

UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Bruno da Silva Barreto

DISSEMINANDO O USO DO CONTROLE DE CARGA NO PLANEJAMENTO E
CONTROLE DA PRODUÇÃO: Modelagem utilizando um software de simulação comercial

SÃO CARLOS

2018

BRUNO DA SILVA BARRETO

DISSEMINANDO O USO DO CONTROLE DE CARGA NO PLANEJAMENTO E
CONTROLE DA PRODUÇÃO: Modelagem utilizando um software de simulação comercial

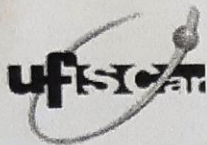
Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de São Carlos, como parte dos requisitos para obtenção do Título de Mestre em Engenharia de Produção

Linha de pesquisa: Planejamento e Controle de Sistemas Produtivo

Orientadora: Profa. Dra. Juliana Keiko Sagawa.

SÃO CARLOS

2018

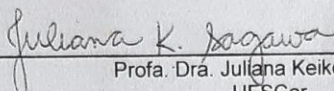


UNIVERSIDADE FEDERAL DE SÃO CARLOS

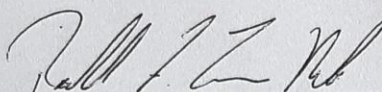
Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Engenharia de Produção

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado do candidato Bruno da Silva Barreto, realizada em 25/04/2018:

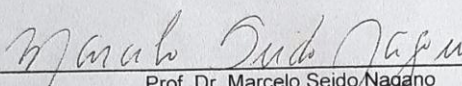


Profa. Dra. Juliana Keiko Sagawa
UFSCar




Prof. Dr. Roberto Fernandes Tavares Neto
UFSCar





Prof. Dr. Marcelo Seido Nagano
USP

CONFERE COM
O ORIGINAL


Regilene Vieira Inácio
ASSIST. ADMINISTRATIVA

AGRADECIMENTOS

À Deus por me abençoar, iluminar o meu caminho e me dar forças para sempre seguir em frente.

Aos meus pais, Enilton e Vanda Lúcia, meus irmãos, sobrinha e avó, Adrian, Matheus, Maria Eduarda e Ivandir, que sempre acreditaram em mim e me apoiaram incondicionalmente.

Aos meus tios, Wilson e Sandra, pelo apoio e ajuda em momentos de dificuldades, e Marcelo e Elizângela, por se fazerem presentes e serem o meu apoio aqui no estado de São Paulo.

Ao meu primo Arthur, a quem considero como irmão, por compartilharmos nossas dificuldades e dividirmos as mesmas experiências de mestrados.

Às minhas amigas Isis Souza e Letícia Isaac, por sempre me presentear com palavras de conforto e incentivo.

À Thais Klemz, que mesmo à distância se fez muito presente desde o início do meu mestrado, não me deixando nunca me sentir só.

À Nayara Cremasco pelo apoio e suporte.

À Juliana Archanjo, a quem eu já conhecia desde as aulas de francês e que em 2015 topou o desafio de ser a minha *coaching*. Hoje é muito mais que isso, é minha amiga, mentora e incentivadora.

Ao meu grande amigo Pedro Bandeira, pelo apoio, sabedoria e experiência compartilhada. Por sempre me incentivar e me ajudar a encontrar alternativas para os meus desafios.

Aos meus amigos do Churrasco Futebol Arte, por compartilharmos momentos de distração e confraternização aqui em São Carlos.

À minha orientadora Juliana Keiko Sagawa, por todo incentivo, dedicação, profissionalismo e contribuições para o meu crescimento como mestrando e pessoa.

Aos integrantes do meu grupo de pesquisa, por nós denominado PlacoPeq: Taíse, Fernanda, Luciano, Letícia, Luiz, Isabela, Renata, Nayara Cardoso, Denise e Fernando. Por serem a minha família aqui em São Carlos, dividindo as angústias e compartilhando as emoções, alegrias e superações. Galera, sem vocês a minha estadia aqui em São Carlos não seria a mesma.

Aos docentes e funcionários do Departamento de Engenharia de Produção da UFSCar.

RESUMO

O objetivo desta pesquisa é construir uma estrutura de conversão (*framework*) demonstrando detalhadamente como simular a abordagem do Controle de Carga (WLC) em um *software* comercial de simulação de eventos discretos. Justifica-se este estudo uma vez que, embora haja diversas pesquisas sobre essa abordagem, não há estudos ou tutoriais que demonstrem detalhadamente como modelar e simular essa abordagem nesses tipos de *softwares*, que por serem comerciais, apresentam fácil aquisição. Este estudo utiliza o método de modelagem e simulação. O principal resultado é a descrição da construção da estrutura de conversão do Controle de Carga em um *software* comercial de simulação de eventos discretos, no qual em um primeiro momento a lógica do problema é desenvolvida no *software* MATLAB, onde baseado em um modelo de chão de fábrica apresentado por Land (2006), é elaborado o algoritmo conceitual do WLC para verificar a consistência da lógica criada antes de fazer a conversão para o *software* de simulação. Por fim, a lógica do WLC é desenvolvida e são construídos e simulados os cenários baseados na literatura de referência. Ao descrever detalhadamente o desenvolvimento de cada etapa da construção dessa estrutura de conversão, este trabalho permite ao leitor uma melhor compreensão do desenvolvimento e da forma de replicar a abordagem do WLC. Possibilita também a construção de diferentes cenários que operem baseados nas políticas do WLC em busca de resultados mais aderentes as necessidades do chão de fábrica modelado, contribuindo para uma situação real. Esse trabalho contribui com o ambiente empresarial e acadêmico, pois facilita a replicação e aplicação dessa estrutura em outros ambientes fabris, permitindo reduzir o tempo do projeto, de avaliação para montar o modelo base, possibilitando o foco diretamente nas comparações e adequações da ambiente avaliado por parte de gestores e/ou pesquisadores.

Palavras-chave: Controle de Carga, Simulação, *Jobshop*, *Make to Order*, Estrutura de Conversão.

ABSTRACT

The aim of this study is to make a framework of a Workload Control (WLC) approach in commercial software for discrete event simulation. This study is justified because although there are a lot of researches on this approach, there are no studies or tutorials that demonstrate how to model and simulate which types of commercial's software. This study uses the modeling and simulation method. The main result is a description of the construction of the framework showing in detail how to simulate the WLC approach in a commercial software for discrete event simulation, in which a first moment of the logic of the problem is developed in MATLAB's system, where it is based on a shop floor model researching of Land (2006), the conceptual algorithm of the WLC is developed to verify the consistency of the invented part before making a conversion to the simulation software. Finally, the WLC logic is developed and are constructed and simulated on a simulation scenarios from of literature. It is also possible to build different scenarios that operate based on WLC policies in search of leaner outcomes such as the needs of the modeled factory floor, contributing to a real situation. This work in the business and academic environment, which facilitates the replication and application of framework in other environments, facilitating the progress of the project, evaluation for the assembly of the base model, allowing the direct focus on the comparisons and adaptations of the environment of managers and / or researchers.

Keywords: Workload Control, Simulation, Jobshop, Make to Order, Framework.

LISTA DE FIGURAS

Figura 1 – Modelo de transformação de um sistema de produção.....	18
Figura 2 – Características básicas dos sistemas produtivos	18
Figura 3 – Comportamento de alguns fatores importantes em função da estratégia de resposta á demanda dos sistemas de produção.....	21
Figura 4 – Modelo Funil.....	25
Figura 5 – Componentes do <i>Lead Time</i>	26
Figura 6 - Hierarquias das cargas de trabalho.....	27
Figura 7 – Modelo geral.....	29
Figura 8 - Representação simplificada do algoritmo de controle de carga.....	33
Figura 9 – Integração entre controles de entrada e saída dentro da liberação de ordens do WLC.....	37
Figura 10 – Componentes principais do modelo de simulação	43
Figura 11 - Parâmetros do FlexSim	47
Figura 12 - Ambiente de simulação	48
Figura 13 - Propriedades do objeto <i>source</i>	49
Figura 14 - Propriedades do objeto <i>queue</i>	50
Figura 15 - Propriedades do objeto <i>processor</i>	51
Figura 16 - Propriedades do objeto <i>sink</i>	52
Figura 17 - Ambiente de programação dentro do objeto.....	52
Figura 18 - Fluxograma detalhado do funcionamento do modelo traduzido	54
Figura 19 – Visão geral do modelo utilizado em <i>FlexSim</i>	60
Figura 20 – Etiquetas do item (ordem) para salvar atributos.....	62
Figura 21 – Detalhe da Tabela cargas	62
Figura 22 – Configuração do source “Chegada”	63
Figura 23 – Etiquetas do item para salvar atributos.....	64
Figura 24 – Definição do tempo de processamento no item, operação 1	64
Figura 25 – Definição do tempo de processamento no item, outras operações.....	65
Figura 26 – Transferência do tempo de processamento e cálculo da data de liberação planejada.....	66
Figura 27 – Definição de tempo de chegada e data de entrega na interface gráfica	67
Figura 28 – Implementação da Abertura periódica do <i>pool</i>	67
Figura 29 – Tela de propriedades do <i>pool</i>	69
Figura 30 – Classificação das ordens dentro do <i>pool</i>	70
Figura 31 – Definição do envio da ordem dentro do <i>pool</i>	71
Figura 32 – Avaliação da carga dentro do <i>pool</i>	72
Figura 33 – Configuração do <i>Processor1</i> temporário para reingresso no <i>pool</i>	73
Figura 34 – Configuração da fila referente a cada máquina.....	74
Figura 35 – Configuração do objeto máquina (<i>processor</i>).....	75
Figura 36 – Contador de entidades processadas por máquina: a) vista na aba <i>Triggers</i> e b) vista na aba <i>Labels</i>	76
Figura 37 – Substração da carga já processada pela máquina.....	77
Figura 38 – Configuração do <i>sink</i> saída para suas próprias etiquetas	78
Figura 39 – Painel dos gráficos da etiqueta “Throughput Total”, no <i>Sink</i> “Saida”.....	78

