dos eixos X, Y e Z. Conforme ilustrado na Figura 59, definiu-se como ponto de partida para os *templates* o ponto zero (x = 0, y = 0, z = 0) na extremidade frontal externa do complemento dianteiro.

Figura 59 - Definição do ponto zero de *template*



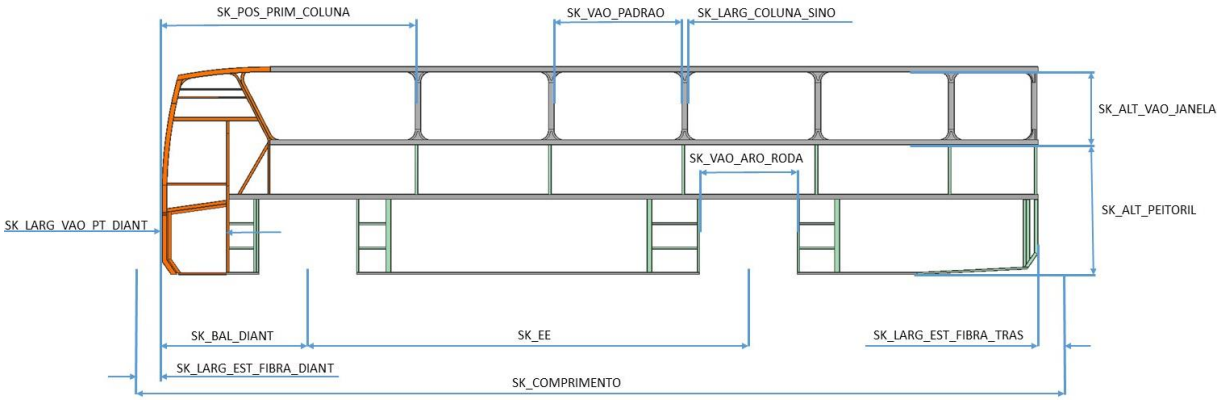
Fonte: Autor, 2017

O posicionamento no eixo X é positivo para a direção interna e negativo para a direção externa da carroceria. O eixo Y é positivo para a direção da traseira e negativo para a direção da dianteira. O eixo Z fica positivo para a direção do teto da carroceria e negativo para a direção do solo.

Os projetos estrutura lateral esquerda e conjunto de vidros laterais possuem importantes constantes de projeto. Uma constante de projeto é um valor fixo, ou um conjunto de valores pré-definidos, definido em fase de desenvolvimento de produto. As nomenclaturas dessas constantes foram padronizadas, iniciando com “SK_”. As constantes para ambos os *templates* estão representados na Figura 60, em que se pode fazer uma referência com a Tabela 7 - Descrição das constantes, a qual possui a definição e origem de cada uma das constantes.

Quanto à origem, são divididas em duas: “Constante de Projeto” referindo-se a medidas fixadas em fase de desenvolvimento de produto e “Variável Conforme Opção de Venda” quando o comportamento pode ser variável, conforme solicitação do cliente.

Figura 60 - Definições de constantes de projeto



Fonte: Autor, 2017

As constantes de projeto são essenciais para a criação dos *templates*. Elas servem como base para todos os cálculos de posicionamento e geometria, como, por exemplo, pode-se obter o valor da largura total da carroceria por meio do cálculo SK_COMPRIMENTO - SK_LARG_EST_FIBRA_DIANT - SK_LARG_EST_FIBRA_TRAS.

Tabela 7 - Descrição das constantes

Constante	Definição	Origem
SK_ALT_PEITORIL	Altura do peitoril	Constante de projeto
SK_ALT_PERF_BAGUETE_LAT	Altura do perfil baguete lateral	Constante de projeto
SK_ALT_VAO_JANELA	Altura do vão da janela	Constante de projeto
SK_BAL_DIANT	Balanço dianteiro	Variável conforme opção de venda
SK_COMPRIMENTO	Comprimento total do veículo	Variável conforme opção de venda
SK_EE	Entre-eixos	Variável conforme opção de venda
SK_ESP_EST_TRAS	Largura da estrutura traseira	Constante de projeto
SK_JANELA	Modelo de janelas	Variável conforme opção de venda
SK_LARG_COLUNA_SINO	Largura da coluna sino	Constante de projeto
SK_LARG_EST_FIBRA_DIANT	Largura da estrutura dianteira	Constante de projeto
SK_LARG_EST_FIBRA_TRAS	Largura da estrutura traseira	Constante de projeto
SK_POS_PRIM_COLUNA	Posição da primeira coluna sino	Constante de projeto
SK_VAO_PADRAO_JAN	Largura padrão das janelas	Variável conforme opção de venda

Fonte: Autor, 2017

Outros valores menos importantes podem ser utilizados e definidos como constantes, como por exemplo: bitolas de tubos, largura de um módulo fixo da estrutura, posicionamento de peças, entre outros.

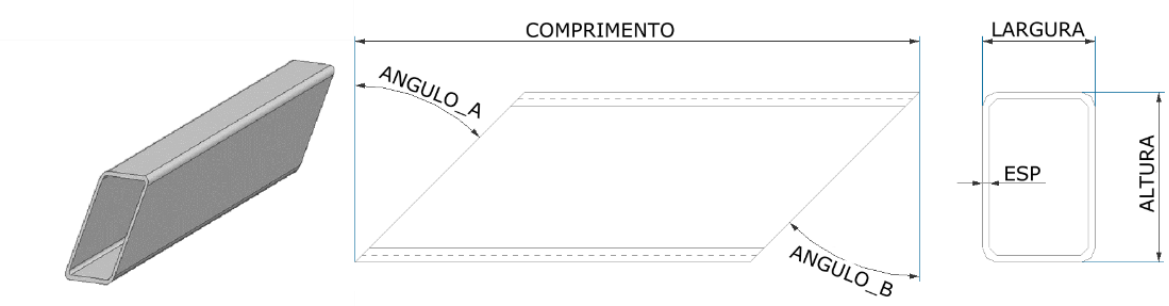
5.5 Configuração dos arquivos CAD

Conforme introduzido na seção 5.2.2, para o desenvolvimento deste estudo de caso foram necessários quinze projetos de produto. Todos os projetos foram configurados conforme as orientações presentes na seção 3.2 e serão explanados nas próximas subseções.

5.5.1 Família de tubo retangular

A família de tubo retangular com código 000527, representada na Figura 61, está classificada com tipo de arquivo CAD – Item Família de Peça (seção 3.3.1) – com configuração apenas do tipo *Part Family* (seção 3.4.1).

Figura 61 - Família de tubo retangular



Fonte: Autor, 2017

Figura 62 - Configuração do tipo *Part Family* para família de tubo retangular

	A	B	C	D	E	F
1	***DB_PART_NO***	***OS_PART_NAME***	***DB_PART_REV***	***DB_PART_DESC***	***DB_PART_NAME***	***DB_PART_TYPE***
2	000585	000585	1	TB_60x40x2.5x45x45x500	TB_60x40x2.5x45x45x500	C9_item_mestrado

G	H	I	J	K	L
COMPRIMENTO	***ALTURA***	***LARGURA***	***ESP***	***ANGULO_A***	***ANGULO_B***
500	60	40	2.5	45	45

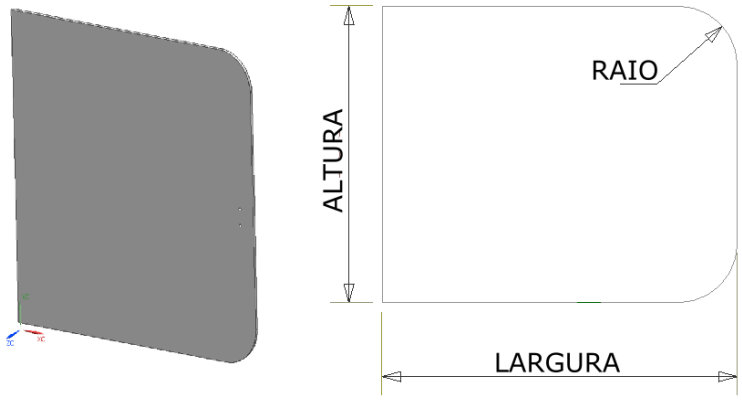
Fonte: Autor, 2017

A Figura 62 representa a configuração do tipo *Part Family* e os parâmetros necessários para que uma nova peça seja criada: COMPRIMENTO, ALTURA, LARGURA, ESP, ÂNGULO_A e ÂNGULO_B. As medidas informadas são em milímetros.

5.5.2 Família de vidro para janela rodoviária

A família de vidro para janela rodoviária, com código 000544 e representada na Figura 63, está classificada com tipo de arquivo CAD – Item Família de Peça (seção 3.3.1) – com configuração apenas do tipo *Part Family* (seção 3.4.1).

Figura 63 - Família de vidro para janela rodoviária



Fonte: Autor, 2017

A Figura 64 representa a configuração do tipo *Part Family* e os parâmetros necessários para que uma nova peça seja criada: ALTURA, LARGURA e RAIO. As medidas informadas são em milímetros.

Figura 64 - Configuração do tipo *Part Family* para família de vidro para janela rodoviária

	A	B	C	D	E	F
1	***DB_PART_NO***	***OS_PART_NAME***	***DB_PART_REV***	***DB_PART_NAME***	***DB_PART_DESC***	***DB_PART_TYPE***
2	000558	000558	001	VIDRO_JAN_ROD_935x960x115	VIDRO_JAN_ROD_935x960x115	C9_item_mestrado
3	005121	005121	001	VIDRO_JAN_ROD_855.0x946x115	VIDRO_JAN_ROD_855.0x946x115	C9_item_mestrado

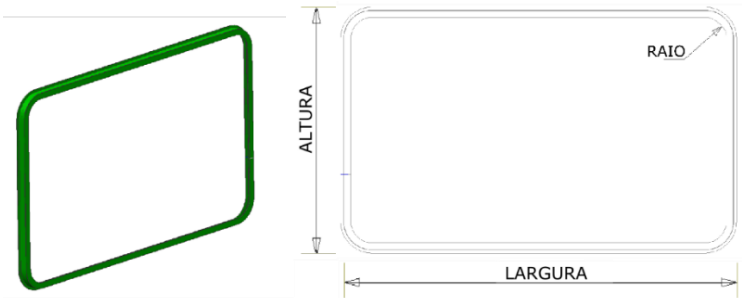
G	H	I
ALTURA	***LARGURA***	***RAIO***
935	960	115
855	946	115

Fonte: Autor, 2017

5.5.3 Família de perfil para janela rodoviária

A família de perfil para janela rodoviária, com código 000540, representada na Figura 65, está classificada com tipo de arquivo CAD – Item Família de Peça (seção 3.3.1) – com configuração apenas do tipo *Part Family* (seção 3.4.2).

Figura 65 - Família perfil janela rodoviária



Fonte: Autor, 2017

A Figura 66 ilustra a configuração do tipo *Part Family* e os parâmetros necessários para que uma nova peça seja criada: ALTURA, LARGURA e RAIO. As medidas informadas são em milímetros.

Figura 66 - Configuração do tipo *Part Family* para família de perfil para janela rodoviária

	A	B	C	D	E	F
1	***DB_PART_NO***	***OS_PART_NAME***	***DB_PART_REV***	***DB_PART_NAME***	***DB_PART_DESC***	***DB_PART_TYPE***
2	000557	000557	001	PERFIL_JAN_ROD_1000x1600x135	PERFIL_JAN_ROD_1000x1600x135	C9_item_mestrado

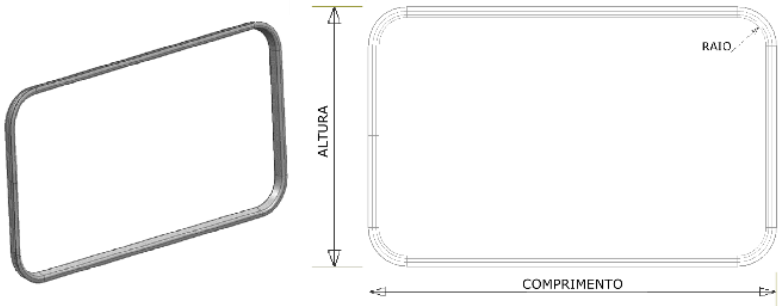
G	H	I
ALTURA	***LARGURA***	***RAIO***
1000	1600	135

Fonte: Autor, 2017

5.5.4 Família de borracha para janela rodoviária

A família de borracha para janela rodoviária, com código 000537, representada na Figura 67, está classificada com tipo de arquivo CAD – Item Família de Peça (seção 3.3.1) – com configuração apenas do tipo *Part Family* (seção 3.4.1).

Figura 67 - Família borracha janela rodoviária



Fonte: Autor, 2017

A Figura 68 ilustra a configuração do tipo *Part Family* e os parâmetros necessários para que uma nova peça seja criada: COMPRIMENTO, ALTURA e RAIO. As medidas informadas são em milímetros.

Figura 68 - Configuração do para família de borracha para janela rodoviária

	A	B	C	D	E	F
1	***DB_PART_NO***	***OS_PART_NAME***	***DB_PART_REV***	***DB_PART_NAME***	***DB_PART_DESC***	***DB_PART_TYPE***
2	000586	000586	001	MOLD_BORR_JAN_ROD_1665x1000x150	MOLD_BORR_JAN_ROD_1665x1000x150	C9_item_mestrado

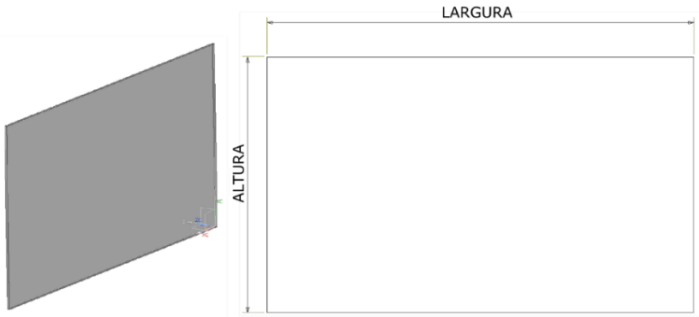
G	H	I
COMPRIMENTO	***ALTURA***	***RAIO***
1665	1000	150

Fonte: Autor, 2017

5.5.5 Família de vidro para janela colada

A família de vidro para janela colada, com código 000569, e representada na Figura 69, está classificada com tipo de arquivo CAD – Item Família de Peça (seção 3.3.1) – com configuração apenas do tipo *Part Family* (seção 3.4.1).

Figura 69 - Família vidro janela colada



Fonte: Autor, 2017

Na Figura 70 é ilustrada a configuração do tipo *Part Family* e os parâmetros necessários para que uma nova peça seja criada: ALTURA e LARGURA. As medidas informadas são em milímetros.

Figura 70 - Configuração do tipo *Part Family* para família de vidro para janela colada

	A	B	C	D	E	F
1	***DB_PART_NO***	***OS_PART_NAME***	***DB_PART_REV***	***DB_PART_NAME***	***DB_PART_DESC***	***DB_PART_TYPE***
2	005060	005060	001	VIDRO_JAN_COL_ROD_855.0x860x115	VIDRO_JAN_COL_ROD_855.0x860x115	C9_item_mestrado

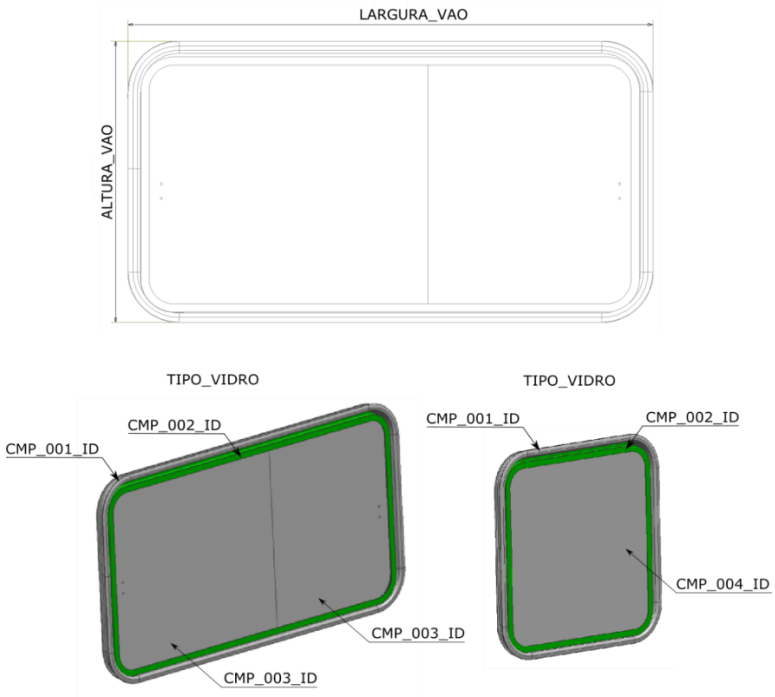
G	H
ALTURA	***LARGURA***
855	860

Fonte: Autor, 2017

5.5.6 Família de janelas corrediça rodoviária

A família de janelas corrediças, cujo código é o 000539, representada na Figura 71, é classificada com tipo de arquivo CAD – Família de Conjunto (seção 3.3.2) – com configurações do tipo *Part Family* (seção 3.4.1), *Assembly* (seção 3.4.2) e *Expressions* (seção 3.4.3).

Figura 71 - Família conjunto janela rodoviária



Fonte: Autor, 2017

A Figura 72 ilustra a configuração do tipo *Part Family* e os parâmetros necessários para que uma nova peça seja criada: ALTURA_VAO, LARGURA_VAO, TIPO_VIDRO, CMP_001_ID, CMP_002_ID, CMP_003_ID e CMP_004_ID. As medidas informadas são em milímetros.

Figura 72 - Configuração do tipo *Part Family* para família de janela rodoviária

	A	B	C	D	E	F
1	***DB_PART_NO***	***OS_PART_NAME***	***DB_PART_REV***	***DB_PART_NAME***	***DB_PART_DESC***	***DB_PART_TYPE***
2	000575	000575	1	CJ_JAN_CORR_ROD_1000x1600	CJ_JAN_CORR_ROD_1000x1600	C9_item_mestrado

	G	H	I	J	K	L	M
	ALTURA_VAO	***LARGURA_VAO***	***TIPO_VIDRO***	***CMP_001_ID***	***CMP_002_ID***	***CMP_003_ID***	***CMP_004_ID***
	1000	1600	0	000586	000557	000558	000569

Fonte: Autor, 2017

A Tabela 8 representa a configuração do tipo *Expressions*. Conforme coluna “Nome”, foram utilizados quatro conjuntos de expressões (CMP_001, CMP_002, CMP_003 e CMP_004).

Tabela 8 - Configuração do tipo *Expressions* para família de janela rodoviária

Nome			
CMP_001_A_X	CMP_002_A_X	CMP_003_A_X	CMP_004_A_X
CMP_001_A_Y	CMP_002_A_Y	CMP_003_A_Y	CMP_004_A_Y
CMP_001_A_Z	CMP_002_A_Z	CMP_003_A_Z	CMP_004_A_Z
CMP_001_DESC	CMP_002_DESC	CMP_003_DESC	CMP_004_DESC
CMP_001_ID	CMP_002_ID	CMP_003_ID	CMP_004_ID
CMP_001_ISASSEMBLY	CMP_002_ISASSEMBLY	CMP_003_ISASSEMBLY	CMP_004_ISASSEMBLY
CMP_001_NOME	CMP_002_NOME	CMP_003_NOME	CMP_004_NOME
CMP_001_PARAM	CMP_002_PARAM	CMP_003_PARAM	CMP_004_PARAM
CMP_001_PARAM_VALUE	CMP_002_PARAM_VALUE	CMP_003_PARAM_VALUE	CMP_004_PARAM_VALUE
CMP_001_SUPP	CMP_002_SUPP	CMP_003_SUPP	CMP_004_SUPP
CMP_001_X	CMP_002_X	CMP_003_X	CMP_004_X
CMP_001_Y	CMP_002_Y	CMP_003_Y	CMP_004_Y
CMP_001_Z	CMP_002_Z	CMP_003_Z	CMP_004_Z

Fonte: Autor, 2017

Conforme coluna “*Component Name*”, da Figura 73, a configuração do tipo *Assembly* foi efetuada renomeando os quatro componentes da estrutura para CMP_001_ID, CMP_002_ID, CMP_003_ID e CMP_004_ID.

Figura 73 - Configuração do tipo *Assembly* para família de janela rodoviária

Fonte: Autor, 2017

Conforme introduzido na seção 3.3.2, o projeto classificado com tipo de arquivo CAD Família de Conjunto e com configurações do tipo *Assembly*, necessita da criação de um projeto na aplicação Automator, sendo que esse processo está descrito a seguir, seção 5.5.7.

5.5.7 Desenvolvimento família de janelas corrediça

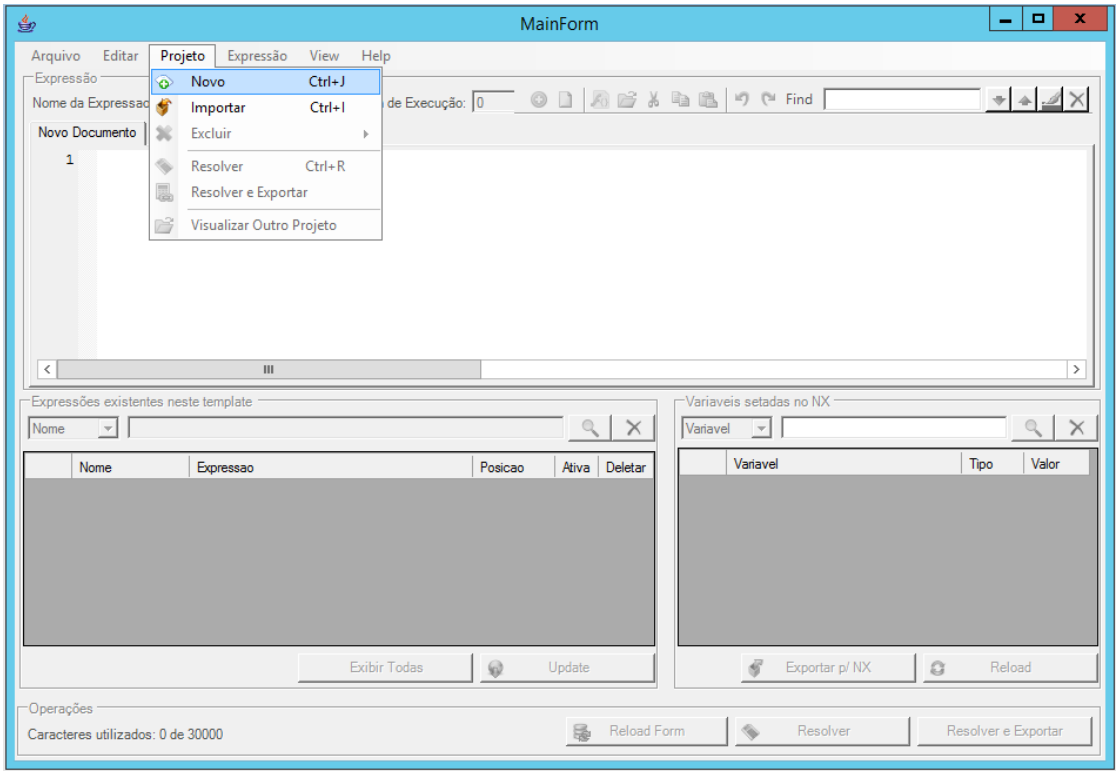
Projetos classificado com tipo de arquivo CAD – Família de Conjunto – e com configurações do tipo *Assembly* necessitam da criação de um projeto na aplicação Automator,

porém esse processo é simplificado em comparação com os projetos classificados com o tipo de arquivo CAD *template*.