Figura 40 – Painel dos gráficos da etiqueta “ <i>Throughput</i> Chão de Fábrica”, no <i>sink</i> “Saida”	79
Figura 41 - Configuração da rodada de simulação (modelo base)	80
Figura 42 – WIP do Pool e das filas de cada máquina.....	81
Figura 43 – Estabilização do tempo de atravessamento total.....	81
Figura 44 – Estabilização do tempo de fluxo na simulação	82
Figura 45 – Utilização das máquinas no modelo	82
Figura 46 – Resultado do tempo de atravessamento total do modelo base.....	83
Figura 47 – Resultado do tempo de fluxo do modelo base	83
Figura 48 – Cenários com diferentes normas	84
Figura 49 – Replicações por cenário	85
Figura 50 – Tempo de atravessamento total.....	86
Figura 51 – Tempo de atravessamento do chão de fábrica	87

LISTA DE TABELAS

Tabela 1 – Regras de sequenciamento	24
Tabela 2 – Parâmetros do ajuste de capacidade.....	37
Tabela 3 - Tempo de atravessamento da literatura versus autor	84

LISTA DE QUADROS

Quadro 1 - Hierarquia de cargas de trabalho e o impacto de um novo trabalho sobre os comprimentos de carga de trabalho.....	28
Quadro 2 – Características do sistema modelado utilizado como referência	44
Quadro 3 – Descrição das componentes do sistema modelado	45
Quadro 4 – Conversão do algoritmo geral de controle de carga em um modelo implementado em software comercial (<i>FlexSim</i>).....	57

LISTA DE ABREVIATURAS E SIGLAS

CEM	- <i>Customer Enquiry Management</i>
ConWIP	- <i>Constant Work in Process</i>
LUMS COR	- <i>Lancaster University Management School Corrected Order Release</i>
LUMS OR	- <i>Lancaster University Management School Order Release</i>
MTO	- <i>Make to Order</i>
PCP	- <i>Planejamento e Controle da Produção</i>
PME	- <i>Pequenas e médias empresas</i>
PWL	- <i>Planned Workload Length</i>
RWL	- <i>Release Workload Length</i>
SLAR	- <i>Superfluous Load Avoidance Release</i>
TWL	- <i>Total Workload Length</i>
WCEDD	- <i>Work Center Workload Trigger Earliest Due Date</i>
WIP	- <i>Work in Process</i>
WLC	- <i>Workload Control</i>

SUMÁRIO

1	INTRODUÇÃO	14
1.1.	FORMULAÇÃO DO PROBLEMA	15
1.2.	OBJETIVO DA PESQUISA	15
1.2.1.	Objetivos específicos	16
1.3.	JUSTIFICATIVA DA PESQUISA	16
1.4.	ORGANIZAÇÃO DO TRABALHO	16
2	FUNDAMENTAÇÃO TEÓRICA	17
2.1.	SISTEMAS DE PRODUÇÃO (SP)	17
2.2.	PLANEJAMENTO E CONTROLE DA PRODUÇÃO (PCP)	21
2.3.	PROGRAMAÇÃO DE OPERAÇÕES	22
2.4.	CONCEITO E HISTÓRICO SOBRE O CONTROLE DE CARGA (WLC)	24
2.5.	HIERARQUIA DO WLC	27
2.6.	ETAPAS DE LIBERAÇÃO	28
2.7.	CLASSIFICAÇÃO DOS MÉTODOS DE LIBERAÇÃO DE ORDEM	29
2.7.1.	Método de liberação periódica	31
2.7.2.	Métodos LUMS OR e LUMS COR	34
2.7.3.	Método com controle de entrada e saída	35
2.8.	PESQUISAS RECENTES SOBRE O CONTROLE DE CARGA	38
3	MÉTODO DE PESQUISA	41
3.1.	COMPONENTES DO SISTEMA A SER SIMULADO NA PESQUISA	42
3.2.	CONSTRUÇÃO DO SISTEMA MODELADO NA LINGUAGEM DO <i>SOFTWARE</i>	45
3.3.	DESENHO DO EXPERIMENTO	46
4	MODELAGEM E CONSTRUÇÃO DA ESTRUTURA DE CONVERSÃO	47
4.1.	O <i>SOFTWARE</i> FLEXSIM	47
4.2.	VISÃO MACRO DO MODELO	53
4.3.	CONSTRUINDO ESTRUTURA DE CONVERSÃO NA FERRAMENTA <i>FLEXSIM</i>	60
4.4.	CRIAÇÃO DAS ORDENS	61
4.5.	ABERTURA PERIÓDICA DO <i>POOL</i>	67
4.6.	LIBERAÇÃO DAS ORDENS	68
4.7.	EXECUÇÃO DA ORDEM E ATUALIZAÇÃO DA CARGA DE CADA MÁQUINA	73
4.8.	FINALIZAÇÃO DAS ORDENS E COLETA DE INDICADORES	77
5	RESULTADOS	80
6	CONSIDERAÇÕES FINAIS	88
	REFERÊNCIAS	90
	APÊNDICE A.1: ALGORITMO BÁSICO DE LIBERAÇÃO DE ORDENS DO WLC EM UM AMBIENTE DE PROGRAMAÇÃO SIMPLES, COM LINGUAGEM ESTRUTURADA E DE ALTO NÍVEL (PARTE 1).	93
	APÊNDICE A.2: ALGORITMO BÁSICO DE LIBERAÇÃO DE ORDENS DO WLC EM UM AMBIENTE DE PROGRAMAÇÃO SIMPLES, COM LINGUAGEM ESTRUTURADA E DE ALTO NÍVEL (PARTE 2).	94
	APÊNDICE A.3: ALGORITMO BÁSICO DE LIBERAÇÃO DE ORDENS DO WLC EM UM AMBIENTE DE PROGRAMAÇÃO SIMPLES, COM LINGUAGEM ESTRUTURADA E DE ALTO NÍVEL (PARTE 3).	95
	APÊNDICE A.4: ALGORITMO BÁSICO DE LIBERAÇÃO DE ORDENS DO WLC EM UM AMBIENTE DE PROGRAMAÇÃO SIMPLES, COM LINGUAGEM ESTRUTURADA E DE ALTO NÍVEL (PARTE 4).	96

APÊNDICE B: ALGORITMO NA ENTIDADE “CHEGADA”, ABA “TRIGGERS”, CAMPO “ONCREATION”	97
APÊNDICE C: ALGORITMO ENTIDADE “TIC”, ABA “TRIGGERS”, CAMPO “ONEXIT”	102
APÊNDICE D: ALGORITMO ENTIDADE “POOL”, ABA “TRIGGERS”, CAMPO “ONENTRY”	103
APÊNDICE E: ALGORITMO ENTIDADE “POOL”, ABA “TRIGGERS”, CAMPO “ONEXIT”	104
APÊNDICE F: ALGORITMO ENTIDADE “POOL”, ABA “FLOW”, CAMPO “SEND TO PORT”	105
APÊNDICE G: ALGORITMO ENTIDADE “FILA EST 1 (QUEUE)”, ABA “TRIGGERS”, CAMPO “ONENTRY”	106
APÊNDICE H: ALGORITMO ENTIDADE “MÁQUINA 3 (PROCESSOR)”, ABA “TRIGGERS”, CAMPO “ONENTRY”	107
APÊNDICE I: ALGORITMO ENTIDADE “MÁQUINA 3 (PROCESSOR)”, ABA “TRIGGERS”, CAMPO “ONEXIT”	109
APÊNDICE J: ALGORITMO ENTIDADE “SAÍDA (SINK)”, ABA “TRIGGERS”, CAMPO “ONENTRY”	110

1 INTRODUÇÃO

O funcionamento de qualquer empresa passa pela adoção de um sistema de produção (SP) que irá realizar as suas operações e gerar seus produtos, serviços ou a combinação de ambos. Tornar o sistema de produção efetivo, ou seja, atingindo aos objetivos propostos pela empresa de forma eficiente, requer um grande esforço gerencial e uma busca constante do aperfeiçoamento desse sistema.