A diferença está em que projeto de família de conjunto não necessita posicionar o componente no espaço 3D, logo as expressões “CMP_xxx_X”, “CMP_xxx_Y”, “CMP_xxx_Z”, “CMP_xxx_A_X”, “CMP_xxx_A_Y” e “CMP_xxx_A_Z” não necessitam ser criadas.

Após as configurações (*Expressions*, *Part Family* e *Assembly*), é necessário criar o projeto na aplicação Automator, conforme ilustrado na Figura 74.

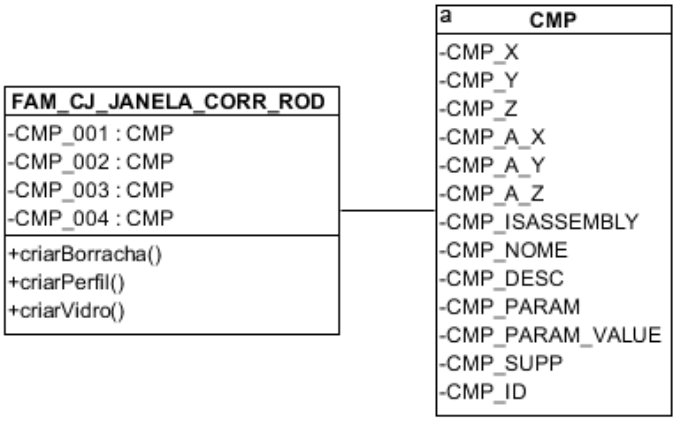
Figura 74 - Criação do projeto na aplicação Automator



Fonte: Autor, 2017

Para entendimento do projeto “família de janelas corrediça rodoviária”, foi criado um diagrama de classes de especificação, conforme ilustrado na Figura 75. O diagrama possui duas classes: CMP e FAM_CJ_JANELA_CORR_ROD. A classe CMP refere-se aos atributos necessários para o comportamento do componente, conforme seção 3.4.3. A classe FAM_CJ_JANELA_CORR_ROD representa os quatro componentes que estão na estrutura de produto e os métodos que interagem com esses componentes.

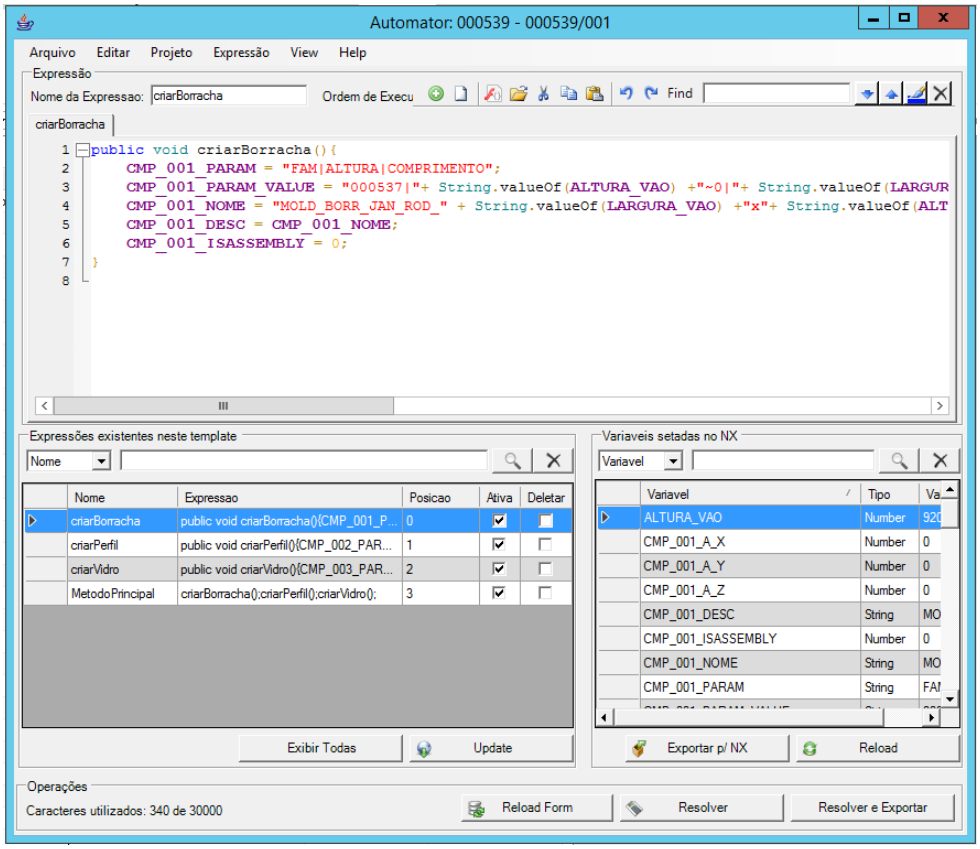
Figura 75 - Diagrama de classes família de janela corrediça



Fonte: Autor, 2017

A Figura 76 exibe a transformação do diagrama de classes (Figura 75) nas fórmulas desenvolvidas na aplicação Automator.

Figura 76 - Projeto criado na aplicação Automator



Fonte: Autor, 2017

Na sequência, serão descritas as fórmulas pela ordem de execução dos métodos, ou seja, “criarBorracha”, “criarPerfil” e “criarVidro”. Todos os componentes possuem criação

automática pela aplicação CustomNX. É possível identificar devido ao preenchimento das expressões “CMP_xxx_PARAM” e “CMP_xxx_PARAM_VALUE”.

O primeiro método a ser executado é o “criarBorracha”. Conforme ilustrado na Figura 77, a primeira linha indica o nome do método e, da linha dois até a linha seis, são definições do comportamento do componente. A origem dos valores FAM, ALTURA e COMPRIMENTO são preenchidos observando-se os parâmetros para criação da peça 000537, conforme

Figura 67 e seção 3.4.3.

Figura 77 - Método criar borracha janela rodoviária

```
criarBorracha
1  public void criarBorracha(){
2      CMP_001_PARAM = "FAM|ALTURA|COMPRIMENTO";
3      CMP_001_PARAM_VALUE = "000537|"+ String.valueOf(ALTURA_VAO) +"~0|"+ String.valueOf(LARGURA_VAO)+"~0";
4      CMP_001_NOME = "MOLD_BORR_JAN_ROD_" + String.valueOf(LARGURA_VAO) +"x"+ String.valueOf(ALTURA_VAO);
5      CMP_001_DESC = CMP_001_NOME;
6      CMP_001_ISASSEMBLY = 0;
7  }
8
```

Fonte: Autor, 2017

O segundo método a ser executado é o “criarPerfil”. Conforme ilustrado na Figura 78, a primeira linha indica o nome do método e, da linha dois até a linha seis, são definições do comportamento do componente. Os valores FAM, ALTURA, LARGURA e RAIO são preenchidos observando os parâmetros para criação da peça 000540, conforme Figura 65 e seção 3.4.3.

Figura 78 - Método criar perfil janela rodoviária

```
criarPerfil
1  public void criarPerfil(){
2      CMP_002_PARAM = "FAM|ALTURA|LARGURA|RAIO";
3      CMP_002_PARAM_VALUE = "000540|"+ String.valueOf(ALTURA_VAO) +"~0|"+ String.valueOf(LARGURA_VAO)+ "~0|135";
4      CMP_002_NOME = "VIDRO_JAN_ROD_" + String.valueOf(ALTURA_VAO) +"x"+ String.valueOf(LARGURA_VAO)+"x135";
5      CMP_002_DESC = CMP_002_NOME;
6      CMP_002_ISASSEMBLY = 0;
7  }
```

Fonte: Autor, 2017

O terceiro método a ser executado é o “criarVidro”. Conforme ilustrado na Figura 79, a primeira linha indica o nome do método e, da linha dois até a linha trinta, são definições do comportamento dos componentes. Neste método é possível observar dois componentes: “003” e “004”. O componente “003” refere-se ao tipo de janela corrediça (com dois vidros) e o componente “004” refere-se ao tipo de janela corrediça com vidro fixo (com apenas um vidro). Os valores FAM, ALTURA, LARGURA e RAIO são preenchidos observando-se os parâmetros para criação das peças 000544 e 000569, conforme Figura 63 e Figura 69 e seção 3.4.3.

Figura 79 - Método criar vidros

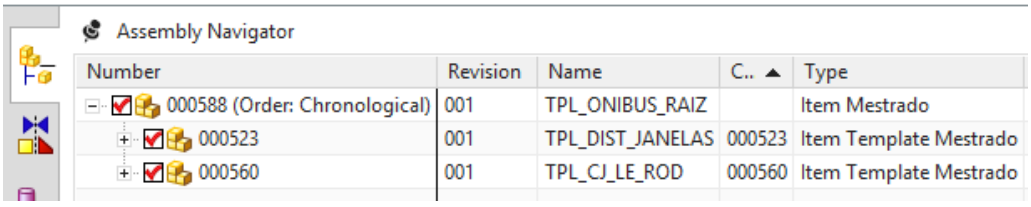
```
criarVidro
1  public void criarVidro(){
2      CMP_003_PARAM = "0";
3      CMP_003_PARAM_VALUE = "";
4      CMP_003_NOME = "";
5      CMP_003_DESC = CMP_003_NOME;
6      CMP_003_ISASSEMBLY = 0;
7      CMP_003_SUPP = 0;
8
9      CMP_004_PARAM = "";
10     CMP_004_PARAM_VALUE = "";
11     CMP_004_NOME = "";
12     CMP_004_DESC = CMP_004_NOME;
13     CMP_004_ISASSEMBLY = 0;
14     CMP_004_SUPP = 0;
15
16     if(TIPO_VIDRO == 0){
17         CMP_003_PARAM = "FAM|ALTURA|LARGURA|RAIO";
18         CMP_003_PARAM_VALUE = "000544|"+ String.valueOf(ALTURA_VAO - 65)  +"~0|"+ String.valueOf(LARGURA_VAO* 0.55) + "~0|115";
19         CMP_003_NOME = "VIDRO_JAN_ROD_" + String.valueOf(ALTURA_VAO - 65) +"x"+ String.valueOf(Math.round(LARGURA_VAO* 0.55))+"x115";
20         CMP_003_DESC = CMP_003_NOME;
21         CMP_003_ISASSEMBLY = 0;
22         CMP_003_SUPP = 1;
23     } else if(TIPO_VIDRO == 1){
24         CMP_004_PARAM = "FAM|ALTURA|LARGURA|RAIO";
25         CMP_004_PARAM_VALUE = "000569|"+ String.valueOf(ALTURA_VAO - 65)  +"~0|"+ String.valueOf(LARGURA_VAO - 60) + "~0|115";
26         CMP_004_NOME = "VIDRO_JAN_FIXA_ROD_" + String.valueOf(ALTURA_VAO - 65) +"x"+ String.valueOf(Math.round(LARGURA_VAO - 60))+"x115";
27         CMP_004_DESC = CMP_004_NOME;
28         CMP_004_ISASSEMBLY = 0;
29         CMP_004_SUPP = 1;
30     }
31 }
```

Fonte: Autor, 2017

5.6 Criação dos *templates*

A Figura 80 exibe a estrutura de produto montada para os *templates*: *Template* Distribuição de Janelas, código 000523 e *Template* Conjunto Estrutura Lateral Esquerda, código 000560.

Figura 80 - Estrutura de *templates*



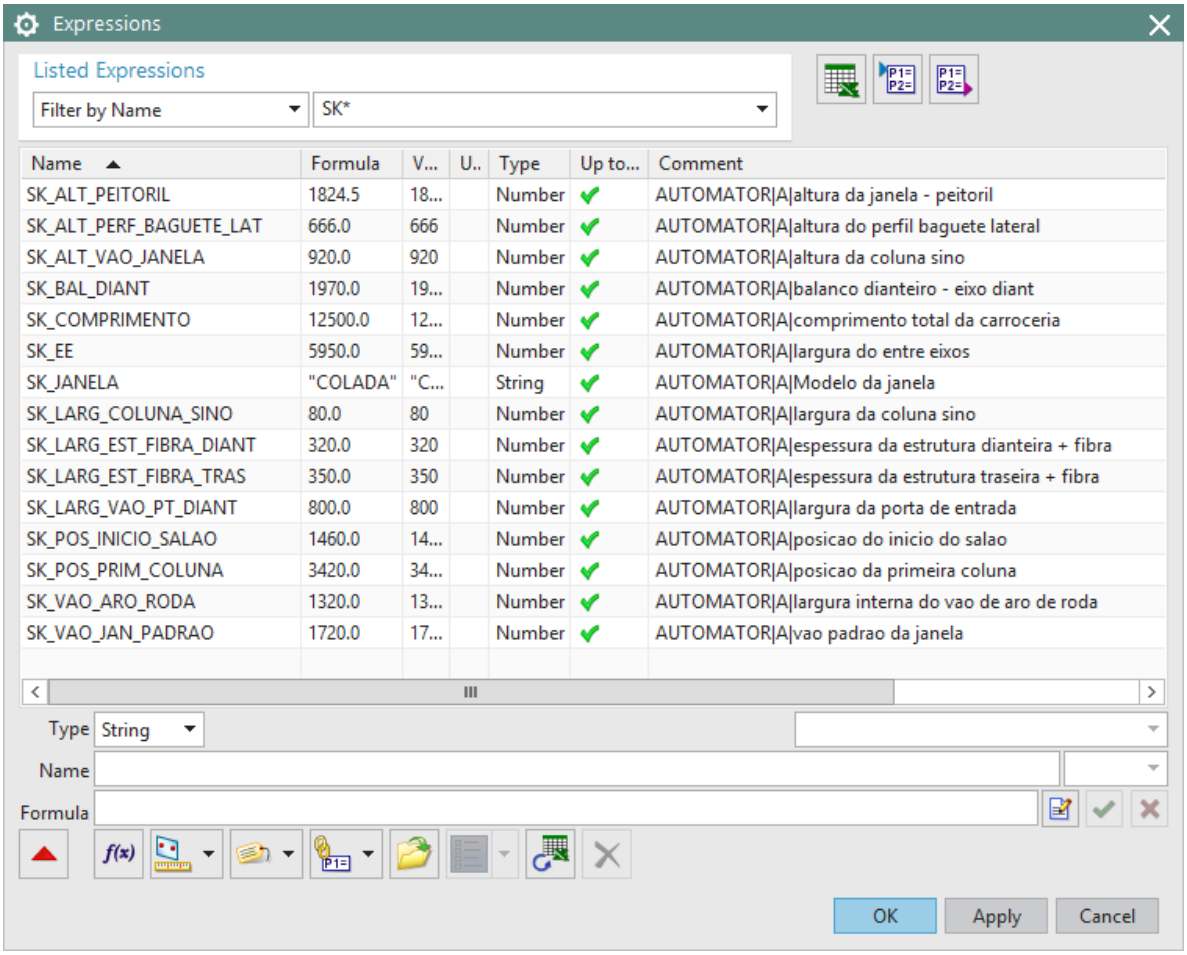
Fonte: Autor, 2017

5.6.1 Constantes de projeto

Após criar os *templates*, é necessário criar as constantes de projeto no *software* NX. Para ambos *templates*, foram criadas quinze constantes: SK_ALT_PEITORIL, SK_ALT_PERF_BAGUETE_LAT, SK_ALT_VAO_JANELA, SK_BAL_DIANT, SK_COMPRIMENTO, SK_EE, SK_JANELA, SK_LARG_COLUNA_SINO,

SK_LARG_EST_FIBRA_DIAN, SK_LARG_EST_FIBRA_TRAS, SK_LARG_VAO_PT_DIANT, SK_POS_INICIO_SALAO, SK_POS_PRIM_COLUNA, SK_VAO_ARO_RODA e SK_VAO_PADRAO_JAN, conforme exibe a Figura 81.

Figura 81 - Criação das constantes de projeto



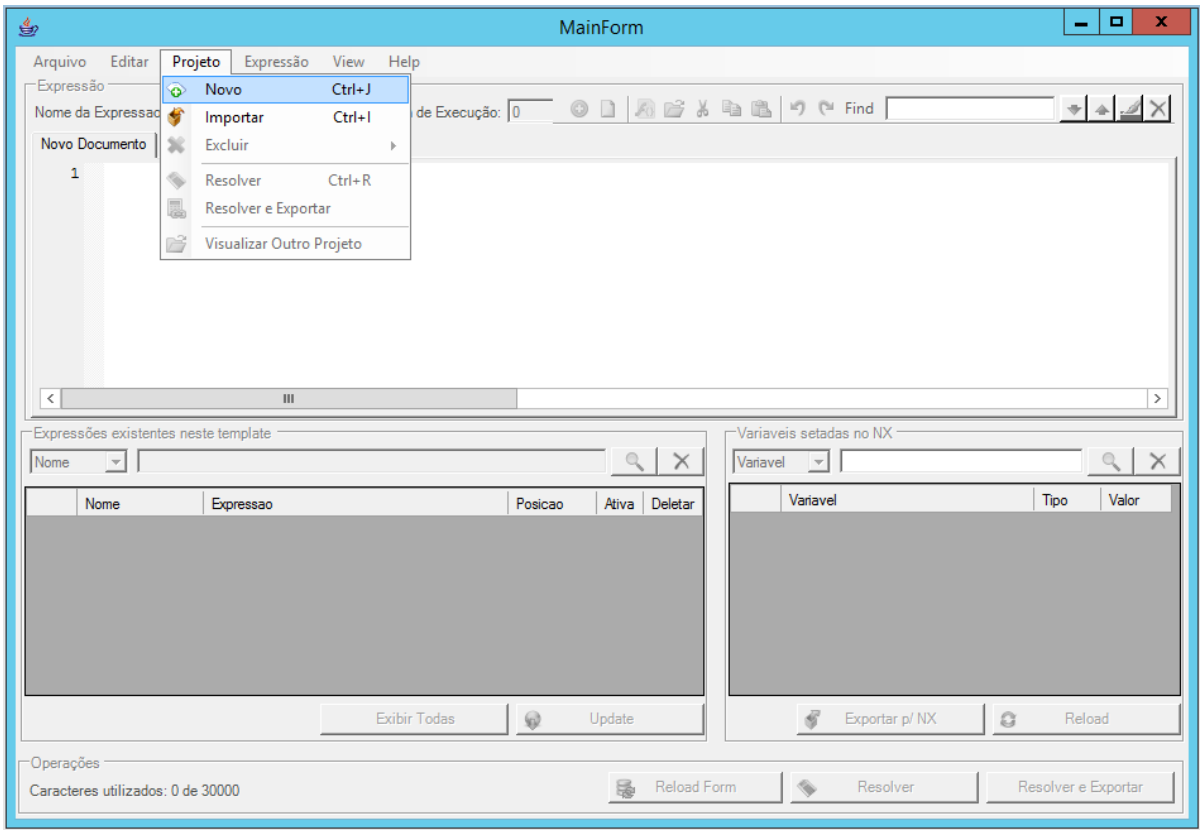
Fonte: Autor, 2017

Na próxima seção será exemplificada a criação destes *templates* na aplicação Automator.

5.6.2 Criação dos templates na aplicação Automator

Nesta etapa cria-se um projeto na aplicação Automator, para cada um dos dois *templates*. A Figura 82 exemplifica a criação do projeto na aplicação Automator.

Figura 82 - Criação do projeto na aplicação Automator

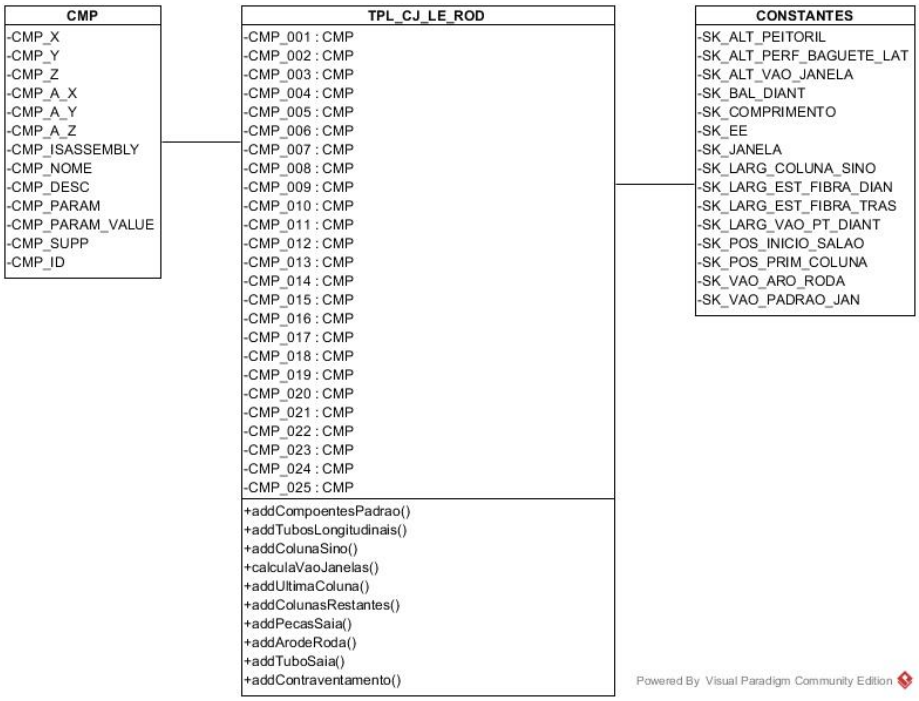


Fonte: Autor, 2017

5.7 Desenvolvimento *template* conjunto estrutura lateral esquerda

Para entendimento do desenvolvimento do *template* estrutura lateral esquerda foi criado um diagrama de classes de especificação, conforme exemplificado na Figura 83. O diagrama possui três classes: CMP, TPL_CJ_LE_ROD e CONSTANTES. A classe CMP refere-se aos atributos necessários para o comportamento do componente, conforme seção 3.4.3. A classe CONSTANTES refere-se às constantes de projeto, explicadas na seção 5.4. A classe TPL_CJ_LE_ROD representa os vinte e cinco componentes e os métodos que interagem com esses componentes.

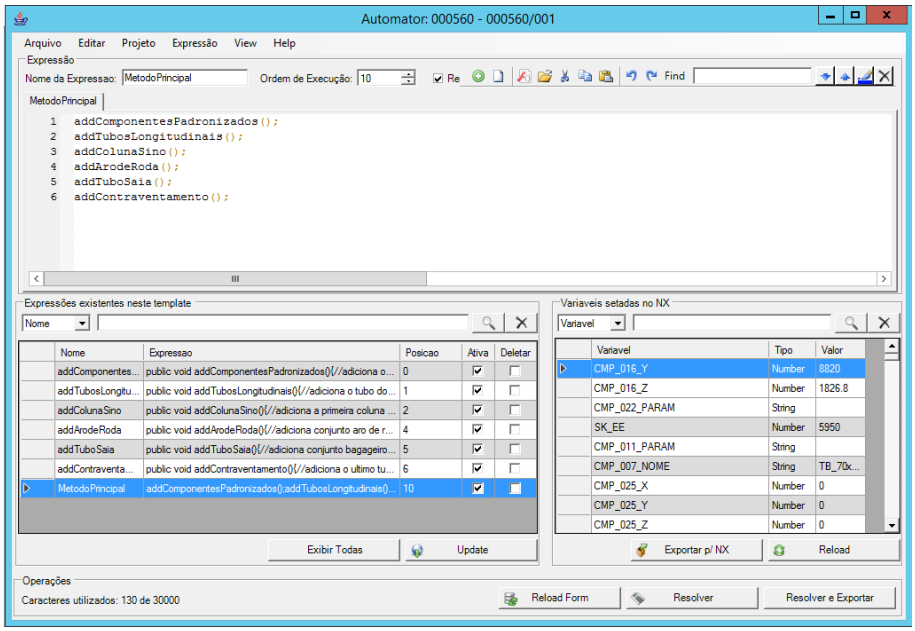
Figura 83 - Diagrama de classes *template* conjunto estrutura lateral esquerda



Fonte: Autor, 2017

A Figura 84 exibe a transformação do diagrama de classes nas equações desenvolvidas na aplicação Automator.