Cada sistema de produção possuirá suas características próprias e específicas cujo a análise dessas características permitirá a sua classificação. Esta classificação é relevante na escolha da abordagem de PCP mais adequada ao sistema de produção da empresa, Uma vez que melhora o seu desempenho fabril, seja por meio do aumento da capacidade de produção, da diminuição do tempo de atravessamento ou de outros fatores.

Este trabalho destacará a abordagem de Controle de Carga (*Workload Control - WLC*). Segundo Hendry, Kingsman e Cheung (1998), o conceito do WLC trata-se de um método de planejamento e controle da produção projetado para controlar filas em um ambiente de fabricação *jobshop*, no qual a sua importância reside na necessidade de manter este tipo de ambiente de fabricação flexível em empresas de fabricação MTO. Thürer e Godinho Filho (2012) destacam também que o WLC é reconhecidamente um método com potencial para trazer melhorias no *lead time*, no nível de estoque em processo (*Work in Process - WIP*) e na pontualidade de entregas nesses ambientes MTO e com uma configuração *jobshop*.

As classificações dos sistemas de produção, que serão mais exploradas no capítulo 2, podem ser baseadas nos tipos de produtos, nos tipos de processos e quanto a estratégia de resposta à demanda. Dentre as classificações quanto aos tipos de produtos e processos, está a do sistema de produção intermitente *jobshop*, no qual os itens fabricados em um setor produtivo não possuem o mesmo roteiro de fabricação (FERNANDES E GODINHO FILHO, 2010). Já referente a classificação da estratégia de resposta à demanda, o sistema de produção pode ser classificado como: fabricação por encomenda (*make to order - MTO*), no qual o pedido só começará a ser processado após a confirmação do pedido pelo cliente.

Deve-se destacar que o WLC foi originalmente concebida para um ambiente MTO e *jobshop*, que apresenta programação detalhada e complexa. As características dessa abordagem são mais adequadas a este tipo de ambiente, ao do que o método *kanban*, por

exemplo, que não é adequado à ambientes com demanda muito instável e alta variedade de produtos (Vollmann *et al.*, 2006).

Diversas pesquisas, como as de Hendry e Kingsman (1989), Hendry, Kingsman e Cheung (1998), Thüerer *et al.* (2014), Battaglia *et al.* (2016), entre outras, estudam e analisam o desempenho do WLC nesse ambiente fabril descrito, mostrando suas vantagens, limitações e condições de utilização. A grande maioria desses estudos utilizam o método de simulação para gerar resultados que lhes permitam obter as conclusões acerca da utilização dessa abordagem.

Para a modelagem e simulação desse SP utilizando os conceitos do WLC, este trabalho irá utilizar um *software* de simulação comercial (FlexSim) que possui grande abrangência em diversos países, o que possibilita maior acesso ao *software*, contribuindo para que a solução desenvolvida seja possível de ser replicada de forma mais íntegra. Os modelos apresentados na literatura são relativamente difíceis de serem replicados por gestores, pois são, em sua grande maioria, desenvolvidos em sistemas restritos a um centro de pesquisa, como em Land e Gaalman (1996), Land e Gaalman (1998), Oosterman, Land e Gaalman (2000), Thüerer *et al.* (2015), entre outros.

Diante desse cenário de estudos sobre o WLC, o intuito deste trabalho é o de criar uma estrutura de conversão desta abordagem para um *software* comercial de simulação de eventos discretos, demonstrando como os conceitos dessa abordagem podem ser modelados com a finalidade de gerar simulações que permitam a compreensão e análise dos resultados que a utilização do WLC traz para o chão de fábrica.

1.1. FORMULAÇÃO DO PROBLEMA

O problema de pesquisa desse trabalho se desenvolveu baseado na circunstância exposta: *Como implantar um modelo de Controle de Carga em um software comercial de simulação de eventos discretos?*

1.2. OBJETIVO DA PESQUISA

O objetivo geral proposto é o de construir uma estrutura de conversão demonstrando detalhadamente como simular a abordagem do controle de carga (WLC) em um *software* comercial de simulação de eventos discretos.

1.2.1. Objetivos específicos

Para se atingir o objetivo proposto, são definidos quatro objetivos específicos:

- (i) Testar o algoritmo básico de liberação de ordens do WLC em um ambiente de programação simples, com linguagem estruturada e de alto nível.
- (ii) Realizar a conversão do algoritmo básico para um *software* de simulação de eventos discretos.
- (iii) Descrever os elementos necessários para a conversão realizada, incluindo a definição dos objetos utilizados, funções e códigos personalizados.

1.3. JUSTIFICATIVA DA PESQUISA

Alguns softwares comerciais de simulação de eventos discretos, como o adotado nesse trabalho, possuem uma grande capacidade descritiva, o que permite a modelagem e simulação de um cenário de chão de fábrica bem próximo ao real. Assim, a construção dessa estrutura de conversão da abordagem do WLC neste tipo de *software* permite uma melhor compreensão dos resultados que a adoção dessa abordagem pode gerar em um sistema fabril. Ao se descrever com clareza o desenvolvimento de cada etapa desse processo de conversão, facilita-se a replicação desses processos em modelos e sistemas similares.

As empresas com política MTO e característica de *jobshop*, que são as abordadas nesse estudo, possuem poucas alternativas de métodos que visam à melhoria do seu setor de Planejamento e Controle da Produção (THÜRER; GODINHO FILHO, 2012). Sendo assim, a descrição do passo a passo da estrutura de conversão desenvolvida, facilita a criação de vários cenários de sistemas fabris com características diferentes, permitindo suas respectivas análises e ocasionando diversas contribuições para uma situação real.

1.4. ORGANIZAÇÃO DO TRABALHO

Capítulo 2 – Fundamentação Teórica: abordar as bases teóricas fundamentais para o estudo. Fez-se um levantamento histórico sobre o surgimento do tema e abordou-se os conceitos, etapas e métodos presentes no WLC.

Capítulo 3 – Método da Pesquisa: destaca os procedimentos metodológicos utilizados, desde a seleção das referências do capítulo anterior até a coleta, tratamento dos

dados e a forma que foi construído a estrutura de conversão do WLC para um modelo de simulação de eventos discretos.

Capítulo 4 – Desenvolvimento: descreve detalhadamente todas as atividades desenvolvidas para se construir uma estrutura de conversão do Controle de Carga (WLC) para ser modelado e simulado em um *software* de simulação de eventos discretos. Por fim, estão descritos os resultados de desempenho do WLC em diferentes cenários de produção, baseados nas variações de alguns parâmetros.

Capítulo 5 – Resultados: demonstra a estrutura de conversão construída da abordagem de controle de carga dentro do *software* de simulação *FlexSim*.

Capítulo 6 – Considerações Finais: traz as considerações relevantes para a criação da estrutura de conversão, bem como as vantagens, desvantagens e restrições encontradas para a realização deste trabalho. Por fim, discute sobre possíveis estudos futuros sobre o WLC que este trabalho possibilitará e que seguem como lacuna na literatura.

2 FUNDAMENTAÇÃO TEÓRICA

2.1. SISTEMAS DE PRODUÇÃO (SP)

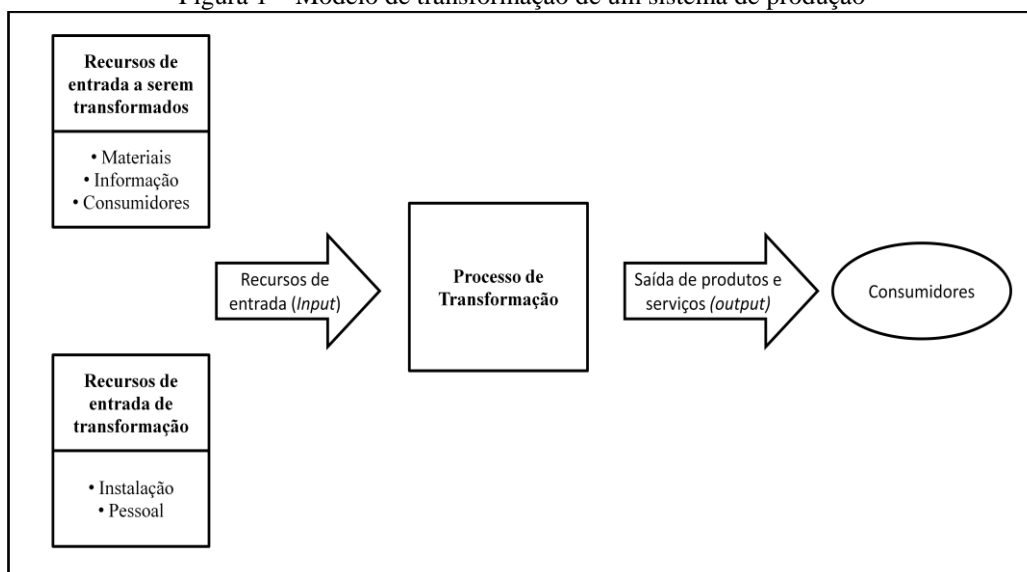
Antes da escolha sobre qual tipo de abordagem de planejamento e controle da produção (PCP) utilizar em um ambiente produtivo, é importante conhecer e classificar bem o ambiente no qual uma determinada empresa está inserida.

Empresas, seja de bens ou serviços, geralmente são estudadas como um sistema que transforma, via um processamento, entradas (insumos) em saídas (produtos) úteis aos clientes. Esse sistema é chamado de Sistema Produtivo (TUBINO, 2000), cuja Figura 1 demonstra o modelo genérico. Além de ser um sistema transformador, o sistema de produção também precisa agregar valor ao que está sendo transformado (SIPPER E BULFIN, 1977).

Embora o funcionamento padrão do sistema de produção seja o mesmo para todas as empresas, cada uma terá em seus sistemas de produção características próprias que os diferem. É importante saber diferenciar essas características para poder classificá-los corretamente.

Segundo Tubino (2000), a classificação dos sistemas produtivos tem por finalidade facilitar o entendimento das características inerentes a cada sistema de produção e sua relação com a complexidade das atividades de planejamento e controle destes sistemas.

Figura 1 – Modelo de transformação de um sistema de produção



Fonte: Slack, Chambers & Johnston (2009)

Existe um grande número de classificações de sistemas de produção encontrados na literatura (FERNANDES E GODINHO FILHO, 2010). Tubino (2000) classifica o sistema de produção em quatro tipos: sistemas contínuos, em massa, em lotes e sob encomenda. Esse autor mencionado baseia a classificação do SP de acordo ao grau de padronização dos produtos e conseqüente volume de produção demandado pelo mercado. A Figura 2 apresenta as características básicas desses sistemas produtivos.

Figura 2 – Características básicas dos sistemas produtivos



Fonte: Tubino (2000)

Outra classificação apresentada por Johnson e Montgomery (1974), diferencia os tipos dos sistemas de produção baseados em tipos de produto e processo, obtendo assim quatro classificações diferentes: sistema contínuo, intermitente, grande projeto e sistemas puros de estoques.

Nos sistemas contínuos são produzidos grandes volumes de produtos similares e de poucas famílias.

Os sistemas intermitentes caracterizam-se pela diversidade dos produtos fabricados, ocorrendo frequentes mudanças nos estágios produtivos de um produto para o outro. Esses sistemas são subdivididos em duas categorias:

- a) Sistema intermitente *flowshop*, onde todos os itens feitos em uma linha possuem a mesma sequência de operações nas diversas máquinas;
- b) Sistema intermitente *jobshop*, em que os itens fabricados em um setor produtivo não têm o mesmo roteiro de fabricação.

Os sistemas de grande projeto caracterizam-se por fabricar produtos complexos e especiais, muitas vezes únicos.

Por fim, têm-se os sistemas puros de estoques, caracterizando-se como um sistema de fluxo de materiais, onde itens são comprados, estocados, distribuídos e revendidos, sem que haja fase de processamento.

Segundo Komesu (2015), uma classificação mais específica é baseada em um problema de programação da produção (no qual as restrições tecnológicas são determinadas principalmente pelos fluxos das operações das diversas tarefas das máquinas) e a medida de desempenho da programação.

MacCarthy e Liu (1993) apresentam essa classificação em oito tipos:

- i. *Jobshop*: cada tarefa possui sua própria sequência de processamento nas máquinas;
- ii. *Flowshop*: todas as tarefas têm a mesma sequência de processamento nas máquinas;
- iii. *Open Shop*: não há sequência específica ou preestabelecida para as tarefas serem processadas nas máquinas;

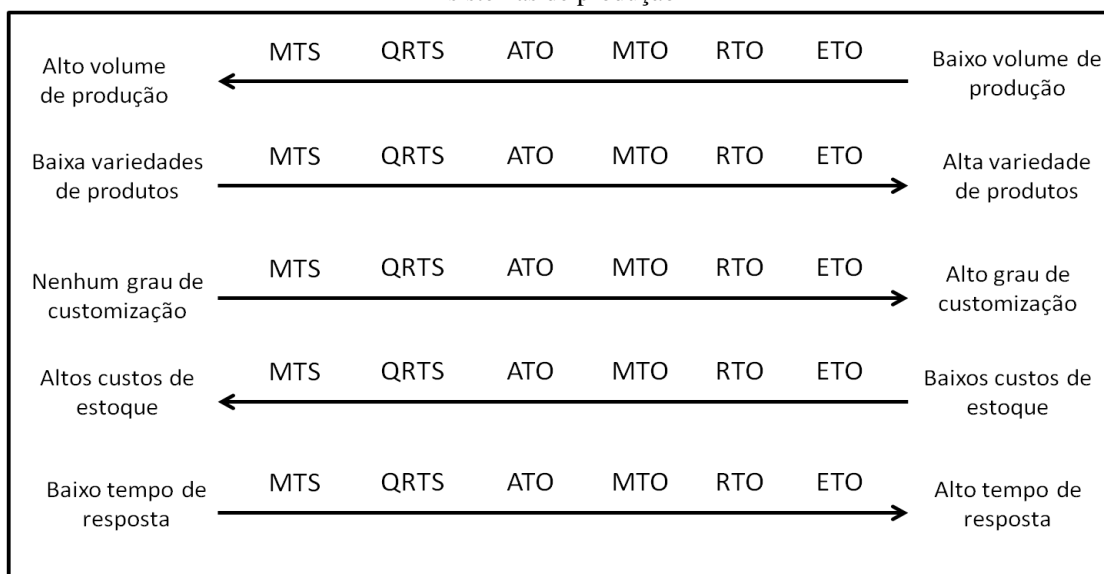
- iv. *Flowshop* Permutacional: é um *flowshop* no qual em cada máquina a sequência da tarefa é a mesma;
- v. Máquina única: existe uma única máquina disponível para o processamento das tarefas;
- vi. Máquinas paralelas: estão disponíveis diversas máquinas, geralmente idênticas, para as mesmas operações;
- vii. *Jobshop* com máquinas múltiplas: é um *jobshop* no qual existe um conjunto de máquinas paralelas em cada estágio de produção;
- viii. *Flowshop* com múltiplas máquinas: é um *flowshop* no qual existe um conjunto de máquinas paralelas em cada estágio de produção.

Por fim, Fernandes e Godinho Filho (2010) destacam a importância da classificação do sistema de produção baseada na estratégia de resposta à demanda, apresentando seis tipos de classificações:

- i. Produção para estoque com base na previsão de demanda (*make to stock-MTS*);
- ii. Produção para estoque com base em uma rápida reposição de estoque (*quick response to stock – QRTS*);
- iii. Montagem sob encomenda (*assembly to order-ATO*);
- iv. Fabricação sob encomenda, com a presença de estoques e insumos (*make to order-MTO*);
- v. Recursos insumos sob encomenda (*resources to order - RTO*);
- vi. Projeto sob encomenda (*engineering to order - ETO*).

A Figura 3 apresenta o comportamento de fatores importantes na produção como: volume, variedade, grau de customização, custos de estoque e tempo de resposta, frente à escolha dessas estratégias de resposta à demanda.

Figura 3 – Comportamento de alguns fatores importantes em função da estratégia de resposta á demanda dos sistemas de produção



Fonte: Fernandes e Godinho Filho (2010)

Diante dessas classificações apresentadas, percebe-se como um sistema de produção pode ser complexo e demonstra o grau de importância que uma escolha adequada da abordagem de produção pode beneficiar o seu desempenho, tal qual a decisão mais adequada sobre a forma de programar as operações que são realizadas pelo sistema de produção, respeitando as suas características.

2.2. PLANEJAMENTO E CONTROLE DA PRODUÇÃO (PCP)

As atividades de PCP envolvem uma série de decisões com o objetivo de definir o que, quanto e quando produzir, comprar e entregar, além de quem e/ ou onde e/ou como produzir (FERNANDES E GODINHO FILHO, 2010).

O grau de complexidade de cada uma destas atividades está intimamente relacionado ao tipo de sistema produtivo no qual o PCP irá desenvolvê-las, cabendo ao mesmo levantar e avaliar as características do sistema em questão para, a partir disto, buscar ferramentas adequadas para cada situação específica (TUBINO, 2009).

Segundo Torres (1999), os principais objetivos de desempenho do PCP relatados na literatura são:

- Melhoria do desempenho no atendimento da demanda no prazo;

- Redução dos tempos de atravessamento;
- Maximização da utilização da capacidade produtiva dos recursos;
- Manutenção dos níveis de estoque desejados.

A melhoria de desempenho no atendimento da demanda no prazo está intrinsecamente ligada ao grau de confiança que a empresa terá com os seus clientes.

A redução dos tempos de atravessamento permitirá tornar a produção mais dinâmica e flexível em relação ao tempo prometido com o recebimento de novas ordens.

A maximização da utilização da capacidade produtiva dos recursos indica que os recursos estão operando em seu nível máximo, ou seja, estão fabricando um número máximo de unidades que podem produzir em um período, com o mínimo de ociosidade possível. A manutenção dos níveis de estoques desejados, sendo eles de estoques finais ou intermediários, chamados de estoques em processo (*work in process - WIP*), permite à empresa se programar melhor quanto ao prazo de entrega dos seus produtos e reduz os custos ligados à manutenção dos estoques.