Figura 84 - Projeto criado na aplicação Automator



Fonte: Autor, 2017

As próximas sessões serão divididas pela ordem de execução dos métodos, ou seja, “addComponentesPadronizados”, “addTubosLongitudinais”, “addColunaSino”, “addArodeRoda”, “addTubosSaia” e “addContraventamento”. Os códigos-fonte estão disponíveis no ANEXO A.

5.7.1 Adicionando componentes padronizados

O primeiro método a ser executado é o “addComponentesPadronizados”. Neste método são definidos a posição e o código do complemento dianteiro.

A Figura 85 exibe o resultado da regra aplicando a configuração do pedido referência, em que o código do componente será o “000533”, complemento dianteiro esquerdo, e será adicionado no ponto zero do *template*, conforme coordenadas informadas X=0, Y=0, Z=0. É possível identificar que as expressões CMP_001_PARAM e CMP_001_PARAM_VALUE estão sem valores, pois não se trata de uma criação automática, o código já está sendo inferido diretamente na expressão CMP_001_ID. Para facilitar o entendimento, nas próximas sessões esta figura será representada em forma de tabela. As medidas informadas são em milímetros.

Figura 85 - Resultado da regra componentes padronizados

Variaveis setadas no NX

Variavel | CMP_001

Variavel	Tipo	Valor
CMP_001_Z	Number	0
CMP_001_Y	Number	0
CMP_001_X	Number	0
CMP_001_SUPP	Number	1
CMP_001_PARAM_VALUE	String	
CMP_001_PARAM	String	
CMP_001_NOME	String	
CMP_001_ISASSEMBLY	Number	0
CMP_001_ID	String	000533
CMP_001_DESC	String	
CMP_001_A_Z	Number	0
CMP_001_A_Y	Number	0
CMP_001_A_X	Number	0

Fonte: Autor, 2017

A Figura 86 exibe o componente “000533” adicionado ao *template*. Como apenas um método foi executado, o *template* possui somente este componente no espaço 3D.

Figura 86 - Complemento dianteiro adicionado no *template*

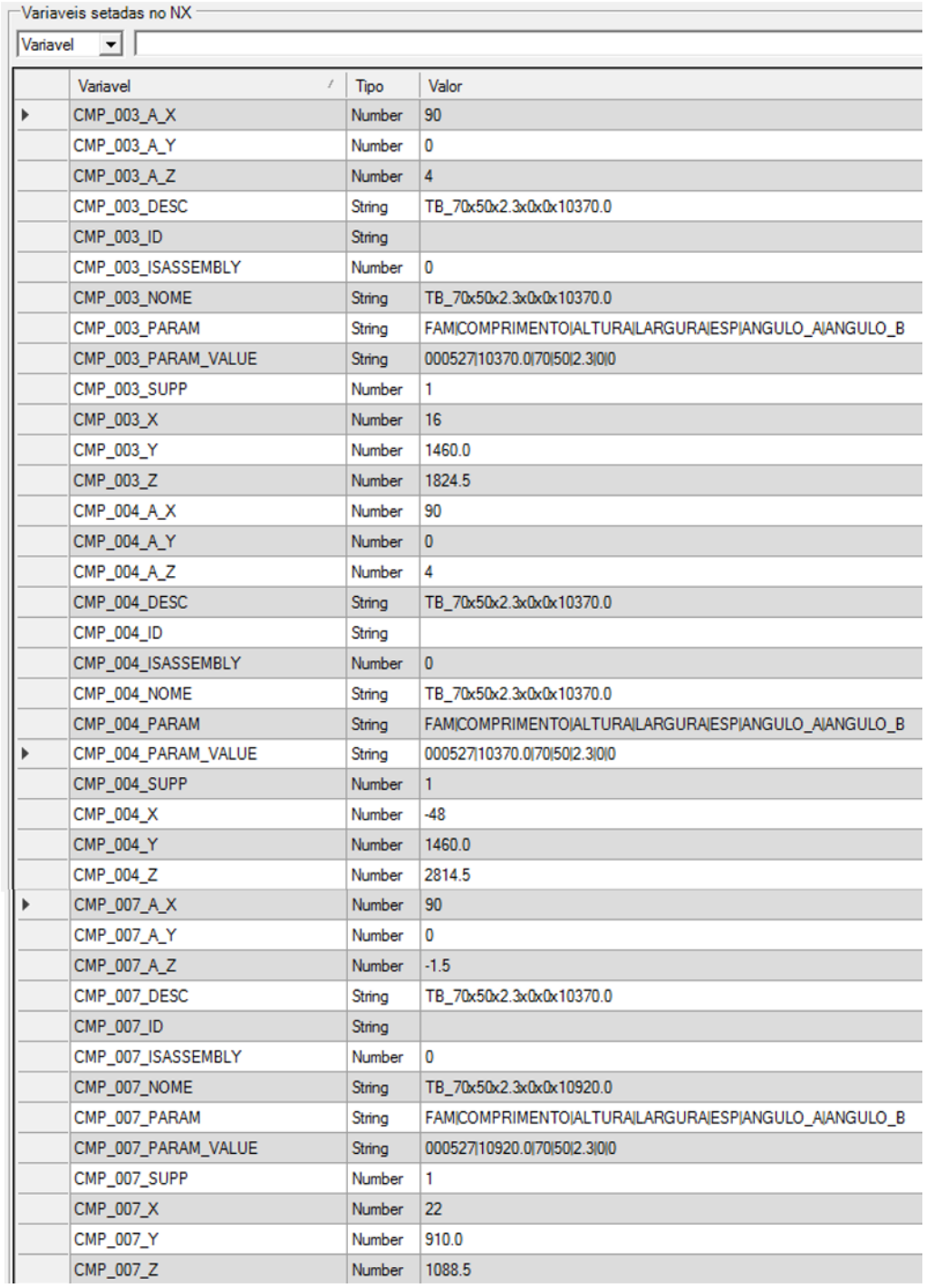
Fonte: Autor, 2017

5.7.2 Adicionando tubos longitudinais

O segundo método a ser executado é o "addTubosLongitudinais". Conforme representado na Figura 88, este método possui três componentes, todos com criação automática pela aplicação CustomNX. É possível identificar devido ao preenchimento das expressões "**CMP_xxx_PARAM**" e "**CMP_xxx_PARAM_VALUE**". Este método utiliza-se das constantes de projeto para definir a posição e comprimento de cada um dos tubos longitudinais. A origem dos valores **COMPRIMENTO**, **ALTURA**, **LARGURA**, **ESP**, **ÂNGULO_A** e **ÂNGULO_B** são preenchidos observando os parâmetros para criação da peça 000527, conforme Figura 61 e seção 3.4.3.

A Figura 87 exibe o resultado da regra aplicando a configuração do pedido referência. É possível identificar que os componentes "003", "004" e "007" já possuem um valor determinado para seu posicionamento no espaço 3D e as regras para criação automática através da aplicação CustomNX. As medidas informadas são em milímetros.

Figura 87 - Resultado da regra tubos longitudinais



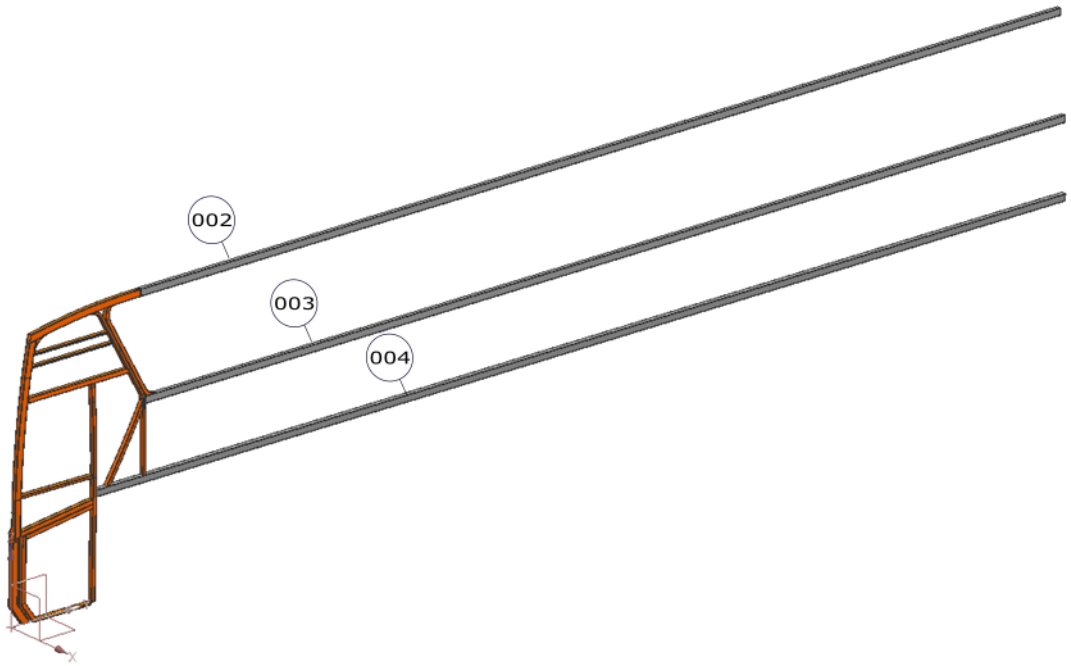
Variaveis setadas no NX

Variavel

Variavel	Tipo	Valor
CMP_003_A_X	Number	90
CMP_003_A_Y	Number	0
CMP_003_A_Z	Number	4
CMP_003_DESC	String	TB_70x50x2.3x0x0x10370.0
CMP_003_ID	String	
CMP_003_ISASSEMBLY	Number	0
CMP_003_NOME	String	TB_70x50x2.3x0x0x10370.0
CMP_003_PARAM	String	FAM|COMPRIMENTO|ALTURA|LARGURA|ESP|ANGULO_A|ANGULO_B
CMP_003_PARAM_VALUE	String	000527|10370.0|70|50|2.3|0|0
CMP_003_SUPP	Number	1
CMP_003_X	Number	16
CMP_003_Y	Number	1460.0
CMP_003_Z	Number	1824.5
CMP_004_A_X	Number	90
CMP_004_A_Y	Number	0
CMP_004_A_Z	Number	4
CMP_004_DESC	String	TB_70x50x2.3x0x0x10370.0
CMP_004_ID	String	
CMP_004_ISASSEMBLY	Number	0
CMP_004_NOME	String	TB_70x50x2.3x0x0x10370.0
CMP_004_PARAM	String	FAM|COMPRIMENTO|ALTURA|LARGURA|ESP|ANGULO_A|ANGULO_B
CMP_004_PARAM_VALUE	String	000527|10370.0|70|50|2.3|0|0
CMP_004_SUPP	Number	1
CMP_004_X	Number	-48
CMP_004_Y	Number	1460.0
CMP_004_Z	Number	2814.5
CMP_007_A_X	Number	90
CMP_007_A_Y	Number	0
CMP_007_A_Z	Number	-1.5
CMP_007_DESC	String	TB_70x50x2.3x0x0x10370.0
CMP_007_ID	String	
CMP_007_ISASSEMBLY	Number	0
CMP_007_NOME	String	TB_70x50x2.3x0x0x10920.0
CMP_007_PARAM	String	FAM|COMPRIMENTO|ALTURA|LARGURA|ESP|ANGULO_A|ANGULO_B
CMP_007_PARAM_VALUE	String	000527|10920.0|70|50|2.3|0|0
CMP_007_SUPP	Number	1
CMP_007_X	Number	22
CMP_007_Y	Number	910.0
CMP_007_Z	Number	1088.5

Fonte: Autor, 2017

Figura 88 - Tubos longitudinais adicionados no *template*



Fonte: Autor, 2017

A Figura 88 exibe os três componentes adicionados ao *template*. Após o segundo método executado, tem-se os quatro componentes no espaço 3D.

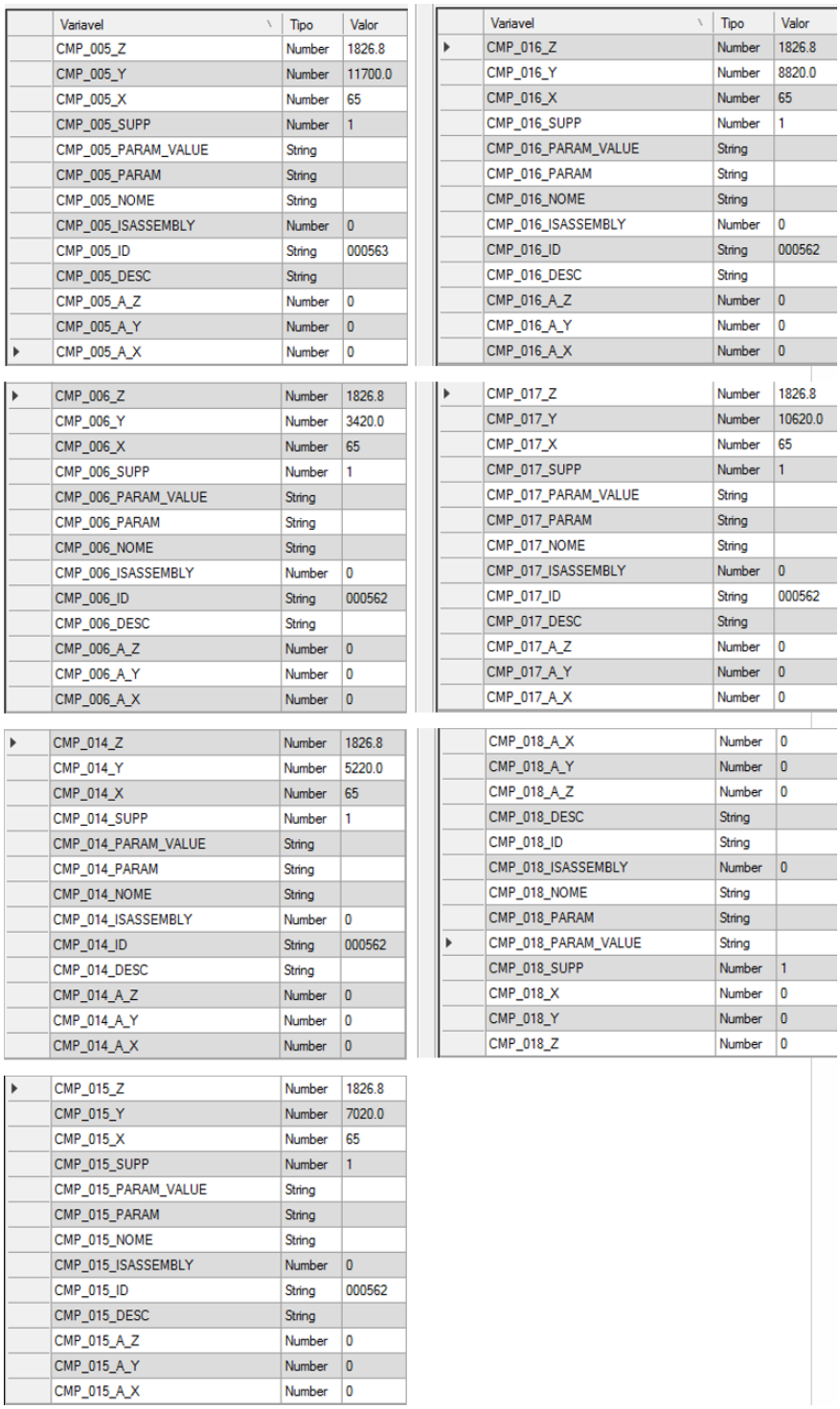
5.7.3 Adicionando colunas sino

O terceiro método a ser executado é o “addColunaSino”. A quantidade de coluna sino no projeto de estrutura lateral depende do comprimento total da carroceria. Para este estudo, utilizou-se um limite de até sete colunas sinos. Este método utiliza-se das constantes de projeto para definir a posição de cada uma das colunas sinos.

A Figura 89 exibe o resultado da regra aplicando a configuração do pedido referência. É possível identificar os sete blocos de expressões e que os componentes “005”, “006”, “014”, “015”, “016” e “017” já possuem um valor determinado para seu posicionamento no espaço 3D. É possível identificar que todas as expressões CMP_xxx_PARAM e CMP_xxx_PARAM_VALUE estão sem valores, pois não se trata de uma criação automática, o código já está sendo inferido diretamente na expressão CMP_xxx_ID. O componente “018” possui todos os valores nulos, isso significa que ele não será adicionado ao 3D, pois a regra

para sua utilização não foi satisfeita, ou seja, dos sete componentes disponíveis para utilização no *template*, somente seis serão adicionados ao 3D. As medidas informadas são em milímetros.

Figura 89 - Resultado da regra coluna sino



Variavel	Tipo	Valor
CMP_005_Z	Number	1826.8
CMP_005_Y	Number	11700.0
CMP_005_X	Number	65
CMP_005_SUPP	Number	1
CMP_005_PARAM_VALUE	String	
CMP_005_PARAM	String	
CMP_005_NOME	String	
CMP_005_ISASSEMBLY	Number	0
CMP_005_ID	String	000563
CMP_005_DESC	String	
CMP_005_A_Z	Number	0
CMP_005_A_Y	Number	0
CMP_005_A_X	Number	0

Variavel	Tipo	Valor
CMP_016_Z	Number	1826.8
CMP_016_Y	Number	8820.0
CMP_016_X	Number	65
CMP_016_SUPP	Number	1
CMP_016_PARAM_VALUE	String	
CMP_016_PARAM	String	
CMP_016_NOME	String	
CMP_016_ISASSEMBLY	Number	0
CMP_016_ID	String	000562
CMP_016_DESC	String	
CMP_016_A_Z	Number	0
CMP_016_A_Y	Number	0
CMP_016_A_X	Number	0

▸	CMP_006_Z	Number	1826.8
	CMP_006_Y	Number	3420.0
	CMP_006_X	Number	65
	CMP_006_SUPP	Number	1
	CMP_006_PARAM_VALUE	String	
	CMP_006_PARAM	String	
	CMP_006_NOME	String	
	CMP_006_ISASSEMBLY	Number	0
	CMP_006_ID	String	000562
	CMP_006_DESC	String	
	CMP_006_A_Z	Number	0
	CMP_006_A_Y	Number	0
	CMP_006_A_X	Number	0

▸	CMP_017_Z	Number	1826.8
	CMP_017_Y	Number	10620.0
	CMP_017_X	Number	65
	CMP_017_SUPP	Number	1
	CMP_017_PARAM_VALUE	String	
	CMP_017_PARAM	String	
	CMP_017_NOME	String	
	CMP_017_ISASSEMBLY	Number	0
	CMP_017_ID	String	000562
	CMP_017_DESC	String	
	CMP_017_A_Z	Number	0
	CMP_017_A_Y	Number	0
	CMP_017_A_X	Number	0

▸	CMP_014_Z	Number	1826.8
	CMP_014_Y	Number	5220.0
	CMP_014_X	Number	65
	CMP_014_SUPP	Number	1
	CMP_014_PARAM_VALUE	String	
	CMP_014_PARAM	String	
	CMP_014_NOME	String	
	CMP_014_ISASSEMBLY	Number	0
	CMP_014_ID	String	000562
	CMP_014_DESC	String	
	CMP_014_A_Z	Number	0
	CMP_014_A_Y	Number	0
	CMP_014_A_X	Number	0

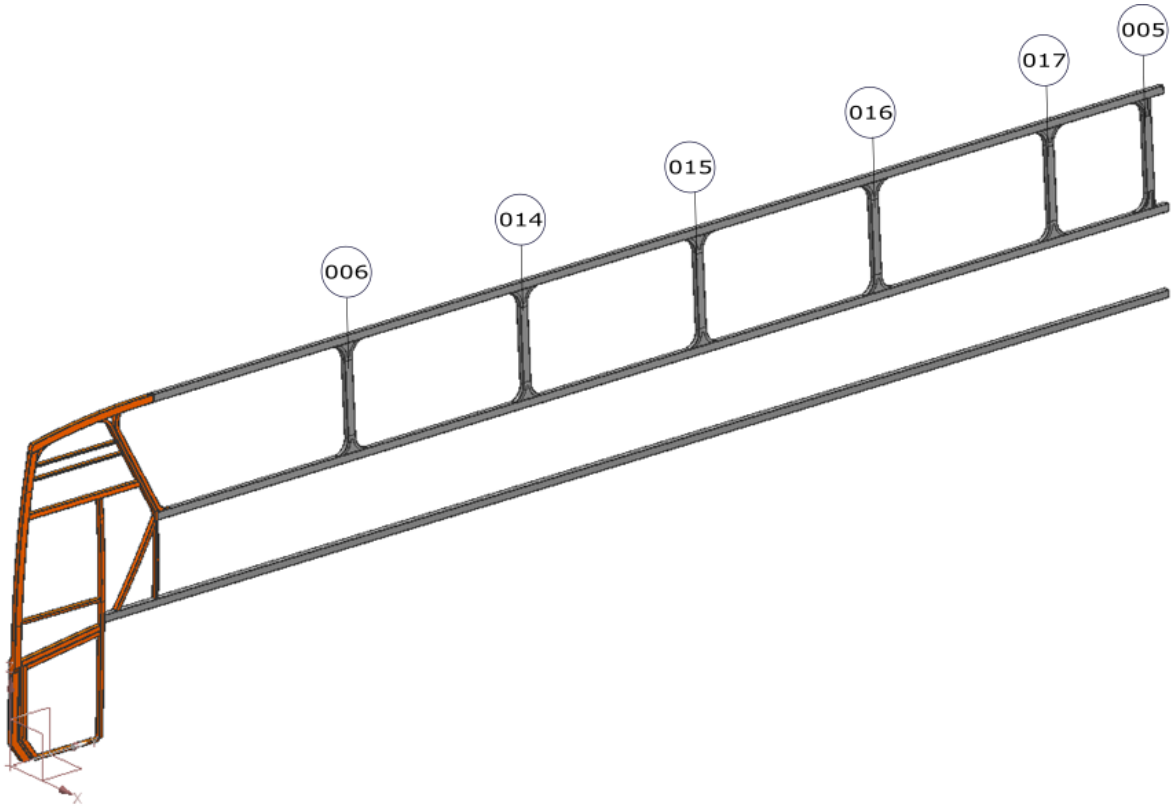
	CMP_018_A_X	Number	0
	CMP_018_A_Y	Number	0
	CMP_018_A_Z	Number	0
	CMP_018_DESC	String	
	CMP_018_ID	String	
	CMP_018_ISASSEMBLY	Number	0
	CMP_018_NOME	String	
	CMP_018_PARAM	String	
▸	CMP_018_PARAM_VALUE	String	
	CMP_018_SUPP	Number	1
	CMP_018_X	Number	0
	CMP_018_Y	Number	0
	CMP_018_Z	Number	0

▸	CMP_015_Z	Number	1826.8
	CMP_015_Y	Number	7020.0
	CMP_015_X	Number	65
	CMP_015_SUPP	Number	1
	CMP_015_PARAM_VALUE	String	
	CMP_015_PARAM	String	
	CMP_015_NOME	String	
	CMP_015_ISASSEMBLY	Number	0
	CMP_015_ID	String	000562
	CMP_015_DESC	String	
	CMP_015_A_Z	Number	0
	CMP_015_A_Y	Number	0
	CMP_015_A_X	Number	0

Fonte: Autor, 2017

A Figura 90 exibe os seis componentes adicionados ao *template*. Após o terceiro método executado, tem-se os nove componentes no espaço 3D.