O PCP é o setor de apoio, dentro do sistema produtivo, com base no desenvolvimento de quatro funções: Planejamento Estratégico da Produção (longo prazo), Planejamento Mestre da Produção (médio prazo) Programação da Produção (curto prazo) e Acompanhamento e Controle da Produção (curto prazo) (TUBINO, 2009).

2.3. PROGRAMAÇÃO DE OPERAÇÕES

Uma etapa importante dentro da programação é a definição do sequenciamento da produção. Segundo Sipper e Bulfin (1997) o sequenciamento da produção é um processo de organizar, escolher e determinar o tempo de o uso de recursos para realizar todas as atividades necessárias para produzir as saídas nos momentos desejados.

O sequenciamento ocorre quando devemos priorizar qual ordem deve ser a próxima a ser executada em um recurso ou devemos determinar a sequência em que n tarefas deverão ser executadas em um dado recurso (FERNANDES E GODINHO FILHO, 2010).

A capacidade de manipular a maneira como as operações são sequenciadas em uma máquina afeta não apenas os tempos de preparação da máquina, mas também as datas de entregas e os trabalhos em andamento (LIDDELL, 2009).

Existem na literatura diversas regras de sequenciamento. A Tabela 1, adaptada de

Fernandes e Godinho Filho (2010), Gaither e Frazier (2002), Lustosa et. al. (2008) e Tubino (2009), demonstra algumas das regras de sequenciamento reconhecidas.

Existem diversos métodos de PCP, cada um com características e objetivos próprios, adequadas a um tipo de sistema de produção. Exemplos desses tipos de abordagens estão o *Kanban*, CONWIP, OPT, WLC, entre outros.

O sistema *Kanban*, inicialmente pensado por Taichi Ohno na década de 60, foi desenvolvido para operar em um sistema de produção puxada, com base no sistema de atendimento ao cliente e na reposição de estoques, por meio de um dispositivo visual de sinalização (*Kanban*) para ativar a produção, compra ou movimentação dos itens pela fábrica.

O sistema CONWIP (*Constant Work in Process*), de acordo com Spearman *et al.* (1990), é uma forma generalizada do *Kanban*. Entretanto, ao invés de fazer uso de cartões para indicar a produção de uma peça específica, no CONWIP os cartões são designados para um circuito que abrange toda a linha de produção e não correspondem a uma peça específica.

O sistema OPT (*Optimized Production Technology*), Segundo Fernandes e Godinho Filho (2010), trata-se de um sistema informatizado de controle de produção desenvolvido na década de 80 por Eliyahu Goldratt que trabalha como um programador de gargalo, do conceito gerencial conhecido como teoria das restrições (TOC).

Por fim, o WLC (*Workload Control*) ou controle de carga, que é o assunto central deste trabalho, é uma abordagem que se propõe controlar simultaneamente o *lead time* dos produtos, a capacidade produtiva e o estoque em processo (THÜRER E GODINHO FILHO, 2012).

Tabela 1 – Regras de sequenciamento

SIGLA	ESPECIFICAÇÃO	DEFINIÇÃO
FIFO	<i>First In, First Out</i>	O sequenciamento é feito de acordo as ordens que entram no sistema. As ordens que entram primeiro devem ser as primeiras a saírem. Essa regra procura minimizar o tempo de permanência nas máquinas ou na fábrica.
LIFO	<i>Last In, First Out</i>	O sequenciamento é feito priorizando a última peça que entra, devendo ser a primeira a sair.
SPT	<i>Shortest Processing Time</i>	As tarefas são sequenciadas na ordem crescente de seus tempos de processamento. Sua utilização visa reduzir o tamanho de filas e o aumento do fluxo.
LPT	<i>Longest Processing Time</i>	A prioridade é dada pelo maior tempo de processamento total. Contrário à regra SPT. Sua utilização visa à redução de troca de máquinas.
EDD	<i>Earliest Due Date</i>	As tarefas são sequenciadas na ordem crescente de seus prazos. A finalidade é reduzir os atrasos.
LS	<i>Least Slack</i>	A prioridade é dada pela menor folga entre a data de entrega e o tempo total de processamento entre as tarefas que estão à espera. É classificada por prazo de entrega e visa reduzir atrasos.
LWQ	<i>Least Work Next Queue</i>	A prioridade é dada para a tarefa com destino à máquina ou estação de trabalho com menor fila no momento. Essa regra objetiva evitar a parada de um processo.
CR	<i>Critical Ratio</i>	As tarefas são sequenciadas na ordem crescente de suas razões críticas (tempo restante até o prazo dividido sobre o tempo de processamento da tarefa).
DLS	<i>Dynamic Least Slack</i>	A prioridade é dada a menor folga (diferença entre a data prometida e o tempo total restante de processamento).

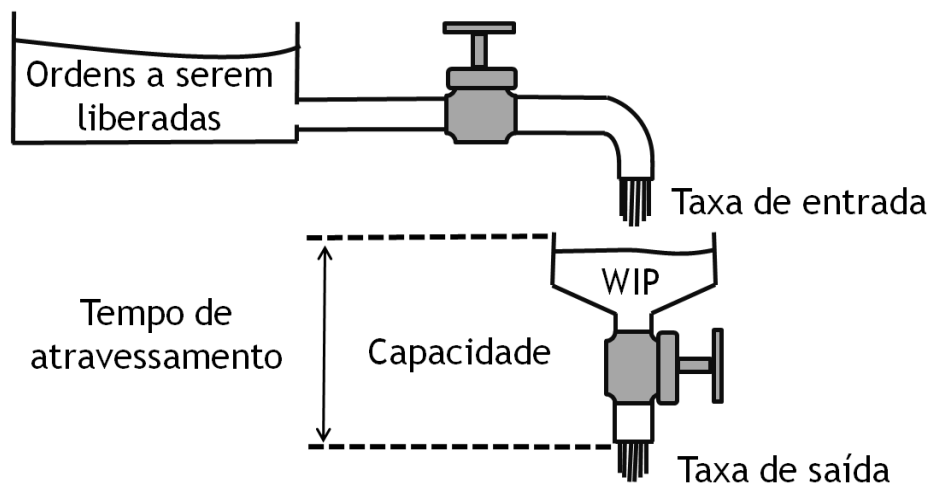
Fonte: Adaptado pelo autor

2.4. CONCEITO E HISTÓRICO SOBRE O CONTROLE DE CARGA (WLC)

O Controle de Carga é um conceito do Planejamento e Controle da Produção (PCP) projetado para ambientes de produção complexos, tais como *jobshop* e produção sob encomenda (*Make-To-Order - MTO*). O método é embasado no conceito dos controles de entrada e saída de Plossl e Wight (1985), cujo controle de entrada se refere a gestão da quantidade de ordens a serem liberadas ao sistema, ou sejam, a quantidade de ordens que entram no sistema, e o controle de saída está relacionado a quantidade de ordens que saem do sistema e é influenciada pela capacidade de processamento do sistema. A Figura 4, chamada de Modelo Funil, representa esse conceito dos controles de entrada e saída, onde é possível observar que o controle dessas taxas está diretamente ligado ao nível do WIP dentro do

sistema. A diferença entre as taxas de entrada e saída, além de determina o nível de WIP, influencia os tempos totais de atravessamento e o *lead time* de produção.

Figura 4 – Modelo Funil



Fonte: Plossl e Wight (1985)

Segundo Hendry, Kigsman e Cheung (1998), o objetivo dos conceitos do controle de carga é melhorar o desempenho da empresa em vários aspectos, incluindo a redução do *lead time*. Este é um objetivo particularmente importante nas empresas de MTO, pois muitas vezes é preciso estabelecer prazos de entrega para seus clientes antes da produção começar.

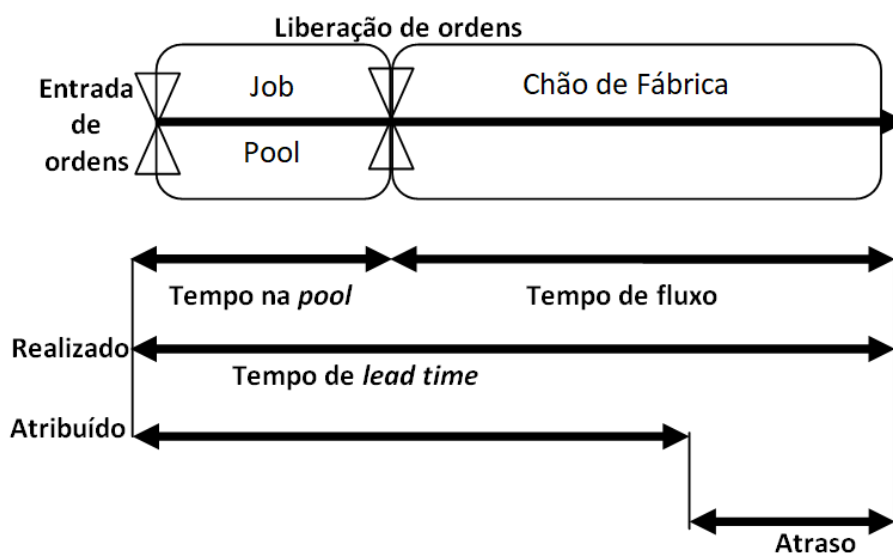
Corroborando com o mencionado acima, Fernandes *et al.* (2016) complementam dizendo que o conceito do WLC é o de tentar superar o dilema de aumentar a velocidade de entrega, mantendo e possivelmente melhorando a confiabilidade de entrega através da liberação de ordens, visando controlar os níveis de carga de trabalho no chão de fábrica e equilibrar a carga de trabalho entre as estações de trabalho, reduzindo e estabilizando os tempos de produção do chão de fábrica. Conseqüentemente, a adequada regulação do nível de carga de trabalho facilita a diminuição simultânea do congestionamento no chão de fábrica e da inanição de recursos, um melhor equilíbrio na distribuição da carga de trabalho e tempos de espera estabilizados (AKILLIOGLU; DIAS-FERREIRA; ONORI, 2016).