Figura 90 - Colunas sinos adicionadas no *template*



Fonte: Autor, 2017

5.7.4 Adicionando conjuntos de aro de roda

O quarto método a ser executado é o “addArodeRoda”. Este método utiliza-se das constantes de projeto para definir a posição de cada um dos projetos.

A Figura 91 exibe o resultado da regra aplicando a configuração do pedido referência. É possível identificar dois componentes – “009” e “010” – que já possuem um valor determinado para seu posicionamento no espaço 3D. É possível identificar que todas as expressões CMP_xxx_PARAM e CMP_xxx_PARAM_VALUE estão sem valores, pois não trata-se de uma criação automática, o código já está sendo inferido diretamente na expressão CMP_009_ID e CMP_010_ID. As medidas informadas são em milímetros.

Figura 91 - Resultado da regra aro de roda

Variaveis setadas no NX

Variavel: CMP_009

Variavel	Tipo	Valor
CMP_009_Z	Number	0
CMP_009_Y	Number	1970
CMP_009_X	Number	23
CMP_009_SUPP	Number	1
CMP_009_PARAM_VALUE	String	
CMP_009_PARAM	String	
CMP_009_NOME	String	
CMP_009_ISASSEMBLY	Number	0
CMP_009_ID	String	000566
CMP_009_DESC	String	
CMP_009_A_Z	Number	0
CMP_009_A_Y	Number	0
CMP_009_A_X	Number	0

Variaveis setadas no NX

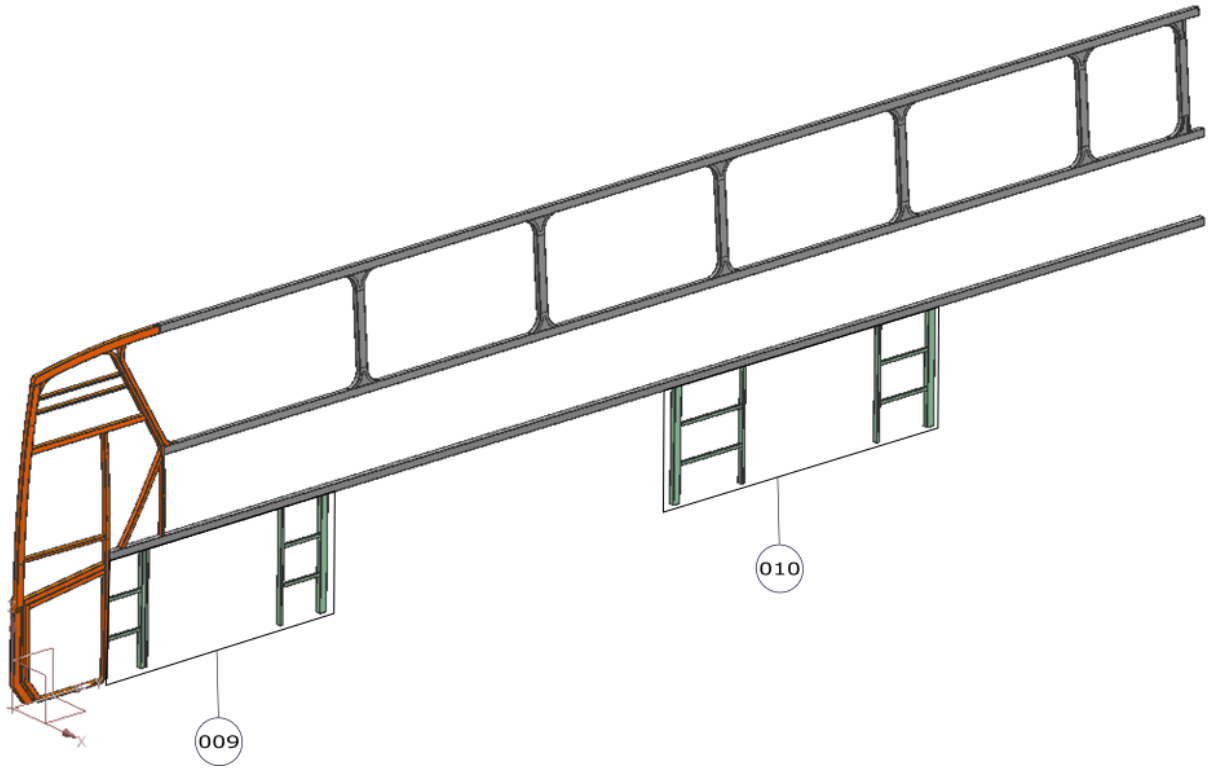
Variavel: CMP_010

Variavel	Tipo	Valor
CMP_010_Z	Number	0
CMP_010_Y	Number	7920
CMP_010_X	Number	48
CMP_010_SUPP	Number	1
CMP_010_PARAM_VALUE	String	
CMP_010_PARAM	String	
CMP_010_NOME	String	
CMP_010_ISASSEMBLY	Number	0
CMP_010_ID	String	000567
CMP_010_DESC	String	
CMP_010_A_Z	Number	0
CMP_010_A_Y	Number	0
CMP_010_A_X	Number	0

Fonte: Autor, 2017

A Figura 92 exibe os dois componentes adicionados ao *template*. Após o terceiro método executado, tem-se os onze componentes no espaço 3D.

Figura 92 - Aros de rodas adicionados no *template*



Fonte: Autor, 2017

5.7.5 Adicionando tubos saia

O quinto método a ser executado é o “addTubosSaia”. Este método utiliza-se das constantes de projeto para definir a posição de cada um dos projetos.

Conforme representado na Figura 93, este método possui três componentes, dois com criação automática pela aplicação CustomNX e um com inferência direta do código. É possível identificar devido ao preenchimento das expressões “CMP_xxx_PARAM” e “CMP_xxx_PARAM_VALUE”. As medidas informadas são em milímetros.

A origem dos valores COMPRIMENTO, ALTURA, LARGURA, ESP, ÂNGULO_A e ÂNGULO_B são preenchidos observando os parâmetros para criação da peça 000527, conforme Figura 61 e seção 3.4.1.

Figura 93 - Resultado da regra tubos saia

Variavel	Tipo	Valor
CMP_008_Z	Number	0
CMP_008_Y	Number	11830
CMP_008_X	Number	45
CMP_008_SUPP	Number	1
CMP_008_PARAM_VALUE	String	
CMP_008_PARAM	String	
CMP_008_NOME	String	
CMP_008_ISASSEMBLY	Number	0
CMP_008_ID	String	000565
CMP_008_DESC	String	
CMP_008_A_Z	Number	0
CMP_008_A_Y	Number	0
CMP_008_A_X	Number	0

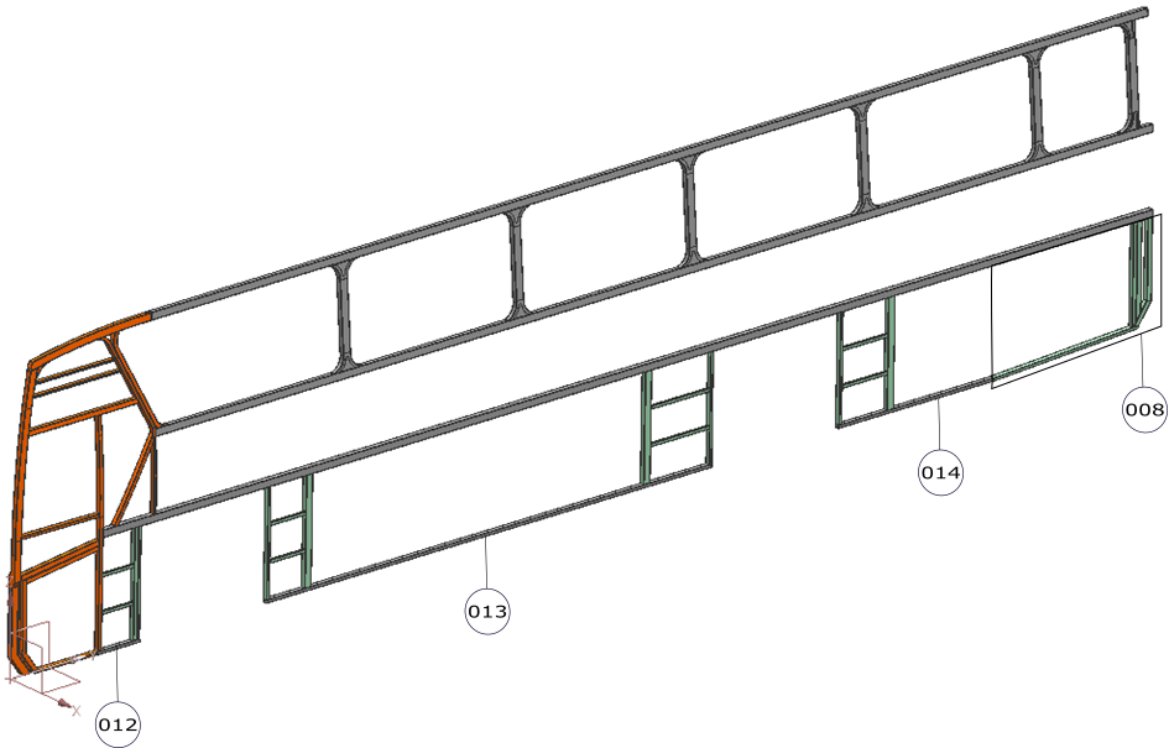
Variavel	Tipo	Valor
CMP_012_Z	Number	0
CMP_012_Y	Number	1310.0
CMP_012_X	Number	0
CMP_012_SUPP	Number	1
CMP_012_PARAM_VALUE	String	000527\440.0\30~0\47~0\1.95~0\0\0
CMP_012_PARAM	String	FAM\COMPRIMENTO\ALTURA\LARGURA\ESP\ANGULO_A\ANGULO_B
CMP_012_NOME	String	TB_30x47x1.95x0x0x440.0
CMP_012_ISASSEMBLY	Number	0
CMP_012_ID	String	
CMP_012_DESC	String	TB_30x47x1.95x0x0x440.0
CMP_012_A_Z	Number	0
CMP_012_A_Y	Number	0
CMP_012_A_X	Number	-90

Variavel	Tipo	Valor
CMP_019_Z	Number	30
CMP_019_Y	Number	8580.0
CMP_019_X	Number	0
CMP_019_SUPP	Number	1
CMP_019_PARAM_VALUE	String	000527\1616.0\30~0\47~0\1.95~0\0\0
CMP_019_PARAM	String	FAM\COMPRIMENTO\ALTURA\LARGURA\ESP\ANGULO_A\ANGULO_B
CMP_019_NOME	String	TB_30x47x1.95x0x0x4630.0
CMP_019_ISASSEMBLY	Number	0
CMP_019_ID	String	
CMP_019_DESC	String	TB_30x47x1.95x0x0x4630.0
CMP_019_A_Z	Number	0
CMP_019_A_Y	Number	0
CMP_019_A_X	Number	90

Fonte: Autor, 2017

A Figura 94 exibe os três componentes adicionados ao *template*. Após o quinto método executado, tem-se os quatorze componentes no espaço 3D.

Figura 94 - Tubos da saia adicionados no *template*



Fonte: Autor, 2017

5.7.6 Adicionando contraventamentos

O sexto método a ser executado é o “addContraventamento”. Este método utiliza-se das constantes de projeto para definir a posição de cada um dos projetos. A quantidade de contraventamentos em um projeto de estrutura lateral depende do comprimento total da carroceria. Para este estudo, utilizou-se um limite de até sete contraventamentos.

A Figura 95 exibe o resultado da regra aplicando a configuração do pedido referência. É possível identificar seis componentes – “011”, “020”, “021”, “022”, “023” e “024” – que já possuem um valor determinado para seu posicionamento no espaço 3D. É possível identificar que todas as expressões CMP_xxx_PARAM e CMP_xxx_PARAM_VALUE estão sem valores, pois não trata-se de uma criação automática, o código já está sendo inferido diretamente nas expressões CMP_xxx_ID. As medidas informadas são em milímetros.

O componente “025” possui todos os valores nulos, significando que ele não será adicionado ao 3D, pois a regra para sua utilização não foi satisfeita, ou seja, dos sete componentes disponíveis para utilização no *template,* somente seis serão adicionados ao 3D.

Figura 95 - Resultado da regra contraventamentos

Variavel	Tipo	Valor
CMP_011_Z	Number	1088.5
CMP_011_Y	Number	11780.0
CMP_011_X	Number	23
CMP_011_SUPP	Number	1
CMP_011_PARAM_VALUE	String	
CMP_011_PARAM	String	
CMP_011_NOME	String	
CMP_011_ISASSEMBLY	Number	0
CMP_011_ID	String	000568
CMP_011_DESC	String	
CMP_011_A_Z	Number	0
CMP_011_A_Y	Number	0
CMP_011_A_X	Number	0

Variavel	Tipo	Valor
CMP_020_Z	Number	1088.5
CMP_020_Y	Number	3420.0
CMP_020_X	Number	23
CMP_020_SUPP	Number	1
CMP_020_PARAM_VALUE	String	
CMP_020_PARAM	String	
CMP_020_NOME	String	
CMP_020_ISASSEMBLY	Number	0
CMP_020_ID	String	000568
CMP_020_DESC	String	
CMP_020_A_Z	Number	0
CMP_020_A_Y	Number	0
CMP_020_A_X	Number	0

Variavel	Tipo	Valor
CMP_021_Z	Number	1088.5
CMP_021_Y	Number	5220.0
CMP_021_X	Number	23
CMP_021_SUPP	Number	1
CMP_021_PARAM_VALUE	String	
CMP_021_PARAM	String	
CMP_021_NOME	String	
CMP_021_ISASSEMBLY	Number	0
CMP_021_ID	String	000568
CMP_021_DESC	String	
CMP_021_A_Z	Number	0
CMP_021_A_Y	Number	0
CMP_021_A_X	Number	0

Variavel	Tipo	Valor
CMP_022_Z	Number	1088.5
CMP_022_Y	Number	7020.0
CMP_022_X	Number	23
CMP_022_SUPP	Number	1
CMP_022_PARAM_VALUE	String	
CMP_022_PARAM	String	
CMP_022_NOME	String	
CMP_022_ISASSEMBLY	Number	0
CMP_022_ID	String	000568
CMP_022_DESC	String	
CMP_022_A_Z	Number	0
CMP_022_A_Y	Number	0
CMP_022_A_X	Number	0

Variavel	Tipo	Valor
CMP_023_Z	Number	1088.5
CMP_023_Y	Number	8820.0
CMP_023_X	Number	23
CMP_023_SUPP	Number	1
CMP_023_PARAM_VALUE	String	
CMP_023_PARAM	String	
CMP_023_NOME	String	
CMP_023_ISASSEMBLY	Number	0
CMP_023_ID	String	000568
CMP_023_DESC	String	
CMP_023_A_Z	Number	0
CMP_023_A_Y	Number	0
CMP_023_A_X	Number	0

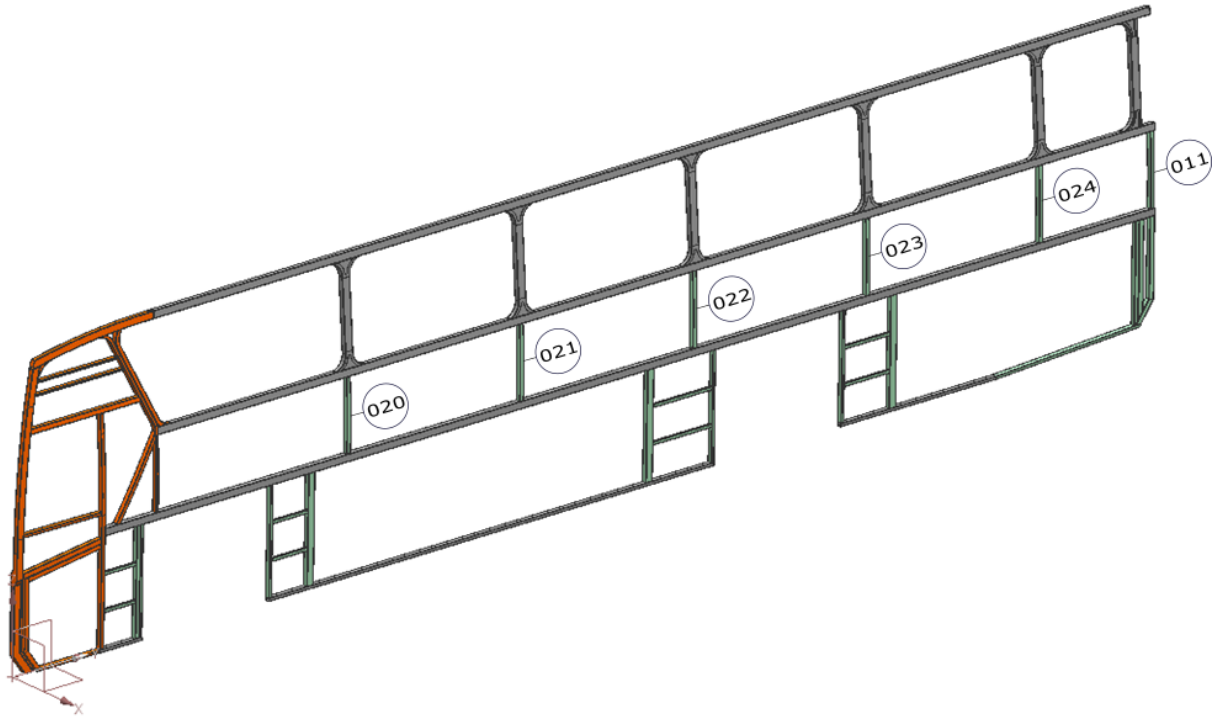
Variavel	Tipo	Valor
CMP_024_Z	Number	1088.5
CMP_024_Y	Number	10620.0
CMP_024_X	Number	23
CMP_024_SUPP	Number	1
CMP_024_PARAM_VALUE	String	
CMP_024_PARAM	String	
CMP_024_NOME	String	
CMP_024_ISASSEMBLY	Number	0
CMP_024_ID	String	000568
CMP_024_DESC	String	
CMP_024_A_Z	Number	0
CMP_024_A_Y	Number	0
CMP_024_A_X	Number	0

Variavel	Tipo	Valor
CMP_025_Z	Number	0.0
CMP_025_Y	Number	0.0
CMP_025_X	Number	0.0
CMP_025_SUPP	Number	1.0
CMP_025_PARAM_VALUE	String	
CMP_025_PARAM	String	
CMP_025_NOME	String	
CMP_025_ISASSEMBLY	Number	0.0
CMP_025_ID	String	
CMP_025_DESC	String	
CMP_025_A_Z	Number	0.0
CMP_025_A_Y	Number	0.0
CMP_025_A_X	Number	0.0

Fonte: Autor, 2017

A Figura 96 exibe os seis componentes adicionados ao *template*. Após o sexto método executado, tem-se os vinte componentes no espaço 3D.

Figura 96 - Contraventamentos adicionados no *template*



Fonte: Autor, 2017

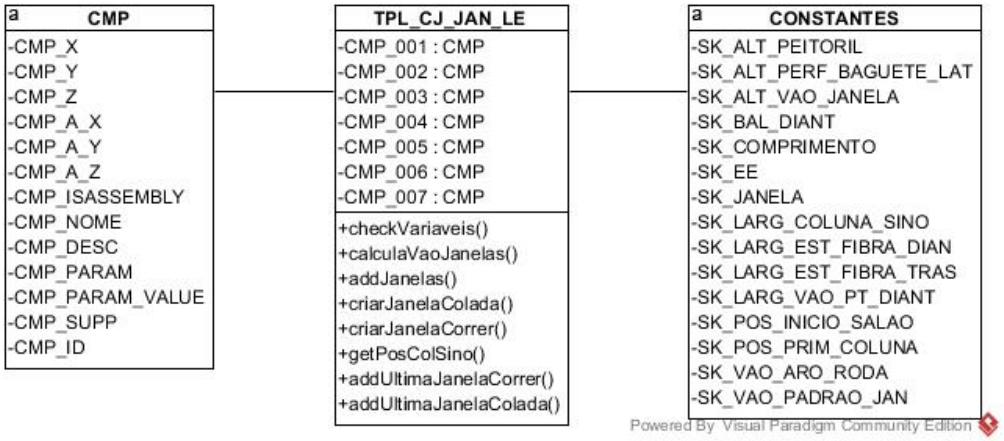
5.8 Desenvolvimento *template* distribuição de janelas lateral esquerda

Para entendimento do desenvolvimento do *template* de distribuição de janelas na lateral esquerda foi criado um diagrama de classes de especificação, ilustrado na Figura 97.

O diagrama possui três classes: CMP, TPL_CJ_JAN_LE e CONSTANTES. A classe CMP refere-se aos atributos necessários para o comportamento do componente, conforme seção 3.4.3. A classe CONSTANTES refere-se àss constantes de projeto, explicadas na seção 5.4. A classe TPL_CJ_JAN_LE representa os sete componentes e os métodos que interagem com esses componentes.

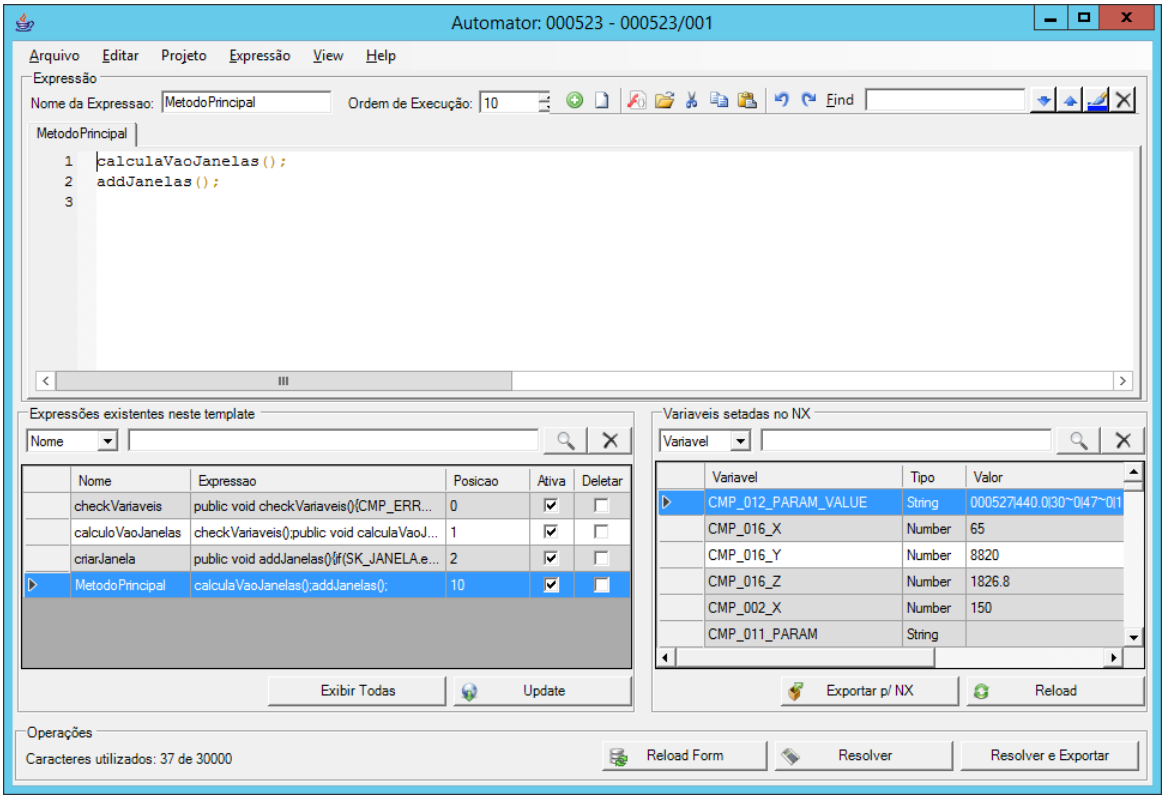
A Figura 98 exibe a transformação do diagrama de classes nas fórmulas desenvolvidas na aplicação Automator: método “calculaVaoJanelas” e “addJanelas”.