O princípio do conceito do WLC é controlar o comprimento das filas de ordens que se formam em frente a cada estação de trabalho por meio da decisão de liberação de ordens, que permite que uma operação entre na fila de sua primeira estação de trabalho (LAND E

GAALMAN, 1996). Uma vez liberada, uma ordem permanece no chão de fábrica até que todas as suas operações tenham sido concluídas.

Os autores supracitados ainda destacam que o WLC não libera as ordens de produção para o chão de fábrica se essas ordens podem causar algum acúmulo de carga que exceda a norma de carga de trabalho estabelecida para as ordens à espera de liberação. Entende-se por norma de carga trabalho a quantidade de trabalho permitida para cada estação de trabalho no chão de fábrica. As ordens não liberadas ficam aguardando a liberação no estoque inicial de ordens (*job pool*). A Figura 5 ilustra o tempo na fila para liberação da ordem no estoque inicial e o intervalo entre a liberação e a conclusão da ordem como o tempo de fluxo no chão de fábrica.

Figura 5 – Componentes do *Lead Time*



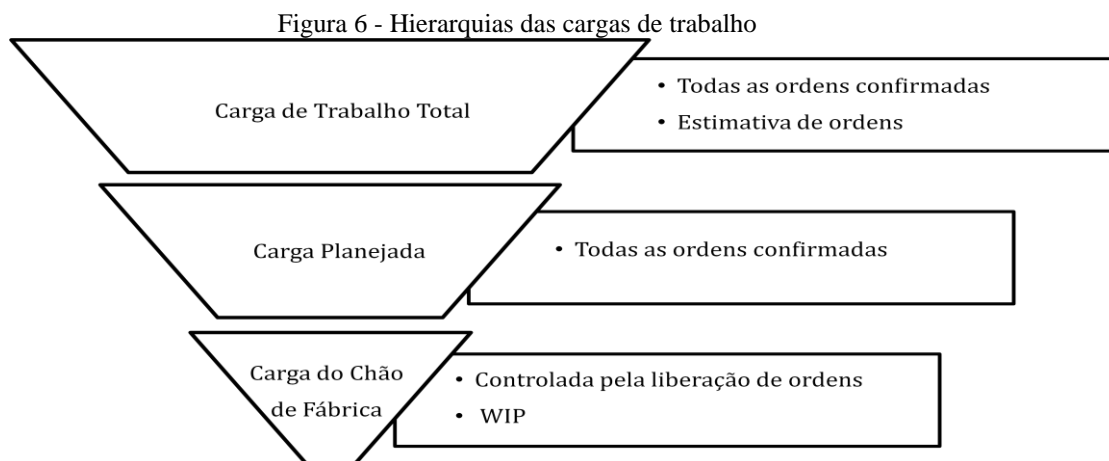
Fonte: Land e Gaalman (1996)

Uma vez liberada, uma ordem permanece no chão fábrica até que todas as suas operações tenham sido concluídas. O progresso das operações no chão de fábrica é controlado por despacho de prioridade em cada estação de trabalho, onde está prioridade é definida pela regra *First In, First Out (FIFO)* (LAND E GAALMAN, 1996).

Essas ordens que foram liberadas compõe a carga do chão de fábrica, entretanto, existem mais duas formas de hierarquizar as cargas de trabalho: carga de trabalho total e carga planejada (THÜRER *et al.*, 2014b).

2.5. HIERARQUIA DO WLC

A carga de trabalho do chão de fábrica, ou *WIP*, é controlada por meio de um mecanismo de liberação de ordens. As ordens são liberadas do *pool* para o chão de fábrica a tempo de cumprir datas de entrega, mantendo o estoque em processo em um nível estável. As cargas de trabalho planejadas e totais são controladas através da gestão de clientes (*customer enquiry management* - CEM). A carga de trabalho planejada consiste em todas as ordens confirmadas e, portanto, inclui tanto a carga de trabalho no chão de fábrica quanto as ordens no estoque inicial (*pre-shop pool*). A carga de trabalho total consiste em todas as ordens confirmadas mais uma carga de trabalho futura estimada - uma porcentagem de ordens não confirmadas é incorporada com base no histórico ou probabilidade das ordens “vencedoras”, ou seja, com grande chance de serem confirmadas, conhecida como taxa de ataque (THÜRER ET AL., 2014b). Essa descrição é detalhada na (Figura 6).



Fonte: Thürer et al. (2014b)

Silva, Steveson e Thürer (2015) resumem mais detalhadamente no Quadro 1 a estrutura hierárquica do conceito do WLC.

Quadro 1 - Hierarquia de cargas de trabalho e o impacto de um novo trabalho sobre os comprimentos de carga de trabalho

Descrição	Metodologia
Tamanho da liberação da carga de trabalho (<i>Release workload Length - RWL</i>)	Consiste em todas as ordens que foram liberadas para o chão de fábrica
Tamanho de carga de trabalho planejada (<i>Planned Workload Length - PWL</i>)	Consiste no RWL juntamente com as ordens que aguardam liberação no <i>pool</i> , ou seja, ordens aceitas para as quais os materiais estão disponíveis
Tamanho total da carga de trabalho (<i>Total workload length - TWL</i>)	Consiste na PWL juntamente com ordens confirmadas à espera de materiais e uma proporção (<i>strike rate</i>) de potenciais ordens aguardando confirmação
Incremento no tamanho da liberação da carga de trabalho (<i>Release workload length (RWL) increase</i>)	A contribuição da carga de trabalho de uma ordem é adicionada ao RWL dos centros de trabalho correspondentes no momento da liberação da ordem
Decréscimo no tamanho da liberação (<i>RWL decrease</i>)	A contribuição da carga de trabalho de uma ordem é subtraída da RWL de uma estação de trabalho quando a operação for concluída e essas informações foram retornadas ao sistema WLC
Incremento no tamanho de carga de trabalho planejada (<i>PWL increase</i>)	Uma ordem é adicionada ao PWL em todas as estações de trabalho afetadas imediatamente após a sua data de liberação mais cedo
Decréscimo no tamanho de carga de trabalho planejada (<i>PWL decrease</i>)	Uma ordem é subtraída da PWL de uma estação de trabalho quando a data de conclusão da operação antecipada foi excedida

Fonte: Adaptado de Silva; Steveson e Thürer (2015)

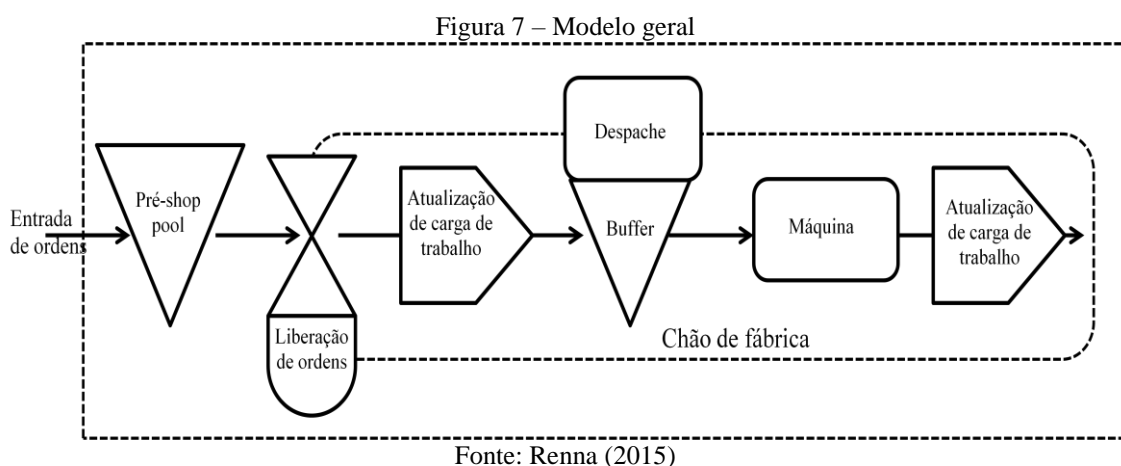
O conceito de controle de carga de trabalho estabelece algumas etapas a serem realizadas para a liberação da ordem ao chão de fábrica a fim de evitar a formação de estoques em processo excessivos e melhorar o tempo de atravessamento.