Figura 97 - Diagrama de classes *template* conjunto janelas lateral esquerda



Fonte: Autor, 2017

Figura 98 - Projeto criado na aplicação Automator



Fonte: Autor, 2017

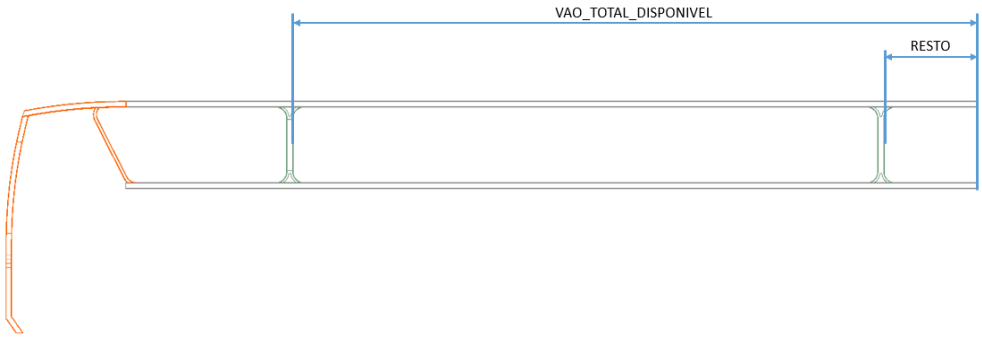
As próximas sessões serão divididas pela ordem de execução dos métodos, ou seja, “calculaVaoJanelas” e “addJanelas”. Os códigos-fonte estão disponíveis no ANEXO B.

5.8.1 Calculando o vão de janelas

O primeiro método a ser executado é o “calculaVaoJanelas”. Com base nas constantes de projeto, apresentadas na Figura 60, o método executa os cálculos necessários para identificar o vão total disponível para a distribuição das janelas, a quantidade total de janelas e o espaço restante no final da carroceria, onde uma janela padronizada deve ser adicionada.

A Figura 99 exemplifica o resultado do cálculo de vão de janelas. Com base no pedido referência, obteve-se 8330 mm de vão disponível (VAO_TOTAL_DISPONIVEL) e 1130 mm de espaço final restante (RESTO). Nenhum componente é adicionado ao 3D neste método, a figura é apenas representativa.

Figura 99 - Exemplo das constantes do vão de janelas



Fonte: Autor, 2017

A quantidade de janelas depende do comprimento total da carroceria além da largura padronizada de cada janela.

5.8.2 Adicionando as janelas

O segundo método a ser executado é o “addJanelas”. Este método utiliza-se das constantes de projeto, apresentadas na Figura 60, para definir a posição de cada um dos projetos. Para este estudo, utilizou-se uma configuração limite de até sete janelas.

A Figura 100 exibe o resultado da regra aplicando a configuração do pedido referência. É possível identificar seis componentes – “001”, “002”, “003”, “004”, “005” e “007” – que já possuem um valor determinado para seu posicionamento no espaço 3D. As medidas informadas são em milímetros.

O componente “006” possui todos os valores nulos. Isso significa que ele não será adicionado ao 3D, pois a regra para sua utilização não foi satisfeita, ou seja, dos sete componentes disponíveis para utilização no *template*, somente seis serão adicionados ao 3D.

Figura 100 - Resultado da regra adicionar janelas

Variavel	Tipo	Valor
CMP_001_Z	Number	1496.5
CMP_001_Y	Number	3801
CMP_001_X	Number	-50
CMP_001_SUPP	Number	1
CMP_001_PARAM_VALUE	String	
CMP_001_PARAM	String	
CMP_001_NOME	String	
CMP_001_ISASSEMBLY	Number	0
CMP_001_ID	String	000584
CMP_001_DESC	String	CJ_JAN_CORR_ROD_920.0x1720.0
CMP_001_A_Z	Number	0
CMP_001_A_Y	Number	0
CMP_001_A_X	Number	0

Variavel	Tipo	Valor
CMP_005_Z	Number	1824.5
CMP_005_Y	Number	10620
CMP_005_X	Number	150
CMP_005_SUPP	Number	1
CMP_005_PARAM_VALUE	String	000539|920.0~0|1720.0~0|0
CMP_005_PARAM	String	FAM|ALTURA_VAO|LARGURA_VAO|TIPO_VIDRO
CMP_005_NOME	String	CJ_JAN_CORR_ROD_920.0x1720.0
CMP_005_ISASSEMBLY	Number	1
CMP_005_ID	String	005122
CMP_005_DESC	String	CJ_JAN_CORR_ROD_920.0x1720.0
CMP_005_A_Z	Number	90
CMP_005_A_Y	Number	93
CMP_005_A_X	Number	0

Variavel	Tipo	Valor
CMP_002_Z	Number	1824.5
CMP_002_Y	Number	5220
CMP_002_X	Number	150
CMP_002_SUPP	Number	1
CMP_002_PARAM_VALUE	String	000539|920.0~0|1720.0~0|0
CMP_002_PARAM	String	FAM|ALTURA_VAO|LARGURA_VAO|TIPO_VIDRO
CMP_002_NOME	String	CJ_JAN_CORR_ROD_920.0x1720.0
CMP_002_ISASSEMBLY	Number	1
CMP_002_ID	String	005122
CMP_002_DESC	String	CJ_JAN_CORR_ROD_920.0x1720.0
CMP_002_A_Z	Number	90
CMP_002_A_Y	Number	93
CMP_002_A_X	Number	0

Variavel	Tipo	Valor
CMP_006_Z	Number	0
CMP_006_Y	Number	0
CMP_006_X	Number	0
CMP_006_SUPP	Number	0
CMP_006_PARAM_VALUE	String	
CMP_006_PARAM	String	
CMP_006_NOME	String	
CMP_006_ISASSEMBLY	Number	0
CMP_006_ID	String	
CMP_006_DESC	String	
CMP_006_A_Z	Number	0
CMP_006_A_Y	Number	0
CMP_006_A_X	Number	0

Variavel	Tipo	Valor
CMP_003_Z	Number	1824.5
CMP_003_Y	Number	7020
CMP_003_X	Number	150
CMP_003_SUPP	Number	1
CMP_003_PARAM_VALUE	String	000539|920.0~0|1720.0~0|0
CMP_003_PARAM	String	FAM|ALTURA_VAO|LARGURA_VAO|TIPO_VIDRO
CMP_003_NOME	String	CJ_JAN_CORR_ROD_920.0x1720.0
CMP_003_ISASSEMBLY	Number	1
CMP_003_ID	String	005122
CMP_003_DESC	String	CJ_JAN_CORR_ROD_920.0x1720.0
CMP_003_A_Z	Number	90
CMP_003_A_Y	Number	93
CMP_003_A_X	Number	0

Variavel	Tipo	Valor
CMP_007_Z	Number	1826.8
CMP_007_Y	Number	11620
CMP_007_X	Number	150
CMP_007_SUPP	Number	1
CMP_007_PARAM_VALUE	String	000539|920.0~0|920~0|1
CMP_007_PARAM	String	FAM|ALTURA_VAO|LARGURA_VAO|TIPO_VIDRO
CMP_007_NOME	String	CJ_JAN_CORR_ROD_920.0x920
CMP_007_ISASSEMBLY	Number	1
CMP_007_ID	String	005118
CMP_007_DESC	String	CJ_JAN_CORR_ROD_920.0x920
CMP_007_A_Z	Number	90
CMP_007_A_Y	Number	93
CMP_007_A_X	Number	0

Variavel	Tipo	Valor
CMP_004_Z	Number	1824.5
CMP_004_Y	Number	8820
CMP_004_X	Number	150
CMP_004_SUPP	Number	1
CMP_004_PARAM_VALUE	String	000539|920.0~0|1720.0~0|0
CMP_004_PARAM	String	FAM|ALTURA_VAO|LARGURA_VAO|TIPO_VIDRO
CMP_004_NOME	String	CJ_JAN_CORR_ROD_920.0x1720.0
CMP_004_ISASSEMBLY	Number	1
CMP_004_ID	String	005122
CMP_004_DESC	String	CJ_JAN_CORR_ROD_920.0x1720.0
CMP_004_A_Z	Number	90
CMP_004_A_Y	Number	93
CMP_004_A_X	Number	0

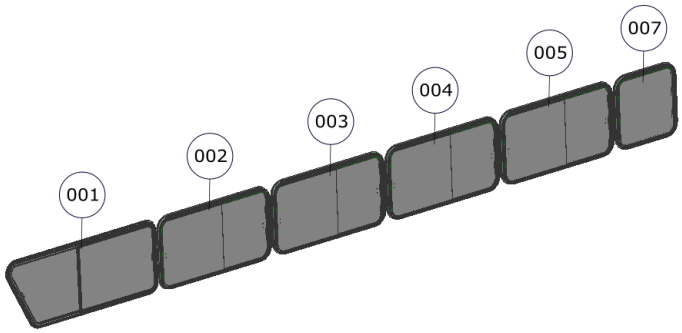
Fonte: Autor, 2017

Outra observação feita na Figura 100, é que este método possui cinco componentes com criação automática pela aplicação CustomNX, são eles: “002”, “003”, “004”, “005” e “007”. É

possível identificar devido ao preenchimento das expressões "**CMP_xxx_PARAM**" e "**CMP_xxx_PARAM_VALUE**". A origem dos valores **ALTURA_VAO**, **LARGURA_VAO** e **TIPO_VIDRO** é preenchida observando-se os parâmetros para criação da peça 000539, conforme Figura 63 e seção 3.4.3.

A Figura 96 exibe os seis componentes adicionados ao *template*. Após o segundo método executado, tem-se todos os componentes no espaço 3D.

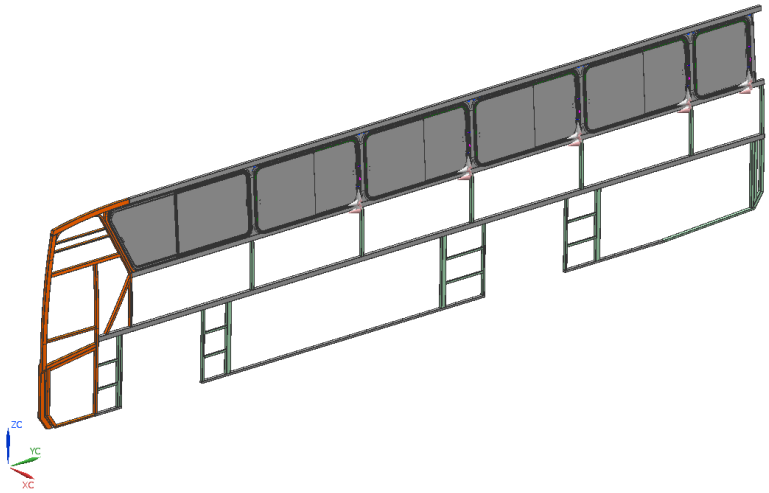
Figura 101 - Janelas adicionadas no *template*



Fonte: Autor, 2017

Após a conclusão e execução dos dois *templates*, *Template* Distribuição de Janelas e *Template* Conjunto Estrutura Lateral Esquerda, tem-se o modelo proposto completo, conforme representado na Figura 102.

Figura 102 - Resultado da união dos *templates*



Fonte: Autor, 2017

O capítulo subsequente é dedicado às conclusões e considerações finais acerca do trabalho desenvolvido, bem como sugestões para trabalhos futuros.

6 CONCLUSÃO

A presente pesquisa descreveu um conjunto de aplicações para apoiar o processo de desenvolvimento de produto, focando na modelagem do projeto 3D. A abordagem baseia-se no conceito de *templates* inteligentes, combinando configuração de produto e técnicas de automação de projetos.

A abordagem proposta obteve êxito na representação computacional do produto ônibus, fazendo a captura comportamental do produto por meio de *templates* e regras matemáticas.

O objetivo geral do presente trabalho constituiu-se em desenvolver uma aplicação que permita a automação de projetos 3D. Procurou-se validar o objetivo no estudo de caso, sendo que a aplicação foi utilizada em dois projetos de alta complexidade e com elevado número de componentes e interdependência. Pode-se observar que as aplicações são estáveis e possuem escalabilidade para empregar-se em toda a linha de produto da empresa e aplicada diretamente nos setores de Engenharia de Desenvolvimento e Engenharia de Produto, validando assim o objetivo geral deste trabalho.

Quanto aos objetivos específicos, conclui-se:

- A integração das aplicações com o *software* NX foi essencial para a obtenção dos resultados, sendo que o *software* forneceu todos os recursos CAD para automação dos projetos.
- Os modelos utilizados no estudo de caso representam a variedade de peças e conjuntos no dia a dia do setor de Engenharia de Produto e Desenvolvimento.
- Os *templates* de produto foram o elo de ligação entre a captura de conhecimento de engenharia de produto e as aplicações desenvolvidas.
- Entende-se que a metodologia desenvolvida torna os *templates* de produtos escaláveis, ou seja, pode-se escolher o nível de customização em que se deseja chegar. O *template* pode iniciar automatizando pequenas partes de um projeto e ir evoluindo para sua totalidade, ficando a cargo da manutenção dada pelo projetista.

Conclui-se que o objetivo geral e os objetivos específicos foram alcançados em sua totalidade, tornando a abordagem apta a ser aplicada na empresa estudada.

A aplicação Automator, integrada ao *software* NX, ajudou a fazer a construção das regras para cada *template* de produto. A linguagem de programação utilizada para construção

das regras é de fácil aprendizado e possui muito material para estudo disponível *on-line*. A criação das regras é flexível quanto a sua escrita, pois pode-se escrever um código simples, como os cálculos utilizados no *software* Excel, ou uma escrita complexa, da mesma forma que o código no desenvolvimento de *software*, por meio de classes e funções. A escolha fica a cargo do usuário.

A servidora de regras teve desempenho e estabilidade computacional suficiente para suportar a integração com a aplicação Automator e CustomNX. Os *webservices*, hospedados na servidora de regras, foram essenciais para a integração entre o *software* NX, banco de dados e cálculo de regras.

A aplicação CustomNX desempenhou um papel importante para integrar o *software* configurador de produto, o NX e as regras desenvolvidas para cada produto. A aplicação está desenvolvida para criar projetos com qualquer tamanho de estrutura de produto, não estando limitada apenas a peças ou conjuntos de apenas um nível de estrutura.

Esse conjunto de aplicações encurta a curva de aprendizado para novos engenheiros e projetistas, pois, após o *template* desenvolvido, qualquer usuário poderá aplicar as configurações e gerar novas versões de produtos, precisando apenas de um conjunto de características de venda ou um pedido referência.

Percebeu-se também que os *templates* não podem ser deixados de lado após o desenvolvimento do novo produto. Os *templates* devem ser periodicamente atualizados pela Engenharia de Produto. Dentre essas atualizações podem-se destacar alterações em normas, alterações nas regras de configuração de produto, alterações nos opcionais ofertados, alterações feitas em linha de produto e correção de erros de projeto.

Entende-se que a abordagem elimina problema de “ilhas de automação” (ambiente em que cada usuário possui uma maneira de automatizar seu dia a dia) gerando um ambiente colaborativo, no qual todos podem contribuir para que o produto seja cada vez mais automatizado.

Por fim, os testes demonstraram que os *templates* possuem inteligência CAD suficiente para diminuir o atravessamento dos pedidos dentro da Engenharia de Produto, disponibilizando com antecedência os projetos para as outras áreas.

6.1 Problemas encontrados no desenvolvimento

Em relação às dificuldades computacionais, houve uma curva de aprendizado alta na biblioteca NX Open, fazendo com que as integrações com o *software* NX fossem difíceis e tendo um tempo de desenvolvimento elevado.

Em relação ao produto, as dificuldades foram em prever, em fase de desenvolvimento, o comportamento do produto diante de todos os opcionais ofertados. Baseado na quantidade de opcionais de venda disponíveis para a carroceria estudada, pode-se chegar ao número de 3.500 configurações de estrutura lateral esquerda, 450.000 configurações de estrutura de teto e 4.700 configurações distintas de estrutura de base, sendo que somente na lateral esquerda pode-se chegar a um total de 600 componentes por lateral projetada.

Para demonstrar o conhecimento obtido no estudo de caso, os problemas foram em transformar as informações virtuais em textos, diagramas, imagens e tabelas.

Em relação à usabilidade da aplicação Automator, ficou um ponto negativo na falta de um recurso visual para analisar inconsistências de regras. Por exemplo, caso a nova regra adicionada entre em conflito com uma regra já existente. À medida que o *template* é atualizado, sua base de regras aumenta. Esse aumento de regras pode gerar uma dificuldade na gestão das regras, criando inconsistências. A forma textual de definir os parâmetros para criação automática pode se tornar crítica, pois podem ocorrer erros nos preenchimentos dos valores, gerando inconsistência nas informações.

6.2 Contribuições

Este trabalho teve contribuição para a empresa na qual se desenvolveu o estudo, sistematizando uma abordagem para apoiar o desenvolvimento de produto e fazendo a aquisição de conhecimento ao longo do ciclo de vida do produto.

Como principais contribuições, pode-se citar:

- Um modelo computacional, baseado em regras, que representa matematicamente o produto ônibus.

- Uma aplicação que, por meio de criação de regras, faz a aquisição de conhecimentos de projeto.

- Definição de constantes de projeto que podem ser utilizadas em qualquer outra aplicação ou projeto dentro da empresa.

- Maior precisão no posicionamento e tolerância geométrica de peças, devido aos parâmetros decimais informados em regras.

- Padronização de componentes devido à utilização em tempo real de projetos e acesso a uma única fonte de informação.

- Criação automática de projetos, diminuindo tempo e erros de projeto.

- Projeto de produto integrado em tempo real ao pedido de vendas, possibilitando a automação.

Entende-se que este trabalho também contribui para outros segmentos que possuam produtos customizados ou cujos produtos sofram variações dentro de um *range* de opcionais. Dentre esses segmentos, destacam-se indústria moveleira, indústria naval, implementos rodoviários e agrícolas, transportes e indústria civil.

6.3 Trabalhos futuros

As sugestões para trabalhos futuros baseiam-se na utilização de todo o sistema para criação do estudo de caso. Essas sugestões são de novos módulos para as aplicações desenvolvidas e podem ter grande potencial para a empresa estudada e para a aplicação em outras indústrias:

- Uma nova abordagem, baseada em regras, para seleção de matéria-prima de projeto. Verificação de matérias-primas disponíveis em estoque e que possam ser substituídas por outras que estão indisponíveis.

- Desenvolver um módulo que analise a configuração de produto aplicada ao *template* e identifique se esta configuração já foi aplicada anteriormente, evitando assim duplicação de grandes projetos.

- Como a abordagem já faz a criação de peças inexistentes, sugere-se uma ampliação da aplicação para gerar planos de processos automáticos, baseados na parametrização da família de produto.

- Criação de um módulo para introdução de inteligência artificial e redes neurais para que o sistema possa fazer a aprendizagem de regras automaticamente.

- Criação de um módulo integrado aos recursos CAD para parametrização do projeto detalhado (2D), fazendo o reuso das regras.

- Um módulo para representação gráfica da árvore de regras, interdependências, dependências e inconsistências de informações.

- Atualização da aplicação Automator para que as regras possam ser escritas em outras linguagens, como, por exemplo, CSharp, Javascript e VisualBasic, deixando a escolha da linguagem a cargo do usuário.

- Sugere-se ainda a criação de novas integrações para outros *softwares* CAD comerciais, como, por exemplo, SolidEdge.

- Por fim, a aplicação dos conceitos em outras áreas da indústria.

REFERÊNCIAS BIBLIOGRÁFICAS

ALEKSIĆ, D. S.; JANKOVIĆ, D. S.; STOIMENOV, L. V. **A case study on the object-oriented framework for modeling product families with the dominant variation of the topology in the one-of-a-kind production.** The International Journal of Advanced Manufacturing Technology, v. 59, n. 1, p. 397-412, 2011.

ASSOCIAÇÃO NACIONAL DOS FABRICANTES DE ÔNIBUS (FABUS). Disponível em: <http://www.fabus.com.br/producao.htm>. Acesso em 05 de Setembro de 2016.

ATHANASOPOULOS, M.; UGAIL, H.; CASTRO, G. G. **Parametric design of aircraft geometry using partial differential equations.** Advances in Engineering Software, v. 40, n. 7, p. 479.486, 2009.

BALDWIN, C. Y.; CLARK, K. B. **Design Rules: The Power of Modularity.** Massachusetts Institute of Tecnology, v. 1, Cambridge MA, 2000.

BALDWIN, C. Y.; CLARK, K. B. **Modularity in the Design of Complex Engineering Systems**. Complex Engineered Systems, p. 175-205. Harvard Business School.

BECK, K.; FOWLER, M., **Extreme Programming Explained: Embrace Change. Beck, K. and Fowler, M. (2000). Planning Extreme Programming.** Addison Wesley Professional. 2000.

CIAPPARINI, J. **Avaliação de Fadiga de uma Carroceria de Ônibus Submetida a Diferentes Perfis de Pista.** Rio Grande do Sul: Escola de Engenharia Mecânica, Universidade Federal do Rio Grande do Sul (Doutorado), 2012.

CIMDATA. **PLM Education, Research & Strategic Management Consulting**. Disponível em: <https://www.cimdata.com/en/resources>. Acesso em 15/11/2016.

CUNHA, C.; AGARD, B.; KUSIAK, A. **Selection of Modules for Mass Customisation.** International Journal of Production Research, v. 48, n. 5, p. 1439-1454, 2010.

FRUTOS, J. D. **Um Modelo para Configuração de Produtos Oferecidos em um Ambiente de Customização em Massa**, 2006. (Doutorado) – Programa de Pós-Graduação em Administração, Universidade Federal do Rio Grande do Sul.

HEVNER, A. R.; CHATTERJEE, S. **Design Research in Information System. Theory and Practice**. Integrated Series in Information Systems, v. 22, 2010.

DINGSØYR, T.; NERUS, S.; BALIJEPALLY, V.; MOE, N. N. **A decade of agile methodologies: Towards explaining agile software development**. The Journal of Systems and Software, v. 85, n. 6, p. 1213-1221, 2012.

GAMBA, A.; FUSARI N. **Valuing Modularity as a Real Option**. Management Science, v. 55, n.11, p. 1877-1896, 2009.

GOMES, A. F; **Agile - Desenvolvimento de *software* com entregas frequentes e foco no valor de negócio**. Casa do Código, 2013.

GRIEVES, M. **Product Lifecycle Management**: Driving the Next Generation of Lean Thinking. United States of America: Mcgraw-hill, 2005.

GRIEVES, M. **Virtually Indistinguishable, System Engineering and PLM**. IFIP International Federation for Information Processing, v. 388, p. 228-244, 2012.

MAVRIDOU, E.; KEHAGIAS, D. D.; TZOVARAS, D.; HASSAPIS, G. **Mining affective needs of automotive industry customers for building a mass-customization recommender system**. Journal of Intelligent Manufacturing, v. 24, n. 2, p. 251-265, 2013.

MEDEIROS, E. **Desenvolvendo *software* com UML 2.0**. Pearson Makron Books, 2014.

MYUNG, S.; HAN, S. **Knowledge-based parametric design of mechanical products based on configuration design method**. Expert Systems with Applications, v. 21, n. 2, p. 99-107, 2001.

NAPPER, R. **Modular route bus design-A method of meeting transport operation and vehicle manufacturing requirements.** Transportation Research Part C: Emerging Technologies, v. 38, p. 56-72, 2014.

PALENCIA, A. E. R.; DELGADILLO, G. E. M, **A Computer Application for a Bus Body Assembly Line Using Genetic Algorithms**. International Journal of Production Economics, v. 140, n. 1, p. 431-438, 2012.

PAPADOPOULOS, G. **Moving from traditional to agile software development methodologies also on large, distributed projects.** Science Direct, v. 175, p. 455-463, 2015.

PATEL, P. C.; JAYARAM, J. **The antecedents and consequences of product variety in new ventures: An empirical study**. Journal of Operations Management, v. 32, n. 1, p. 34-50, 2014.

PAULA, F.; AMARAL, D.; ROZENFELD, H. **Análise da integração entre um sistema de gestão de dados de documentos e um sistema de gestão de projetos: contexto da gestão do clico de vida de produtos (PLM)**. XXVII Encontro Nacional de Engenharia de Produção. Universidade Federal de Santa Catarina, 2007.

PEFFERS, K.; ROTHENBERGER M. A.; TUUNANEN, T.; VAEZI, R. **Design Science Research Evaluation**. Design Science Research in Information Systems. Advances in Theory and Practice, v. 7285, p. 398-410, 2012.

PEFFERS, K.; TUUNANEN, T.; GENGLER, C. E.; ROSSI, M.; HUI, W.; VIRTANEN, V.; BRAGGE, J. **The Design Science Research Process: A Model for Producing and Presenting Information Systems Research**. International Conference on Design Science in Information Systems and Technology, p. 83-106, 2006.

PEFFERS, K.; TUUNANEN, T., ROTHENBERGER M. A.; CHATTERJEE S. A. **Design Science Research Methodology for Information Systems Research**. Journal of Management Information Systems, v. 24 n. 3, p. 45–78, 2007.

PFLEEGER, S. L. **Engenharia de *Software*: Teoria e Prática**, Pearson, 2ª Edição, 2004.

PLM WORLD. Teamcenter Manufacturing Deployment at Honda. Siemens PLM Connection Americas 2010 Users Conference. Disponível em: < http://www.plmworld.org/Past_Conferences>. Acesso em 19 de Novembro de 2016.

QUEVEDO, D. B. **Aplicação de Métodos de Projeto no Desenvolvimento de Carrocerias de Ônibus: um estudo de caso.** Passo Fundo: Faculdade de Engenharia Mecânica, Programa de Pós-Graduação em Projeto e Processos de Fabricação, Universidade de Passo Fundo, Dissertação (Mestrado), 2014.

RAFFAELI, R.; MENGONI, M.; GERMANI, M. **Improving the link between computer-assisted design and configuration tools for the design of mechanical products**. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, v. 27, n. 1, p. 51-64, 2013.

REN, B.; QIU, L.; ZHANG, S.; TAN, J.; CHENG, J. **Configurable Product Design Considering the Transition of Multi-hierarchical Models**. Chinese Journal of Mechanical Engineering, v. 26, n. 2, p. 217-224, 2013.

SCANIA. Disponível em: <https://www.scania.com/br/pt/home/products-and-services/buses-and-coaches/our-range/intercity-chassis.html>. Acesso em 22 de Novembro de 2016.

SIEMENS PLM. Disponível em: < https://www.plm.automation.siemens.com>. Acesso em 10 de Julho de 2016.

SOMMERVILLE, I. **Engenharia de *Software***, Pearson, 9ª Edição, 2011.

STARK, J. **Product Lifecycle Management: 21st Century Paradigm for Product Realisation**. 2. ed. Springer London, 2011.

STARK, J. **Product Lifecycle Management (Volume 1): 21st Century Paradigm for Product Realisation**. 3. ed. Springer London, 2015.

ULRICH, K. **The role of product architecture in the manufacturing firm**. Research policy, v 24, n. 3, p. 419-440, 1995.

VIERO, C. **Metodologia de Projeto para Arranjo Estrutural de Carroceria de Ônibus Através de Sistemas Modulares: um estudo de caso.** Passo Fundo: Faculdade de Engenharia Mecânica, Programa de Pós-Graduação em Projeto e Processos de Fabricação, Universidade de Passo Fundo, Dissertação (Mestrado), 2013.

WALBER, M. **Avaliação dos Níveis de Vibração Existentes em Passageiros de Ônibus Rodoviários Intermunicipais, Análise e Modificação Projetual.** Rio Grande do Sul: Escola de Engenharia Mecânica, Universidade Federal do Rio Grande do Sul, (Doutorado) 2009.

ZORRIASSATIME, F; WYKES, C; PARKIN, R; GINDY, N. **A Survey of Virtual Prototyping Techniques for Mechanical Product Development**. Proceedings of the Institution of Mechanical Engineers, Journal of Engineering Manufacture, v. 217(4), p. 513-530, 2003.

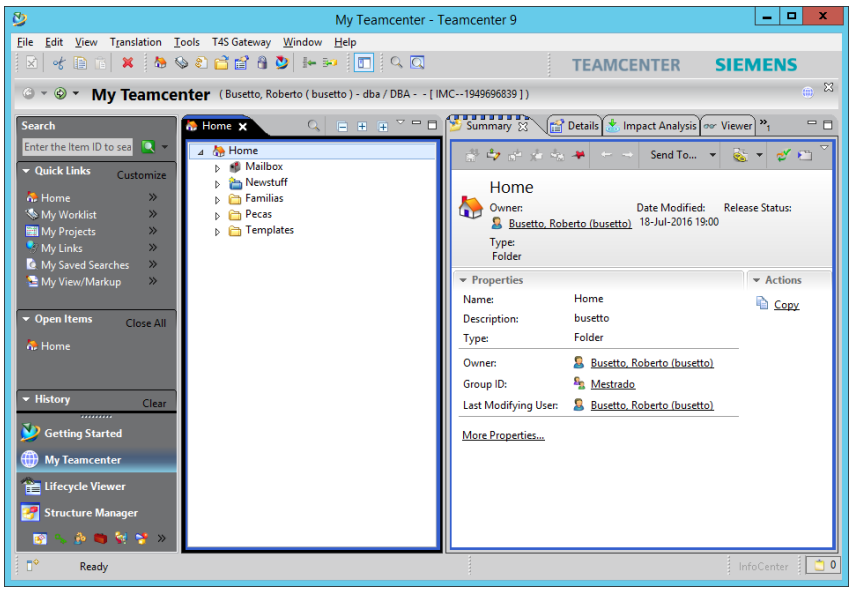
APÊNDICE A – Sistemas de engenharia

A empresa estudada possui um ambiente de engenharia sistematicamente integrado com os *softwares* Siemens PLM. Esta seção é dedicada ao entendimento dos principais *softwares* envolvidos no processo de desenvolvimento de uma carroceria de ônibus. Neste estudo, dois *softwares* do sistema PLM serão utilizados: Siemens Teamcenter e Siemens CAD NX.

O *software* Teamcenter é um conjunto de soluções para o gerenciamento do clico de vida do produto. O portfólio modular de soluções digitais pode ser configurado de acordo com cada cliente para gestão e planejamento de requisitos, desenvolvimento de produto e processos de fabricação, liga parceiros e fornecedores a uma base de dados unificada. Teamcenter oferece uma única fonte de conhecimento de processos e de engenharia de produto, totalmente integrada a CAD, CAM e CAE (SIEMENS, 2016).

Teamcenter permite gerenciar digitalmente os dados do produto e de manufatura no contexto do ciclo de vida do produto, proporcionando acesso global aos dados da organização, conectando pessoas, processos e informações, faz a gestão de alterações de produto e integrações com outros sistemas (SIEMENS, 2016).

Figura A. 1 - Interface do *software* Teamcenter

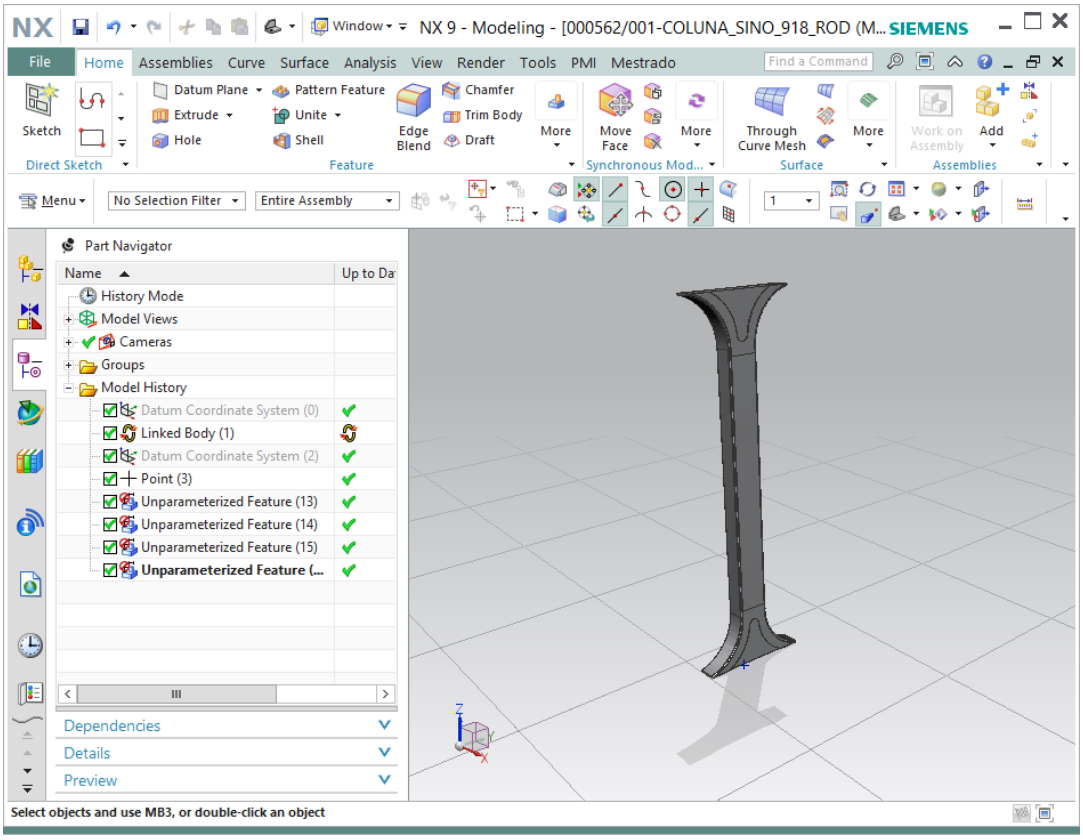


Fonte: Autor, 2017

No contexto dessa dissertação, o Teamcenter será utilizado para gestão do desenvolvimento de produto, organizando os arquivos CAD e integrado ao *software* NX. A Figura A. 1 exibe a interface de usário do *software* NX.

O NX é um *software* de desenvolvimento de produto digital que ajuda as empresas a transformarem o ciclo de vida do produto com um conjunto de aplicativos CAD/CAM/CAE totalmente associativos e integrados. O NX alcança toda a variedade de processos de desenvolvimento de produto. A Figura A. 2 exibe a *interface* do *software* NX.

Figura A. 2 - Interface do *software* NX



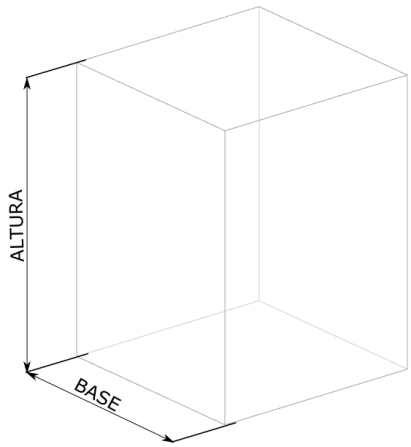
Fonte: Autor, 2017

No contexto dessa dissertação, três recursos do *software* NX são essenciais: *Expressions*, *Part Family* e NX *Open*. Os três recursos são detalhados nas próximas três sessões.

As expressões (*expressions*) são entidades presentes no *software* CAD NX que possuem fórmulas aritméticas e/ou valores que podem definir o comportamento dos objetos. É possível controlar o relacionamento entre as *features* de uma peça ou montagem, como, por exemplo, na Figura A. 3, utilizam-se as expressões ALTURA e BASE para controlar as dimensões da

geometria. Caso as expressões tenham seus valores alterados, a altura e base do cubo são alteradas automaticamente.

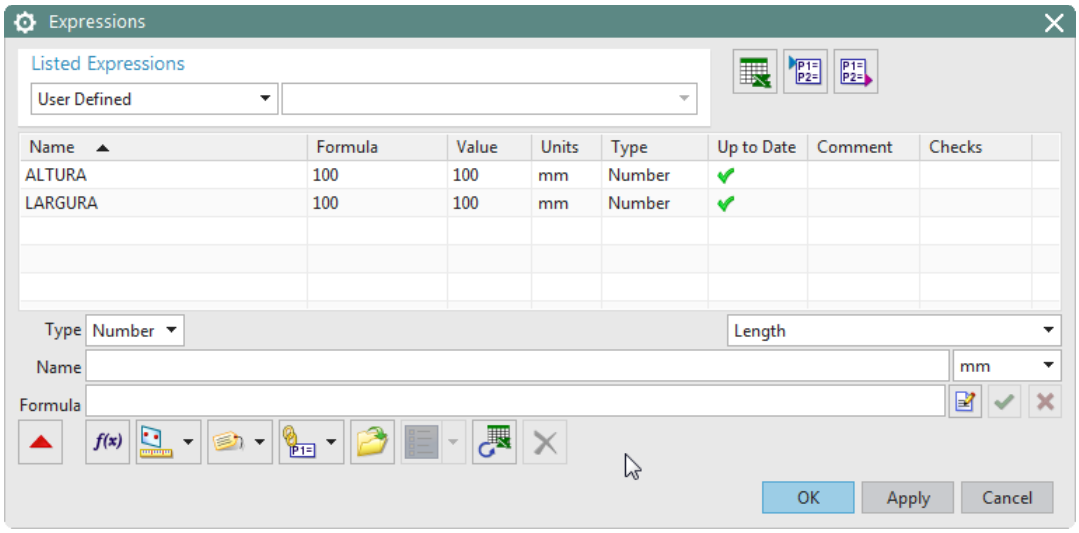
Figura A. 3 - Exemplo de utilização de expressão



Fonte: Autor, 2017

Conforme a interface para criação de expressões, representada na Figura A. 4, as fórmulas podem conter uma combinação de variáveis, funções, números, operadores e símbolos. Os campos obrigatórios são: *Type* (tipo de dado da expressão), *Name* (nome da expressão) e *Formula* (fórmula ou valor).

Figura A. 4 - Interface para criação de expressões

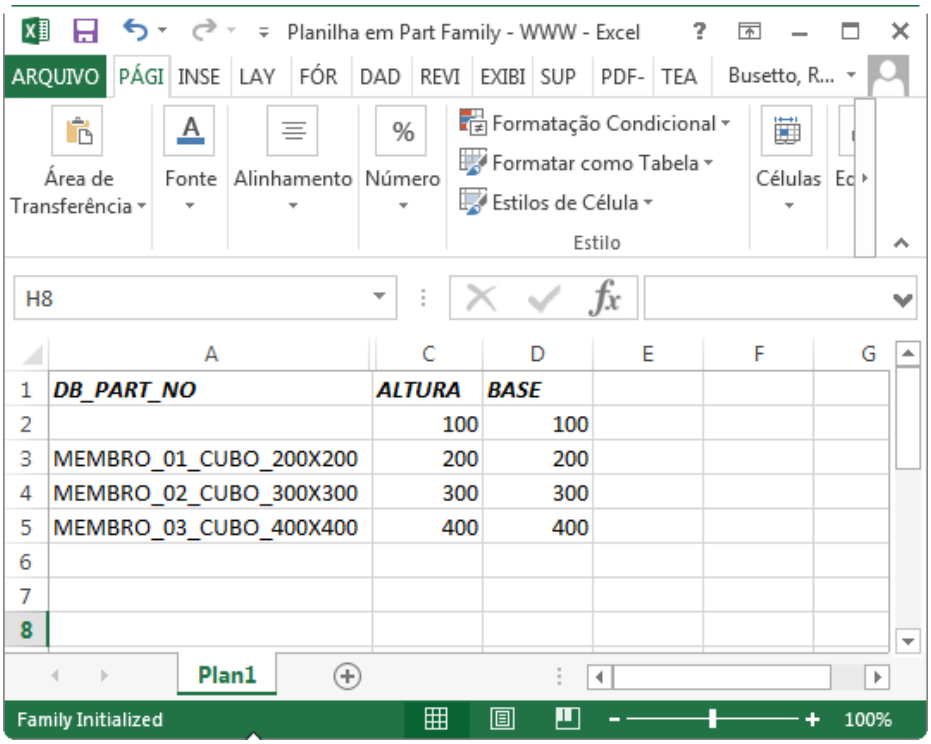


Fonte: Autor, 2017

O módulo Part Family é utilizado para gerar famílias de peças e conjuntos com similaridade geométrica. Utiliza-se um projeto como modelo e em seguida utiliza-se uma

planilha de Excel para criar uma tabela que descreve as variações deste modelo principal, denominados membros de família. Os membros de família são criados, alterados e deletados somente pela tabela de Excel. A utilização de famílias de peças e conjuntos gera reuso e padronização de projeto.

Figura A. 5 - Exemplo de *part family*



Fonte: Autor, 2017

Utilizando como modelo o cubo da Figura A. 3, criou-se o *Part Family*, representado na Figura A. 5, em que foram geradas três variações do cubo modelo.

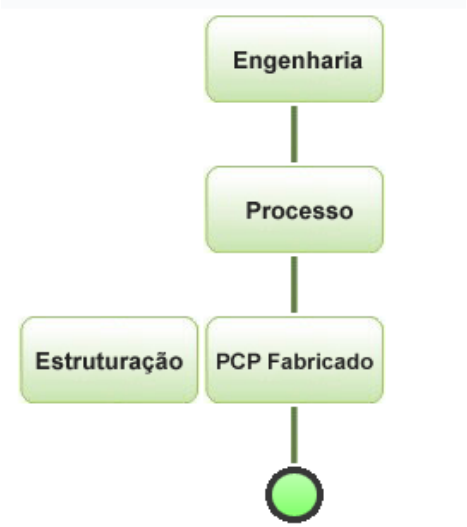
O *software* NX fornece uma arquitetura de automação denominada NX Open. NX Open é uma coleção de interfaces (API) que permite a desenvolvedores integrar suas aplicações com o NX. As APIs fornecem uma arquitetura aberta que pode ser utilizada por terceiros para integrações com aplicações personalizadas, podendo, assim, colaborar de forma eficiente na concepção, desenvolvimento e fabricação de produtos. As APIs suportam as tecnologias C, C++, .NET e JAVA (SIEMENS PLM, 2016).

Nesse estudo optou-se pela utilização da API NX Open para .NET, com desenvolvimento em linguagem CSharp, pois todo o ambiente da empresa estudada possui sistema operacional Microsoft Windows, tirando, assim, o máximo de proveito de todos os benefícios proporcionados pela API.

Todos os códigos de produtos abertos dentro da empresa obrigatoriamente são criados utilizando um *software* de *workflow* desenvolvido internamente. Os fluxos de *workflow* não possuem tempo estipulado, em cada etapa, para cadastro e liberação do fluxo.

Conforme ilustrado na Figura A. 6, todos os itens (códigos) produzidos internamente passam por quatro etapas até sua ativação no sistema: Engenharia, Processo, Estruturação e PCP Fabricado. Engenharia informa dados básicos para cadastro do item, Processo desenvolve os roteiros de fabricação e/ou montagem, Estruturação efetua a conferência da estrutura de produto, PCP Fabricado cadastra as opções de MRP. O fluxo é sequencial entre Engenharia e Processo e, após liberação de Processo, o fluxo é aberto paralelamente nas áreas de Estruturação e PCP Fabricado.

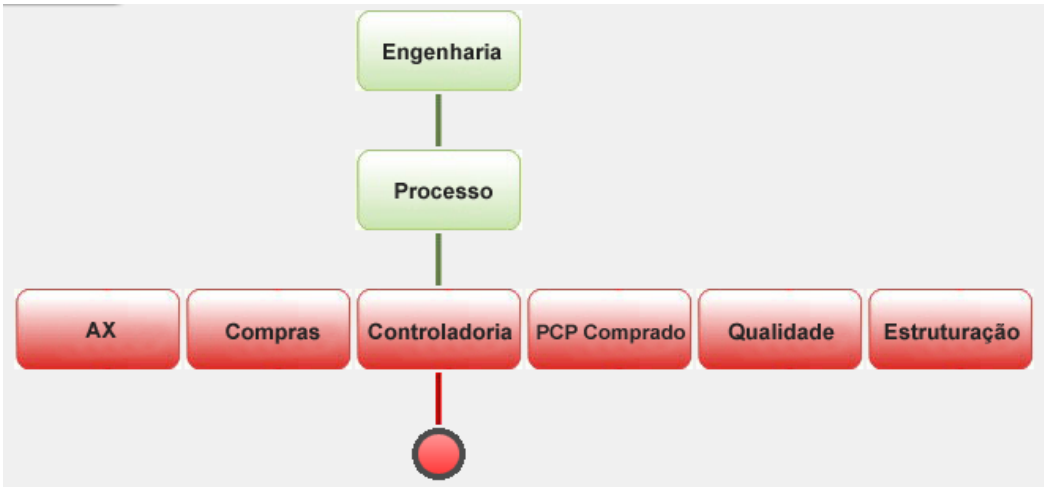
Figura A. 6 - Fluxo de *workflow* para itens produzidos



Fonte: Autor, 2017

Para itens comprados, representados na Figura A. 7, o fluxo passa por oito etapas até a ativação total: Engenharia, Processo, AX (almoxarifado), Compras, Controladoria, PCP Comprado, Qualidade e Estruturação. Engenharia informa dados básicos para cadastro do item, Processo desenvolve os roteiros de fabricação e/ou montagem, AX aponta local de estoque/armazenagem, Compras cadastra fornecedor, Controladoria informa itens fiscais, PCP Comprado cadastra as opções de MRP, Qualidade cadastra informações para acompanhamento de qualidade por amostra e Estruturação efetua a conferencia da estrutura de produto. O fluxo é sequencial entre Engenharia e Processo, após liberação de Processo o fluxo é aberto paralelamente nas seis áreas restantes.