2.6. ETAPAS DE LIBERAÇÃO

Um dos princípios para a liberação da ordem é estabelecer uma norma que servirá como parâmetro para determinar se uma carga pode ou não ser liberada ao chão de fábrica. O uso de uma norma apropriada pode gerar reduções consideráveis do WIP utilizando níveis de norma com valores baixos, sem deteriorar outros aspectos de desempenho (LAND, 2006). Caso a ordem possua uma carga cujo o valor ultrapasse o nível de cargas estabelecido na norma, essa não é liberada ao chão de fábrica, evitando assim uma formação demasiada de fila em frente a estação de trabalho. Neste caso, essa ordem é encaminhada para o *pool* de ordens não liberadas que ficará aguardando qual o momento da próxima liberação na qual a

sua carga não ultrapasse o nível de cargas estabelecido pela norma. (OOSTERMAN; LAND; GAALMAN, 2000).

Segundo Thüerer *et al.* (2016) existem diversos métodos de liberação de ordem no Controle de Carga. Uma abordagem mais clássica desses métodos, com as principais etapas de liberação de ordens, é apresentada por Renna (2015) que traz em sua pesquisa o fluxo dessas etapas demonstradas na Figura 7, abordando as principais decisões referentes ao despacho da ordem baseado no conceito do WLC.



Como descrito na Figura 7, o conceito do WLC inicia-se com a entrada da ordem que é realizada após as encomendas serem aceitas e geradas no setor responsável pelo planejamento de ordens da empresa. Essas ordens aceitas seguem para o *pré-shop pool* onde irão ser classificadas de acordo com uma regra de classificação. A liberação dessa ordem para o chão de fábrica seguirá uma política de liberação que determinará onde a ordem poderá entrar no chão de fábrica e após entrar no chão de fábrica a carga de trabalho em cada centro de trabalho é atualizada. As ordens na fila dos centros de trabalho são ordenadas de acordo com a política de despacho. Quando uma ordem deixa o centro de trabalho, a carga deste centro de trabalho é atualizada. (RENNA, 2015).

2.7. CLASSIFICAÇÃO DOS MÉTODOS DE LIBERAÇÃO DE ORDEM

Segundo Thüerer *et al.* (2012), os métodos de liberação de ordem no WLC tem por principal objetivo controlar a carga de trabalho no chão de fábrica. Esses métodos são classificados, segundo Fernandes e Carmo-Silva (2011) como periódicos (ex: início de cada turno, dia ou semana) ou contínuos (ex: a qualquer momento durante a operação do sistema).

Entretanto, muitos pesquisadores têm desenvolvido novos métodos de liberação de ordens baseados nas necessidades e características do chão de fábrica. Thüerer *et al.* (2012) apresentam cinco diferentes métodos de liberação que foram desenvolvidos ao longo de pesquisas relacionadas sobre o tema:

- Liberação periódica, na qual os trabalhos são liberados periodicamente até a norma de trabalho (limite superior);
- Liberação contínua através do método *Work Center Workload trigger Earliest Due Date* (WCEDD), no qual se despacha uma ordem ao centro de trabalho com base na sua data de entrega mais cedo, sempre que o centro de trabalho estiver na iminência de ficar com capacidade disponível;
- Liberação contínua baseado no método *Superfluous Load Avoidance Release* (SLAR), que distingue entre trabalhos urgentes e não urgentes (ver mais em (LANDE GAALMAN, 1998));
- LUMS OR (*Lancaster University Management School Order Release*), Hendry e Kingsman (1991) publicaram um estudo onde combinam a liberação de ordem contínua com a periódica (LUMS OR). Segundo Thüerer e Godinho Filho (2012) o LUMS OR combinava esses dois mecanismos na tentativa de evitar que um centro de trabalho fique ocioso (*starvation avoidance mechanism*) devido a uma possível alta carga indireta.
- Liberação periódica e contínua baseado no método *Lancaster University Management School Order Release* (LUMS COR), no qual a liberação periódica é combinada com um mecanismo contínuo que tenta evitar que um centro de trabalho fique ocioso (*starvation avoidance mechanism*) devido a uma possível alta carga indireta (ociosidade prematura) (THÜERER *et al.*, 2012).
- Liberação contínua baseada no método *Constant Work in Process* (ConWIP), que é um método baseado em cartões, similar ao método Kanban, no qual se liberam tarefas se o número de tarefas no chão de fábrica cair abaixo de um nível predeterminado (limite inferior). Ver mais em (SPEARMAN; WOODRUFF; HOPP, 1990).

Quanto ao estudo do desempenho no impacto do controle de carga, apresentados por Thürer, Steveson e Land (2016), quando o controle de entrada (*input control*) e o controle de saída (*output control*) são combinados, os efeitos de desempenho parecem complementar uns aos outros.

Esses métodos de liberação de ordens e seus algoritmos serão detalhados nos tópicos adiante.

2.7.1. Método de liberação periódica

Os métodos clássicos de liberação de ordem periódica apresentados por Land e Gaalman (1998) resulta na decisão de qual ordem poderia ser liberada do *job pool* para o chão de fábrica, considerando a situação da carga no chão de fábrica em combinação com as cargas das ordens e suas relativas urgências (LAND, 2006).

A carga direta é o total de carga dos trabalhos que estão esperando na fila de um centro de trabalho específico ou estão sendo processados nesse centro de trabalho (AKILLIOGLU, DIAS-FERREIRA E ONORI, 2016).

No entanto, para a maioria das estações, a carga direta (*Direct Load* - L_{st}^D) não é afetada diretamente pela liberação de uma ordem, uma vez que tanto as operações que já foram liberadas ao chão de fábrica quanto as que ainda chegarão (operações *upstream* ou à montante) podem ser que sejam processadas primeiro (LAND, 2006). Portanto, para cada estação existirá alguma carga futura que já foi liberada ao chão de fábrica, mas que ainda não chegou ao centro de trabalho para ser processada (carga a montante - L_{st}^U) (LAND, 2006). Originalmente, dois métodos diferentes foram desenvolvidos para lidar com esta carga a montante:

O Método A, desenvolvido em Hanover por Bechte, 1980; Bechte, 1988; Bechte, 1994; Wiendahl, 1995, utiliza uma abordagem chamada de conversão de carga (*load conversion*) para estimar as entradas a partir das cargas das ordens liberadas que ainda chegarão na estação “s” (carga a montante) para a carga direta durante a liberação no período T. Em cada estação, a carga direta mais a entrada de carga estimada (soma indicada como L_{st}^H) é sujeita a uma norma. A carga é convertida depreciando a contribuição de carga de uma ordem “j” a montante (tempo de processamento p_{js}) com um fator de ajuste d_{jst} .

O Método B, desenvolvido por Bertrand e Wortmann (1981) e (Tatsiopoulou, 1983; Kingsman *et al.*, 1989; Hendry, 1989; Hendry and Kingsman, 1991) permite a estimativa de

entradas considerando a carga agregada $L_{st}^A = L_{st}^U + L_{st}^D$. Para cada estação de trabalho, a carga agregada após ser liberada é submetida à norma (LAND, 2006).

O procedimento de liberação periódica primeiro seleciona as ordens no *pre-shop pool* para cada data de liberação planejada (t_j^{*R}) dentro do limite de tempo θ a partir da data atual. Essas tarefas são sequenciadas na ordem de suas datas de liberação planejadas (t_j^{*R}) para determinar suas relativas urgências. De acordo com essa sequência, cada trabalho é considerado para liberação (LAND, 2006).

A seguir, apresenta-se o algoritmo geral do procedimento de liberação de carga a partir do tempo t :

1. Cada ordem j esperando para ser liberada ao chão de fábrica é sequenciada com base na data de liberação planejada (t_j^{*R}), a qual é calculada com uma programação para trás a partir da data de entrega δ_j e do tempo de atravessamento planejado (T_s^{*D} *planned throughput time*), para todas as estações no roteamento de j (o conjunto S_j), ou seja:

$$t_j^{*R} = \delta_j - \sum_{s \in S_j} T_s^{*D}.$$

2. As ordens j com $t_j^{*R} \leq t + \theta$ são incluídos no conjunto J , com θ sendo o tempo limite.
3. A ordem $j \in J$ com a data de liberação mais cedo t_j^{*R} é considerada primeiro.
4. Se os tempos de processamento das ordens p_{js} somados à carga já existente atenderem às normas de carga (Λ_s^D ou Λ_s^A), ou seja,

$$L_{st}^H + d_{jst} \cdot p_{js} \leq \Lambda_s^D \quad \forall s \in S_j \text{ (para o método A) ou}$$

$$L_{st}^A + p_{js} \leq \Lambda_s^A \quad \forall s \in S_j \text{ (para o método B),}$$

então a ordem é selecionada para liberação e esta contribuição de carga é incluída, ou seja,

$$L_{st}^H = L_{st}^H + d_{jst} \cdot p_{js} \quad \forall s \in S_j \text{ (para o método A) ou}$$

$$L_{st}^A = L_{st}^A + p_{js} \quad \forall s \in S_j \text{ (para o método B).}$$