Figura A. 7 - Fluxo de *workflow* para itens comprados

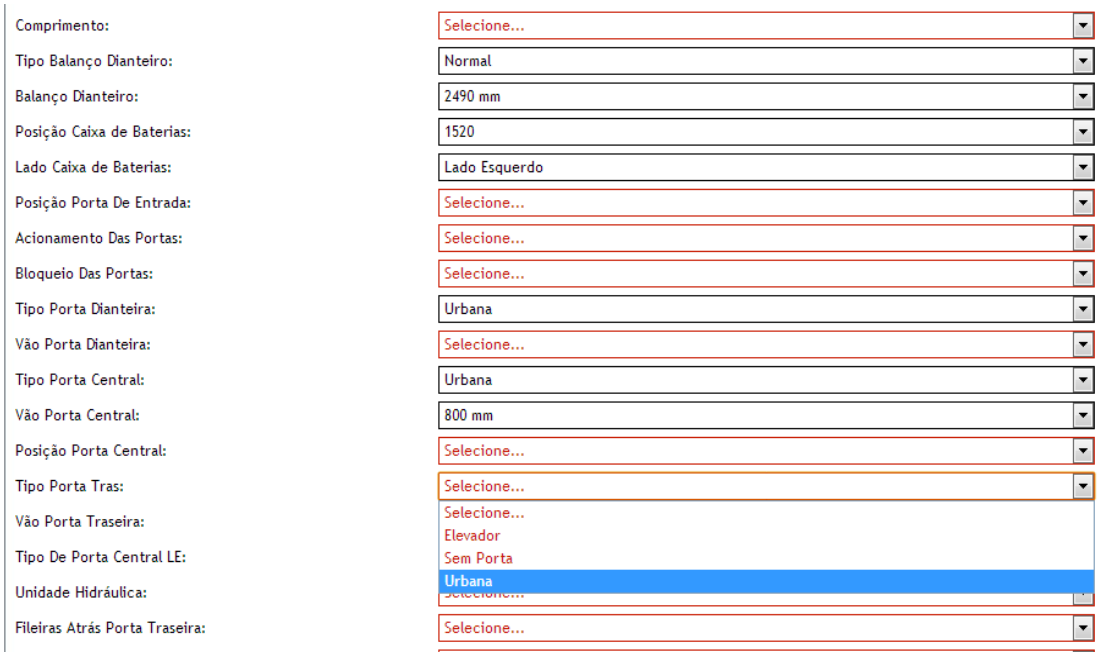


Fonte: Autor, 2017

O *software* para configuração de produto também foi desenvolvido internamente. O configurador de produto trabalha como interface entre o departamento comercial e engenharia, sendo que todos os pedidos efetivados têm sua configuração válida para a Engenharia de Produto, ou seja, todos os pedidos que chegam para a Engenharia de Produto já estão validados, como, por exemplo: o vendedor só poderá oferecer determinado comprimento da carroceria se esta opção estiver disponível nas tarefas de venda. Todas as regras de configuração de venda são definidas e desenvolvidas na Engenharia de Desenvolvimento e Engenharia de Produto.

Conforme ilustrado na Figura A. 8, a coluna da esquerda refere-se à característica de venda e a coluna da direita refere-se às opções de venda. Entende-se por tarefa de venda cada uma das características e opções do configurador, como, por exemplo, onde a característica de venda “Tipo Porta Traseira” possui as opções: Elevador, Sem Porta e Urbana.

Figura A. 8 - Exemplo de tarefa de venda



Fonte: Autor, 2017

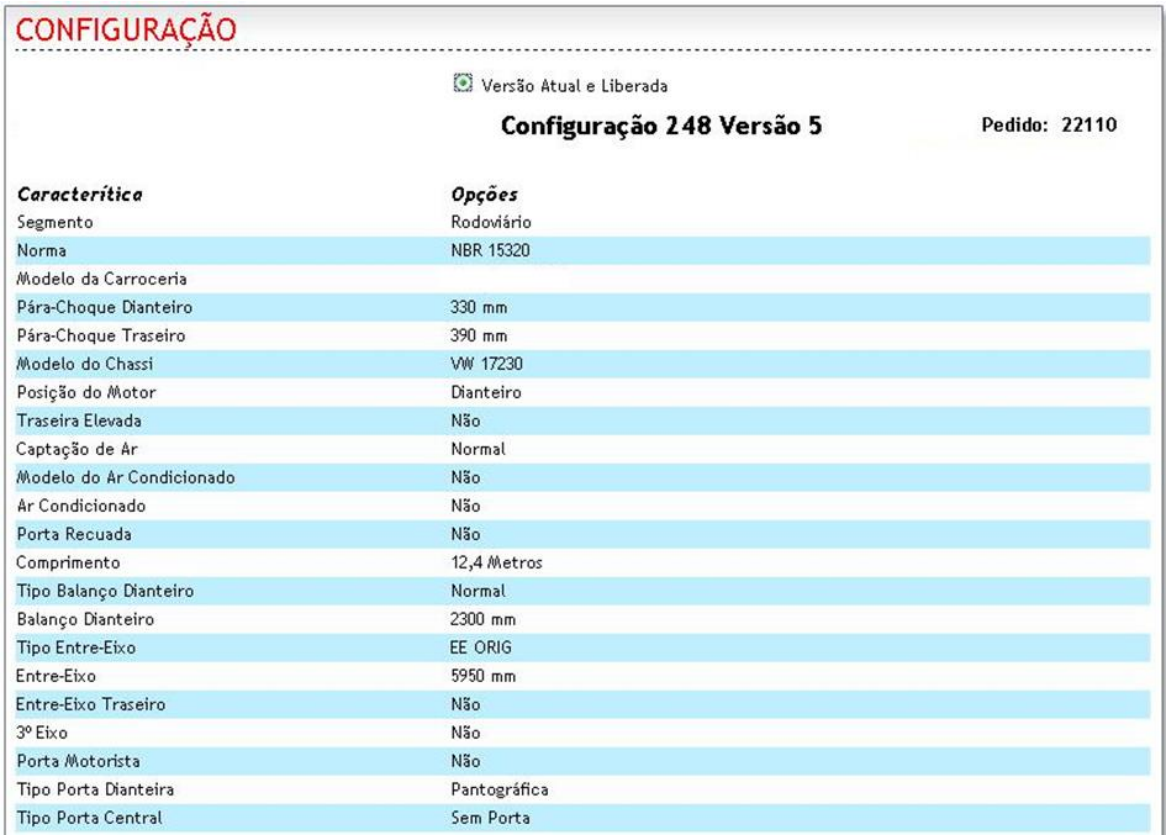
Obrigatoriamente, toda configuração de produto possui ligação com um pedido. A Figura A. 9 exemplifica a tela inicial do *software* de configuração de produto, na qual é possível observar seis pedidos de clientes distintos.

Figura A. 9 - Interface do *software* configurador de produto

Fonte: Autor, 2017

A Figura A. 10 exemplifica uma configuração de produto já liberada para a Engenharia de Produto trabalhar nos projetos, seno que a coluna da esquerda é a característica de venda e os valores da direita são as opções de venda.

Figura A. 10 - Configuração de produto



CONFIGURAÇÃO

Versão Atual e Liberada

Configuração 248 Versão 5 **Pedido: 22110**

Caracterítica	***Opções***
Segmento	Rodoviário
Norma	NBR 15320
Modelo da Carroceria	
Pára-Choque Dianteiro	330 mm
Pára-Choque Traseiro	390 mm
Modelo do Chassi	VW 17230
Posição do Motor	Dianteiro
Traseira Elevada	Não
Captação de Ar	Normal
Modelo do Ar Condicionado	Não
Ar Condicionado	Não
Porta Recuada	Não
Comprimento	12,4 Metros
Tipo Balanço Dianteiro	Normal
Balanço Dianteiro	2300 mm
Tipo Entre-Eixo	EE ORIG
Entre-Eixo	5950 mm
Entre-Eixo Traseiro	Não
3º Eixo	Não
Porta Motorista	Não
Tipo Porta Dianteira	Pantográfica
Tipo Porta Central	Sem Porta

Fonte: Autor, 2017

ANEXO A – Código-fonte do *template* conjunto estrutura lateral esquerda

Código-fonte do *template* conjunto estrutura lateral esquerda para a aplicação *Automator*.

```
public void addComponentesPadronizados(){
        //adiciona o complemento dianteiro
        CMP_001_X = 0;
        CMP_001_Y = 0;
        CMP_001_Z = 0;
        CMP_001_A_X = 0;
        CMP_001_A_Y = 0;
        CMP_001_A_Z = 0;
        CMP_001_ISASSEMBLY = 0;
        CMP_001_NOME = "";
        CMP_001_DESC = CMP_001_NOME;
        CMP_001_PARAM = "";
        CMP_001_PARAM_VALUE = "";
        CMP_001_SUPP = 1;
        CMP_001_ID = "000533";
}
public void addTubosLongitudinais(){
        //adiciona o tubo do peitoril inferior da janela
        CMP_003_X = 16;
        CMP_003_Y = SK_POS_INICIO_SALAO;
        CMP_003_Z = SK_ALT_PEITORIL;
        CMP_003_A_X = 90;
        CMP_003_A_Y = 0;
        CMP_003_A_Z = 4;
        CMP_003_ISASSEMBLY = 0;
        CMP_003_NOME = "TB_70x50x2.3x0x0x" + (SK_COMPRIMENTO - SK_POS_INICIO_SALAO -
        SK_LARG_EST_FIBRA_TRAS - SK_LARG_EST_FIBRA_DIANT);
        CMP_003_DESC = CMP_003_NOME;
        CMP_003_PARAM = "FAM|COMPRIMENTO|ALTURA|LARGURA|ESP|ANGULO_A|ANGULO_B";
        CMP_003_PARAM_VALUE = "000527|" + (SK_COMPRIMENTO - SK_POS_INICIO_SALAO -
        SK_LARG_EST_FIBRA_TRAS - SK_LARG_EST_FIBRA_DIANT)+"|70|50|2.3|0|0";
        CMP_003_SUPP = 1;
        CMP_003_ID = "";

        //adiciona o tubo do peitoril superior da janela
        CMP_004_X = -48;
        CMP_004_Y = SK_POS_INICIO_SALAO;
```

```
        CMP_004_Z = SK_ALT_PEITORIL + SK_ALT_VAO_JANELA + 70;
        CMP_004_A_X = 90;
        CMP_004_A_Y = 0;
        CMP_004_A_Z = 4;
        CMP_004_ISASSEMBLY = 0;
        CMP_004_NOME = "TB_70x50x2.3x0x0x" + (SK_COMPRIMENTO - SK_POS_INICIO_SALAO -
        SK_LARG_EST_FIBRA_TRAS - SK_LARG_EST_FIBRA_DIANT);
        CMP_004_DESC = CMP_003_NOME;
        CMP_004_PARAM = "FAM|COMPRIMENTO|ALTURA|LARGURA|ESP|ANGULO_A|ANGULO_B";
        CMP_004_PARAM_VALUE = "000527|"+(SK_COMPRIMENTO - SK_POS_INICIO_SALAO -
        SK_LARG_EST_FIBRA_TRAS - SK_LARG_EST_FIBRA_DIANT)+"|70|50|2.3|0|0";
        CMP_004_SUPP = 1;
        CMP_004_ID = "";

        //adiciona o tubo do perfil lateral - centro
        CMP_007_X = 22;
        CMP_007_Y = (SK_LARG_VAO_PT_DIANT + 40 + 70);
        CMP_007_Z = SK_ALT_PEITORIL - SK_ALT_PERF_BAGUETE_LAT - 70; //70 eh a largura do tubo
        CMP_007_A_X = 90;
        CMP_007_A_Y = 0;
        CMP_007_A_Z = -1.5;
        CMP_007_ISASSEMBLY = 0;
        CMP_007_NOME = "TB_70x50x2.3x0x0x" + (SK_COMPRIMENTO - (SK_LARG_VAO_PT_DIANT + 40 + 70) -
        SK_LARG_EST_FIBRA_TRAS - SK_LARG_EST_FIBRA_DIANT);
        CMP_007_DESC = CMP_003_NOME;
        CMP_007_PARAM = "FAM|COMPRIMENTO|ALTURA|LARGURA|ESP|ANGULO_A|ANGULO_B";
        CMP_007_PARAM_VALUE = "000527|"+(SK_COMPRIMENTO - (SK_LARG_VAO_PT_DIANT + 40 + 70) -
        SK_LARG_EST_FIBRA_TRAS - SK_LARG_EST_FIBRA_DIANT)+"|70|50|2.3|0|0";
        CMP_007_SUPP = 1;
        CMP_007_ID = "";
}
public void addColunaSino(){
        //adiciona a primeira coluna sino, padrao
        CMP_006_X = 65;
        CMP_006_Y = SK_POS_PRIM_COLUNA; //eh a posicao fixa da primeira coluna sino
        CMP_006_Z = SK_ALT_PEITORIL + 2.3;
        CMP_006_A_X = 0;
        CMP_006_A_Y = 0;
        CMP_006_A_Z = 0;
        CMP_006_ISASSEMBLY = 0;
        CMP_006_NOME = "";
        CMP_006_DESC = CMP_006_NOME;
        CMP_006_PARAM = "";
```

```
        CMP_006_PARAM_VALUE = "";
        CMP_006_SUPP = 1;
        CMP_006_ID = "000562";

        calculaVaoJanelas();
        addColunasRestantes();
        addUltimaColuna();

        QTD_JANELAS_PADRAO = (VAO_TOTAL_DISPONIVEL / (SK_VAO_PADRAO +
        SK_LARG_COLUNA_SINO)).intValue();
}

public void calculaVaoJanelas(){
        VAO_TOTAL_DISPONIVEL = SK_COMPRIMENTO - (SK_LARG_EST_FIBRA_DIANT +
        SK_LARG_EST_FIBRA_TRAS) - SK_LARG_COLUNA_SINO - SK_POS_PRIM_COLUNA; //128.35 eh da coluna
        sino final + largura do tubo
        QTD_JANELAS_PADRAO = (VAO_TOTAL_DISPONIVEL / (SK_VAO_PADRAO +
        SK_LARG_COLUNA_SINO)).intValue();
        RESTO = (VAO_TOTAL_DISPONIVEL % (SK_VAO_PADRAO + SK_LARG_COLUNA_SINO));
}

public double getPosColSino(boolean primeira){
        if(primeira){
                POS_ULT_COL_SINO = SK_POS_PRIM_COLUNA + SK_LARG_COLUNA_SINO +
                SK_VAO_PADRAO;
        }else{
                POS_ULT_COL_SINO = POS_ULT_COL_SINO + SK_LARG_COLUNA_SINO + SK_VAO_PADRAO;
        }
        return POS_ULT_COL_SINO;
}

public void addColunasRestantes(){
        //adiciona coluna sino, padrao
        CMP_014_X = 65;
        CMP_014_Y = getPosColSino(true);
        CMP_014_Z = SK_ALT_PEITORIL + 2.3;
        CMP_014_A_X = 0;
        CMP_014_A_Y = 0;
        CMP_014_A_Z = 0;
        CMP_014_ISASSEMBLY = 0;
        CMP_014_NOME = "";
        CMP_014_DESC = CMP_014_NOME;
        CMP_014_PARAM = "";
```

```
CMP_014_PARAM_VALUE = "";
CMP_014_SUPP = 1;
CMP_014_ID = "000562";
if(--QTD_JANELAS_PADRAO == 0) return;

//adiciona coluna sino, padrao
CMP_015_X = 65;
CMP_015_Y = getPosColSino(false);
CMP_015_Z = SK_ALT_PEITORIL + 2.3;
CMP_015_A_X = 0;
CMP_015_A_Y = 0;
CMP_015_A_Z = 0;
CMP_015_ISASSEMBLY = 0;
CMP_015_NOME = "";
CMP_015_DESC = CMP_015_NOME;
CMP_015_PARAM = "";
CMP_015_PARAM_VALUE = "";
CMP_015_SUPP = 1;
CMP_015_ID = "000562";
if(--QTD_JANELAS_PADRAO == 0) return;

//adiciona coluna sino, padrao
CMP_016_X = 65;
CMP_016_Y = getPosColSino(false);
CMP_016_Z = SK_ALT_PEITORIL + 2.3;
CMP_016_A_X = 0;
CMP_016_A_Y = 0;
CMP_016_A_Z = 0;
CMP_016_ISASSEMBLY = 0;
CMP_016_NOME = "";
CMP_016_DESC = CMP_016_NOME;
CMP_016_PARAM = "";
CMP_016_PARAM_VALUE = "";
CMP_016_SUPP = 1;
CMP_016_ID = "000562";
if(--QTD_JANELAS_PADRAO == 0) return;

//adiciona coluna sno, padrao
CMP_017_X = 65;
CMP_017_Y = getPosColSino(false);
CMP_017_Z = SK_ALT_PEITORIL + 2.3;
CMP_017_A_X = 0;
CMP_017_A_Y = 0;
```

```
        CMP_017_A_Z = 0;
        CMP_017_ISASSEMBLY = 0;
        CMP_017_NOME = "";
        CMP_017_DESC = CMP_017_NOME;
        CMP_017_PARAM = "";
        CMP_017_PARAM_VALUE = "";
        CMP_017_SUPP = 1;
        CMP_017_ID = "000562";
        if(--QTD_JANELAS_PADRAO == 0) return;

        //adiciona coluna sno, padrao
        CMP_018_X = 65;
        CMP_018_Y = getPosColSino(false);
        CMP_018_Z = SK_ALT_PEITORIL + 2.3;
        CMP_018_A_X = 0;
        CMP_018_A_Y = 0;
        CMP_018_A_Z = 0;
        CMP_018_ISASSEMBLY = 0;
        CMP_018_NOME = "";
        CMP_018_DESC = CMP_018_NOME;
        CMP_018_PARAM = "";
        CMP_018_PARAM_VALUE = "";
        CMP_018_SUPP = 1;
        CMP_018_ID = "000562";
        if(--QTD_JANELAS_PADRAO == 0) return;
}

public void addUltimaColuna(){
        if(RESTO > 1320){
                //adiciona a ultima coluna sino, cortada.
                CMP_005_X = 65;
                //CMP_005_Y = SK_COMPRIMENTO - SK_LARG_EST_FIBRA_TRAS - SK_LARG_EST_FIBRA_DIANT
                - 58;
                CMP_005_Y = POS_ULT_COL_SINO + SK_LARG_COLUNA_SINO + SK_LARG_COLUNA_SINO +
                1320;
                CMP_005_Z = SK_ALT_PEITORIL + 2.3;
                CMP_005_A_X = 0;
                CMP_005_A_Y = 0;
                CMP_005_A_Z = 0;
                CMP_005_ISASSEMBLY = 0;
                CMP_005_NOME = "";
                CMP_005_DESC = CMP_005_NOME;
                CMP_005_PARAM = "";
```

```
                CMP_005_PARAM_VALUE = "";
                CMP_005_SUPP = 1;
                CMP_005_ID = "000563";
                RESTO = 0;
        }else if(RESTO > 920){
                //adiciona a ultima coluna sino, cortada.
                CMP_005_X = 65;
                CMP_005_Y = POS_ULT_COL_SINO + SK_LARG_COLUNA_SINO + SK_LARG_COLUNA_SINO +
                920;
                CMP_005_Z = SK_ALT_PEITORIL + 2.3;
                CMP_005_A_X = 0;
                CMP_005_A_Y = 0;
                CMP_005_A_Z = 0;
                CMP_005_ISASSEMBLY = 0;
                CMP_005_NOME = "";
                CMP_005_DESC = CMP_005_NOME;
                CMP_005_PARAM = "";
                CMP_005_PARAM_VALUE = "";
                CMP_005_SUPP = 1;
                CMP_005_ID = "000563";
                RESTO = 0;
        }else{
        //adiciona a ultima coluna sino, cortada.
                CMP_005_X = 0;
                CMP_005_Y = 0;
                CMP_005_Z = 0;
                CMP_005_A_X = 0;
                CMP_005_A_Y = 0;
                CMP_005_A_Z = 0;
                CMP_005_ISASSEMBLY = 0;
                CMP_005_NOME = "";
                CMP_005_DESC = CMP_005_NOME;
                CMP_005_PARAM = "";
                CMP_005_PARAM_VALUE = "";
                CMP_005_SUPP = 1;
                CMP_005_ID = "0";
                RESTO = 0;
        }
}
public void addArodeRoda(){
        //adiciona conjunto aro de roda dianteiro
        CMP_009_X = 23;
        CMP_009_Y = SK_BAL_DIANT;
```

```
        CMP_009_Z = 0;
        CMP_009_A_X = 0;
        CMP_009_A_Y = 0;
        CMP_009_A_Z = 0;
        CMP_009_ISASSEMBLY = 0;
        CMP_009_NOME = "";
        CMP_009_DESC = CMP_009_NOME;
        CMP_009_PARAM = "";
        CMP_009_PARAM_VALUE = "";
        CMP_009_SUPP = 1;
        CMP_009_ID = "000566";

        //adiciona conjunto aro de roda traseiro
        CMP_010_X = 48;
        CMP_010_Y = SK_BAL_DIANT + SK_EE;
        CMP_010_Z = 0;
        CMP_010_A_X = 0;
        CMP_010_A_Y = 0;
        CMP_010_A_Z = 0;
        CMP_010_ISASSEMBLY = 0;
        CMP_010_NOME = "";
        CMP_010_DESC = CMP_010_NOME;
        CMP_010_PARAM = "";
        CMP_010_PARAM_VALUE = "";
        CMP_010_SUPP = 1;
        CMP_010_ID = "000567";
}
public void addTuboSaia(){
        //adiciona conjunto bagageiro lateral traseiro
        CMP_008_X = 45;
        CMP_008_Y = SK_COMPRIMENTO - SK_LARG_EST_FIBRA_TRAS - SK_LARG_EST_FIBRA_DIANT;
        CMP_008_Z = 0;
        CMP_008_A_X = 0;
        CMP_008_A_Y = 0;
        CMP_008_A_Z = 0;
        CMP_008_ISASSEMBLY = 0;
        CMP_008_NOME = "";
        CMP_008_DESC = CMP_008_NOME;
        CMP_008_PARAM = "";
        CMP_008_PARAM_VALUE = "";
        CMP_008_SUPP = 1;
        CMP_008_ID = "000565";
```

```
        //adiciona o primeiro tudo da saia lateral
        CMP_012_X = 0;
        CMP_012_Y = SK_BAL_DIANT - (SK_VAO_ARO_RODA / 2);
        CMP_012_Z = 0;
        CMP_012_A_X = -90;
        CMP_012_A_Y = 0;
        CMP_012_A_Z = 0;
        CMP_012_ISASSEMBLY = 0;
        CMP_012_NOME = "TB_30x47x1.95x0x0x" + (SK_BAL_DIANT - (SK_VAO_ARO_RODA / 2) -
        (SK_LARG_VAO_PT_DIANT + 70));
        CMP_012_DESC = CMP_012_NOME;
        CMP_012_PARAM = "FAM|COMPRIMENTO|ALTURA|LARGURA|ESP|ANGULO_A|ANGULO_B";
        CMP_012_PARAM_VALUE = "000527|" + (SK_BAL_DIANT - (SK_VAO_ARO_RODA / 2) -
        (SK_LARG_VAO_PT_DIANT + 70)) + "|30~0|47~0|1.95~0|0|0";
        CMP_012_SUPP = 1;
        CMP_012_ID = "";

        //adiciona o tudo FINAL da saia lateral
        CMP_019_X = 0;
        CMP_019_Y = (SK_BAL_DIANT + SK_EE) + (SK_VAO_ARO_RODA / 2);
        CMP_019_Z = 30;
        CMP_019_A_X = 90;
        CMP_019_A_Y = 0;
        CMP_019_A_Z = 0;
        CMP_019_ISASSEMBLY = 0;
        CMP_019_NOME = "TB_30x47x1.95x0x0x" + (SK_EE - SK_VAO_ARO_RODA);
        CMP_019_DESC = CMP_019_NOME;
        CMP_019_PARAM = "FAM|COMPRIMENTO|ALTURA|LARGURA|ESP|ANGULO_A|ANGULO_B";
        CMP_019_PARAM_VALUE = "000527|" + ((SK_COMPRIMENTO -SK_LARG_EST_FIBRA_TRAS -
        SK_LARG_EST_FIBRA_DIANT - 1634) - (SK_BAL_DIANT + SK_EE + (SK_VAO_ARO_RODA / 2))) +
        "|30~0|47~0|1.95~0|0|0"; //1634 eh o tamanho do tubo final
        CMP_019_SUPP = 1;
        CMP_019_ID = "";
}
public void addContraventamento(){
        //adiciona o ultimo tubo vertical da linha das janelas - CENTRAL
        CMP_011_X = 23;
        CMP_011_Y = SK_COMPRIMENTO - SK_LARG_EST_FIBRA_TRAS - SK_LARG_EST_FIBRA_DIANT - 50;
        CMP_011_Z = SK_ALT_PEITORIL - SK_ALT_PERF_BAGUETE_LAT - 70;
        CMP_011_A_X = 0;
        CMP_011_A_Y = 0;
        CMP_011_A_Z = 0;
        CMP_011_ISASSEMBLY = 0;
```

```
CMP_011_NOME = "";
CMP_011_DESC = CMP_011_NOME;
CMP_011_PARAM = "";
CMP_011_PARAM_VALUE = "";
CMP_011_SUPP = 1;
CMP_011_ID = "000568";
CMP_020_X = 23;
CMP_020_Y = SK_POS_PRIM_COLUNA;
CMP_020_Z = SK_ALT_PEITORIL - SK_ALT_PERF_BAGUETE_LAT - 70;
CMP_020_A_X = 0;
CMP_020_A_Y = 0;
CMP_020_A_Z = 0;
CMP_020_ISASSEMBLY = 0;
CMP_020_NOME = "";
CMP_020_DESC = CMP_020_NOME;
CMP_020_PARAM = "";
CMP_020_PARAM_VALUE = "";
CMP_020_SUPP = 1;
CMP_020_ID = "000568";

CMP_021_X = 23;
CMP_021_Y = SK_POS_PRIM_COLUNA + SK_VAO_PADRAO + SK_LARG_COLUNA_SINO;
CMP_021_Z = SK_ALT_PEITORIL - SK_ALT_PERF_BAGUETE_LAT - 70;
CMP_021_A_X = 0;
CMP_021_A_Y = 0;
CMP_021_A_Z = 0;
CMP_021_ISASSEMBLY = 0;
CMP_021_NOME = "";
CMP_021_DESC = CMP_021_NOME;
CMP_021_PARAM = "";
CMP_021_PARAM_VALUE = "";
CMP_021_SUPP = 1;
CMP_021_ID = "000568";

if(QTD_JANELAS_PADRAO == 1) return;
CMP_022_X = 23;
CMP_022_Y = SK_POS_PRIM_COLUNA + ((SK_VAO_PADRAO + SK_LARG_COLUNA_SINO) * 2);
CMP_022_Z = SK_ALT_PEITORIL - SK_ALT_PERF_BAGUETE_LAT - 70;
CMP_022_A_X = 0;
CMP_022_A_Y = 0;
CMP_022_A_Z = 0;
CMP_022_ISASSEMBLY = 0;
CMP_022_NOME = "";
```

```
CMP_022_DESC = CMP_022_NOME;
CMP_022_PARAM = "";
CMP_022_PARAM_VALUE = "";
CMP_022_SUPP = 1;
CMP_022_ID = "000568";

if(QTD_JANELAS_PADRAO == 2) return;
CMP_023_X = 23;
CMP_023_Y = SK_POS_PRIM_COLUNA + ((SK_VAO_PADRAO + SK_LARG_COLUNA_SINO) * 3);
CMP_023_Z = SK_ALT_PEITORIL - SK_ALT_PERF_BAGUETE_LAT - 70;
CMP_023_A_X = 0;
CMP_023_A_Y = 0;
CMP_023_A_Z = 0;
CMP_023_ISASSEMBLY = 0;
CMP_023_NOME = "";
CMP_023_DESC = CMP_023_NOME;
CMP_023_PARAM = "";
CMP_023_PARAM_VALUE = "";
CMP_023_SUPP = 1;
CMP_023_ID = "000568";

if(QTD_JANELAS_PADRAO == 3) return;
CMP_024_X = 23;
CMP_024_Y = SK_POS_PRIM_COLUNA + ((SK_VAO_PADRAO + SK_LARG_COLUNA_SINO) * 4);
CMP_024_Z = SK_ALT_PEITORIL - SK_ALT_PERF_BAGUETE_LAT - 70;
CMP_024_A_X = 0;
CMP_024_A_Y = 0;
CMP_024_A_Z = 0;
CMP_024_ISASSEMBLY = 0;
CMP_024_NOME = "";
CMP_024_DESC = CMP_024_NOME;
CMP_024_PARAM = "";
CMP_024_PARAM_VALUE = "";
CMP_024_SUPP = 1;
CMP_024_ID = "000568";

if(QTD_JANELAS_PADRAO == 4) return;
CMP_025_X = 23;
CMP_025_Y = SK_POS_PRIM_COLUNA + ((SK_VAO_PADRAO + SK_LARG_COLUNA_SINO) * 5);
CMP_025_Z = SK_ALT_PEITORIL - SK_ALT_PERF_BAGUETE_LAT - 70;
CMP_025_A_X = 0;
CMP_025_A_Y = 0;
CMP_025_A_Z = 0;
```

```
        CMP_025_ISASSEMBLY = 0;
        CMP_025_NOME = "";
        CMP_025_DESC = CMP_025_NOME;
        CMP_025_PARAM = "";
        CMP_025_PARAM_VALUE = "";
        CMP_025_SUPP = 1;
        CMP_025_ID = "000568";
}

        addComponentesPadronizados();
        addTubosLongitudinais();
        addColunaSino();
        addArodeRoda();
        addTuboSaia();
        addContraventamento();
```

ANEXO B – Código-fonte conjunto janelas lateral esquerda

Código-fonte do *Template* Conjunto Janelas Lateral Esquerda para a aplicação *Automator*.

```
public void checkVariaveis(){
	CMP_ERRO = "";
	if(!SK_JANELA.equals("COLADA") && !SK_JANELA.equals("RODO")){
		CMP_ERRO = "Tipo de janela nao desenvolvida.";
	}
}

public void calculaVaoJanelas(){
	VAO_TOTAL_DISPONIVEL = SK_COMPRIMENTO - (SK_LARG_EST_FIBRA_DIANT + SK_LARG_EST_FIBRA_TRAS) - SK_LARG_COLUNA_SINO - SK_POS_PRIM_COLUNA; //128.35 eh da coluna sino final + largura do tubo
	QTD_JANELAS_PADRAO = (VAO_TOTAL_DISPONIVEL / (SK_VAO_JAN_PADRAO + SK_LARG_COLUNA_SINO)).intValue();
	RESTO = (VAO_TOTAL_DISPONIVEL % (SK_VAO_JAN_PADRAO + SK_LARG_COLUNA_SINO));
	}
	public void addJanelas(){
	if(SK_JANELA.equals("RODO")){
		criarJanelaCorrer();
		addUltimaJanelaCorrer();
	}else if(SK_JANELA.equals("COLADA")){
		criarJanelaColada();
		addUltimaJanelaColada();
	}
	}
	public void criarJanelaColada(){
	CMP_001_ID = "000583";
	CMP_001_PARAM = "";
	CMP_001_PARAM_VALUE = "";
	CMP_001_NOME = "";
	CMP_001_DESC = CMP_002_NOME;
	CMP_001_ISASSEMBLY = 0;
	CMP_001_A_X = 0;
	CMP_001_A_Y = -86.5;
	CMP_001_A_Z = -90;
	CMP_001_X = 16;
	CMP_001_Y = SK_POS_PRIM_COLUNA + (SK_LARG_COLUNA_SINO /2);
```

```
CMP_001_Z = SK_ALT_PEITORIL + 10;

//segunda janela
CMP_002_PARAM = "FAM|ALTURA|LARGURA|RAIO";
CMP_002_PARAM_VALUE = "000569|"+ SK_ALT_VAO_JANELA +"~0|"+ (SK_VAO_JAN_PADRAO + SK_LARG_COLUNA_SINO) +"~0|0";
CMP_002_NOME = "VIDRO_JAN_COL_ROD_"+ SK_ALT_VAO_JANELA +"x" + (SK_VAO_JAN_PADRAO + SK_LARG_COLUNA_SINO)+"x0";
CMP_002_DESC = CMP_002_NOME;
CMP_002_ISASSEMBLY = 0;
CMP_002_ID = "";
CMP_002_A_X = 0;
CMP_002_A_Y = 93.5;
CMP_002_A_Z = 90;
CMP_002_X = 16;
CMP_002_Y = getPosColSino(true)+(SK_LARG_COLUNA_SINO/2);
CMP_002_Z = SK_ALT_PEITORIL + 10;
if(--QTD_JANELAS_PADRAO == 0) return;

CMP_003_PARAM = "FAM|ALTURA|LARGURA|RAIO";
CMP_003_PARAM_VALUE = "000569|"+ SK_ALT_VAO_JANELA +"~0|"+ (SK_VAO_JAN_PADRAO + SK_LARG_COLUNA_SINO) +"~0|0";
CMP_003_NOME = "VIDRO_JAN_COL_ROD_"+ SK_ALT_VAO_JANELA +"x" + (SK_VAO_JAN_PADRAO + SK_LARG_COLUNA_SINO)+"x0";
CMP_003_DESC = CMP_003_NOME;
CMP_003_ISASSEMBLY = 0;
CMP_003_ID = "";
CMP_003_A_X = 0;
CMP_003_A_Y = 93.5;
CMP_003_A_Z = 90;
CMP_003_X = 16;
CMP_003_Y = getPosColSino(false)+(SK_LARG_COLUNA_SINO/2);
CMP_003_Z = SK_ALT_PEITORIL + 10;
if(--QTD_JANELAS_PADRAO == 0) return;

CMP_004_PARAM = "FAM|ALTURA|LARGURA|RAIO";
CMP_004_PARAM_VALUE = "000569|"+ SK_ALT_VAO_JANELA +"~0|"+ (SK_VAO_JAN_PADRAO + SK_LARG_COLUNA_SINO) +"~0|0";
CMP_004_NOME = "VIDRO_JAN_COL_ROD_"+ SK_ALT_VAO_JANELA +"x" + (SK_VAO_JAN_PADRAO + SK_LARG_COLUNA_SINO)+"x0";
CMP_004_DESC = CMP_004_NOME;
CMP_004_ISASSEMBLY = 0;
CMP_004_ID = "";
```

```
CMP_004_A_X = 0;
CMP_004_A_Y = 93.5;
CMP_004_A_Z = 90;
CMP_004_X = 16;
CMP_004_Y = getPosColSino(false)+(SK_LARG_COLUNA_SINO/2);
CMP_004_Z = SK_ALT_PEITORIL + 10;
if(--QTD_JANELAS_PADRAO == 0) return;

CMP_005_PARAM = "FAM|ALTURA|LARGURA|RAIO";
CMP_005_PARAM_VALUE = "000569|"+ SK_ALT_VAO_JANELA +"~0|"+ (SK_VAO_JAN_PADRAO + SK_LARG_COLUNA_SINO) +"~0|0";
CMP_005_NOME = "VIDRO_JAN_COL_ROD_"+ SK_ALT_VAO_JANELA +"x" + (SK_VAO_JAN_PADRAO + SK_LARG_COLUNA_SINO)+"x0";
CMP_005_DESC = CMP_005_NOME;
CMP_005_ISASSEMBLY = 0;
CMP_005_ID = "";
CMP_005_A_X = 0;
CMP_005_A_Y = 93.5;
CMP_005_A_Z = 90;
CMP_005_X = 16;
CMP_005_Y = getPosColSino(false)+(SK_LARG_COLUNA_SINO/2);
CMP_005_Z = SK_ALT_PEITORIL + 10;
if(--QTD_JANELAS_PADRAO == 0) return;

CMP_006_PARAM = "FAM|ALTURA|LARGURA|RAIO";
CMP_006_PARAM_VALUE = "000569|"+ SK_ALT_VAO_JANELA +"~0|"+ (SK_VAO_JAN_PADRAO + SK_LARG_COLUNA_SINO) +"~0|0";
CMP_006_NOME = "VIDRO_JAN_COL_ROD_"+ SK_ALT_VAO_JANELA +"x" + (SK_VAO_JAN_PADRAO + SK_LARG_COLUNA_SINO)+"x0";
CMP_006_DESC = CMP_006_NOME;
CMP_006_ISASSEMBLY = 0;
CMP_006_ID = "";
CMP_006_A_X = 0;
CMP_006_A_Y = 93.5;
CMP_006_A_Z = 90;
CMP_006_X = 16;
CMP_006_Y = getPosColSino(false)+(SK_LARG_COLUNA_SINO/2);
CMP_006_Z = SK_ALT_PEITORIL + 10;
if(--QTD_JANELAS_PADRAO == 0) return;
}

public void criarJanelaCorrer(){
CMP_001_ID = "000584";
```

```
CMP_001_PARAM = "";
CMP_001_PARAM_VALUE = "";
CMP_001_NOME = "";
CMP_001_DESC = CMP_002_NOME;
CMP_001_ISASSEMBLY = 0;
CMP_001_A_X = 0;
CMP_001_A_Y = 0;
CMP_001_A_Z = 0;
CMP_001_X = -50;
CMP_001_Y = SK_POS_PRIM_COLUNA +381;
CMP_001_Z = SK_ALT_PEITORIL - 328;

//segunda janela
CMP_002_PARAM = "FAM|ALTURA_VAO|LARGURA_VAO|TIPO_VIDRO";
CMP_002_PARAM_VALUE = "000539|"+ SK_ALT_VAO_JANELA +"~0|"+ SK_VAO_JAN_PADRAO +"~0|0";
CMP_002_NOME = "CJ_JAN_CORR_ROD_"+ SK_ALT_VAO_JANELA +"x" + SK_VAO_JAN_PADRAO;
CMP_002_DESC = CMP_002_NOME;
CMP_002_ISASSEMBLY = 1;
CMP_002_ID = "";
CMP_002_A_X = 0;
CMP_002_A_Y = 93;
CMP_002_A_Z = 90;
CMP_002_X = 150;
CMP_002_Y = getPosColSino(true);
CMP_002_Z = SK_ALT_PEITORIL;
if(--QTD_JANELAS_PADRAO == 0) return;

CMP_003_PARAM = "FAM|ALTURA_VAO|LARGURA_VAO|TIPO_VIDRO";
CMP_003_PARAM_VALUE = "000539|"+ SK_ALT_VAO_JANELA +"~0|"+ SK_VAO_JAN_PADRAO +"~0|0";
CMP_003_NOME = "CJ_JAN_CORR_ROD_"+ SK_ALT_VAO_JANELA +"x" + SK_VAO_JAN_PADRAO;
CMP_003_DESC = CMP_003_NOME;
CMP_003_ISASSEMBLY = 1;
CMP_003_A_X = 0;
CMP_003_A_Y = 93;
CMP_003_A_Z = 90;
CMP_003_X = 150;
CMP_003_Y = getPosColSino(false);
CMP_003_Z = SK_ALT_PEITORIL;
if(--QTD_JANELAS_PADRAO == 0) return;

CMP_004_PARAM = "FAM|ALTURA_VAO|LARGURA_VAO|TIPO_VIDRO";
CMP_004_PARAM_VALUE = "000539|"+ SK_ALT_VAO_JANELA +"~0|"+ SK_VAO_JAN_PADRAO +"~0|0";
CMP_004_NOME = "CJ_JAN_CORR_ROD_"+ SK_ALT_VAO_JANELA +"x" + SK_VAO_JAN_PADRAO;
```

```
CMP_004_DESC = CMP_004_NOME;
CMP_004_ISASSEMBLY = 1;
CMP_004_A_X = 0;
CMP_004_A_Y = 93;
CMP_004_A_Z = 90;
CMP_004_X = 150;
CMP_004_Y = getPosColSino(false);
CMP_004_Z = SK_ALT_PEITORIL;
if(--QTD_JANELAS_PADRAO == 0) return;

CMP_005_PARAM = "FAM|ALTURA_VAO|LARGURA_VAO|TIPO_VIDRO";
CMP_005_PARAM_VALUE = "000539|"+ SK_ALT_VAO_JANELA +"~0|"+ SK_VAO_JAN_PADRAO +"~0|0";
CMP_005_NOME = "CJ_JAN_CORR_ROD_"+ SK_ALT_VAO_JANELA +"x" + SK_VAO_JAN_PADRAO;
CMP_005_DESC = CMP_005_NOME;
CMP_005_ISASSEMBLY = 1;
CMP_005_A_X = 0;
CMP_005_A_Y = 93;
CMP_005_A_Z = 90;
CMP_005_X = 150;
CMP_005_Y = getPosColSino(false);
CMP_005_Z = SK_ALT_PEITORIL;
if(--QTD_JANELAS_PADRAO == 0) return;

CMP_006_PARAM = "FAM|ALTURA_VAO|LARGURA_VAO|TIPO_VIDRO";
CMP_006_PARAM_VALUE = "000539|"+ SK_ALT_VAO_JANELA +"~0|"+ SK_VAO_JAN_PADRAO +"~0|0";
CMP_006_NOME = "CJ_JAN_CORR_ROD_"+ SK_ALT_VAO_JANELA +"x" + SK_VAO_JAN_PADRAO;
CMP_006_DESC = CMP_006_NOME;
CMP_006_ISASSEMBLY = 1;
CMP_006_A_X = 0;
CMP_006_A_Y = 93;
CMP_006_A_Z = 90;
CMP_006_X = 150;
CMP_006_Y = getPosColSino(false);
CMP_006_Z = SK_ALT_PEITORIL;
CMP_006_ID = "";
if(--QTD_JANELAS_PADRAO == 0) return;
}

public double getPosColSino(boolean primeira){
if(primeira){
        POS_ULT_COL_SINO = SK_POS_PRIM_COLUNA + SK_LARG_COLUNA_SINO +
SK_VAO_JAN_PADRAO;
}else{
```

```
                POS_ULT_COL_SINO = POS_ULT_COL_SINO + SK_LARG_COLUNA_SINO +
SK_VAO_JAN_PADRAO;
        }
        return POS_ULT_COL_SINO;
        }

        public void addUltimaJanelaCorrer(){
        if(RESTO > 1320){
                //adiciona a ultima coluna sino, cortada.
                CMP_007_X = 150;
                CMP_007_Y = POS_ULT_COL_SINO + SK_LARG_COLUNA_SINO + 1320; //800 eh o tamanho minimo
de uma janela
                CMP_007_Z = SK_ALT_PEITORIL;
                CMP_007_A_X = 0;
                CMP_007_A_Y = 93;
                CMP_007_A_Z = 90;
                CMP_007_PARAM = "FAM|ALTURA_VAO|LARGURA_VAO|TIPO_VIDRO";
                CMP_007_PARAM_VALUE = "000539|"+ SK_ALT_VAO_JANELA +"~0|"+ 1320 +"~0|1";
                CMP_007_NOME = "CJ_JAN_CORR_ROD_"+ SK_ALT_VAO_JANELA +"x" + 1320;
                CMP_007_DESC = CMP_007_NOME;
                CMP_007_SUPP = 1;
                CMP_007_ISASSEMBLY = 1;
                CMP_007_ID = "";
                RESTO = 0;
        }else if(RESTO > 920){
                CMP_007_X = 150;
                CMP_007_Y = POS_ULT_COL_SINO + SK_LARG_COLUNA_SINO + 920; //800 eh o tamanho minimo
de uma janela
                CMP_007_Z = SK_ALT_PEITORIL + 2.3;
                CMP_007_A_X = 0;
                CMP_007_A_Y = 93;
                CMP_007_A_Z = 90;
                CMP_007_PARAM = "FAM|ALTURA_VAO|LARGURA_VAO|TIPO_VIDRO";
                CMP_007_PARAM_VALUE = "000539|"+ SK_ALT_VAO_JANELA +"~0|"+ 920 +"~0|1";
                CMP_007_NOME = "CJ_JAN_CORR_ROD_"+ SK_ALT_VAO_JANELA +"x" + 920;
                CMP_007_DESC = CMP_007_NOME;
                CMP_007_SUPP = 1;
                CMP_007_ISASSEMBLY = 1;
                CMP_007_ID = "";
                RESTO = 0;
        }else{
        //adiciona a ultima coluna sino, cortada.
                CMP_007_X = 150;
```

```
CMP_007_Y = 0;
CMP_007_Z = 0;
CMP_007_A_X = 0;
CMP_007_A_Y = 93;
CMP_007_A_Z = 90;
CMP_007_PARAM = "FAM|ALTURA_VAO|LARGURA_VAO|TIPO_VIDRO";
CMP_007_PARAM_VALUE = "000539|"+ SK_ALT_VAO_JANELA +"~0|"+ RESTO +"~0|1";
CMP_007_NOME = "CJ_JAN_CORR_ROD_"+ SK_ALT_VAO_JANELA +"x" + RESTO;
CMP_007_DESC = CMP_007_NOME;
CMP_007_SUPP = 1;
CMP_007_ISASSEMBLY = 1;
CMP_007_ID = "";
}
}

public void addUltimaJanelaColada(){
CMP_007_PARAM = "FAM|ALTURA|LARGURA|RAIO";
CMP_007_PARAM_VALUE = "000569|"+ SK_ALT_VAO_JANELA +"~0|"+ (RESTO) +"~0|0";
CMP_007_NOME = "VIDRO_JAN_COL_ROD_"+ SK_ALT_VAO_JANELA +"x" + (RESTO)+"x0";
CMP_007_DESC = CMP_007_NOME;
CMP_007_ISASSEMBLY = 0;
CMP_007_ID = "";
CMP_007_A_X = 0;
CMP_007_A_Y = 93.5;
CMP_007_A_Z = 90;
CMP_007_X = 16;
CMP_007_Y = POS_ULT_COL_SINO + SK_VAO_JAN_PADRAO + SK_LARG_COLUNA_SINO +
(SK_LARG_COLUNA_SINO/2);
CMP_007_Z = SK_ALT_PEITORIL + 10;
}
checkVariaveis();
calculaVaoJanelas();
addJanelas();
```

UPF
UNIVERSIDADE
DE PASSO FUNDO

UPF Campus I - BR 285, São José
Passo Fundo - RS - CEP: 99052-900
(54) 3316 7000 - www.upf.br