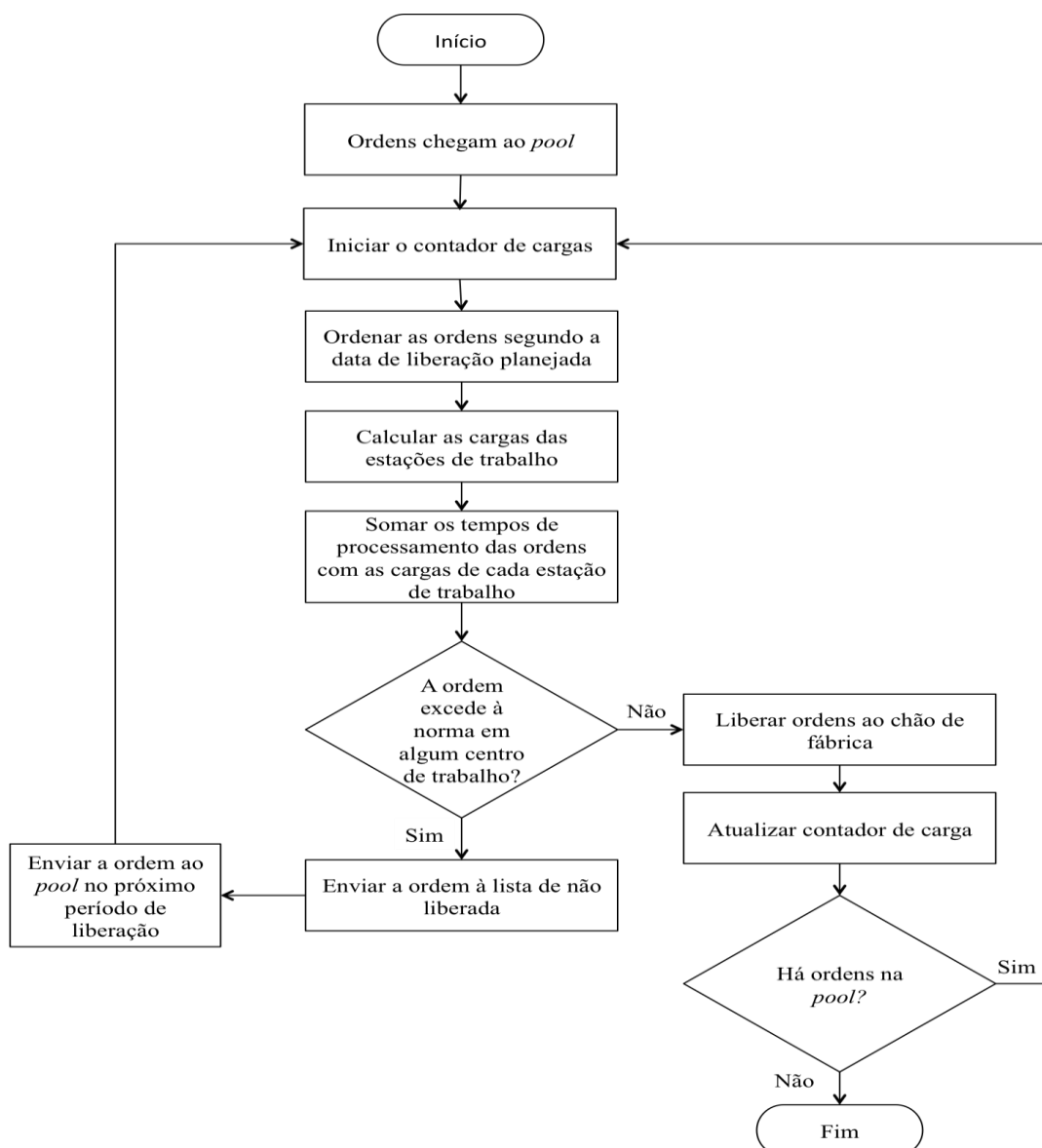
Caso contrário, a ordem deve esperar no estoque até o próximo tempo de liberação.

5. Se o conjunto J possuir alguma ordem que não foi considerada ainda, então retorne ao passo 3 considerando a ordem com a próxima data de liberação mais

cedo, senão, o procedimento de liberação é terminado e ordens selecionadas são liberadas.

O algoritmo descrito é representado no diagrama da Figura 8 **Erro! Fonte de referência não encontrada.** Esse diagrama representa o fluxo clássico de funcionamento do WLC, no qual é possível perceber que caso uma ordem não seja liberada ao chão de fábrica, esta pode entrar em uma espécie de ciclo contínuo (*loop*) e irá permanecer assim até que a sua carga não exceda à norma estipulada. Esse método de liberação clássico do WLC assume que em algum momento a carga de trabalho da ordem não será tão grande a ponto de nunca ser liberada.

Figura 8 - Representação simplificada do algoritmo de controle de carga



Fonte: Elaborado pelo autor

2.7.2. Métodos LUMS OR e LUMS COR

O método *Lancaster University Management School Order Release* ou LUMS OR, foi desenvolvido por (HENDRY E KINGSMAN, 1991) e utiliza um procedimento de liberação periódica, executada em intervalos fixos, para controlar e balancear a carga de trabalho (W_s) no chão de fábrica, mantendo a liberação de carga dentro de uma norma pré-estabelecida para uma dada estação de trabalho (THÜRER; STEVESON; LAND; 2016).

O método de liberação de ordens *Lancaster University Management School Corrected Order Release*, ou LUMS COR, trata-se de um procedimento de liberação de ordens periódico aprimorado do método original LUMS OR.

Thürer; Steveson; Land, (2016) detalham o algoritmo de liberação periódica em quatro partes:

1. Todos as ordens no conjunto de ordens J no *pre-shop pool* são priorizadas de acordo com a regra de sequenciamento do *pool* (ex: data de liberação planejada);
2. A ordem $j \in J$ com alta prioridade é considerada para liberação primeiro;
3. Admita R_j como sendo o conjunto de operações ordenadas do roteiro da ordem j . Seja o tempo de processamento p_{ij} de j na i -ésima operação do roteiro - corrigido para estação i - junto com a carga de trabalho W_s liberada para estação s (correspondente a operação i) e ainda ser completada dentro da norma N_s desta estação, tem-se então:

$$\frac{p_{ij}}{i} + W_s \leq N_s \quad \forall i \in R_j \quad (1)$$

A ordem é selecionada para liberação. O que significa que esta é removida do conjunto J e a sua contribuição de carga é incluída:

$$W_s = W_s + \frac{p_{ij}}{i} \quad \forall i \in R_j \quad (2)$$

Caso contrário, a ordem permanece no *pool* e o tempo de processamento não contribui para a carga da estação.

4. Se o conjunto de ordens J no *pool* contém alguma ordem que ainda não foi considerada para liberação, então retorne ao passo 2 e considere a ordem com alta prioridade. Caso contrário, o procedimento de liberação é completo e as ordens selecionadas são liberadas ao chão de fábrica.

Segundo Thürer *et al.*, (2015) o método LUMS COR incorporou a esse procedimento de liberação da ordens periódica, uma espécie de gatilho contínuo da carga de trabalho que permite que a primeira ordem da sequência no *pre-shop pool* seja liberada ao chão de fábrica sempre que uma estação estiver com carga zero, ou seja, totalmente disponível, e a primeira tarefa do roteiro da ordem seja a tarefa a ser processada na estação disponível,

independentemente se a sua liberação exceder as normas de carga em qualquer estação de trabalho em seu roteiro. Com isso, evita-se a ociosidade em um centro de trabalho.

A contribuição de carga para a estação na LUMS COR é calculada baseado no método de carga agregada corrigida de Windall (1995) e Oosterman *et al.* (2000), que corresponde à divisão do tempo de processamento da operação na estação pela posição da estação no roteiro da ordem. O cálculo considera que 100% da carga é atribuída para a primeira estação de trabalho, 50% para a segunda, 33,33% para a terceira e assim por diante. Entretanto, caso a ordem não seja liberada, a ordem permanece no *pool* e o seu tempo de processamento não contribui para a carga da estação (THÜRER *ET AL.*, 2015).

2.7.3. Método com controle de entrada e saída

Segundo Thürer; Steveson e Land (2016) o controle de carga de trabalho integra dois mecanismos de controle: o controle de entrada (*input control*), que por meio do método de liberação de ordens visa regular o fluxo de entrada de trabalho no sistema, e o controle de saída (*output control*), que utiliza um ajuste de capacidade para regular o fluxo de saída de trabalho do sistema. Entretanto, a maioria dos pesquisadores desenvolveram suas pesquisas com o foco mais voltado para o controle de entrada, negligenciando o controle de saída. Embora tenham surgido mais pesquisas com o foco no controle de saída, demonstrando a importância que o ajuste do fluxo de saída tem para o sistema, segundo os autores supracitados, ainda existe uma lacuna na literatura no que diz respeito ao estudo integrando esses dois mecanismos. Thürer; Steveson e Land (2016) desenvolveram um estudo integrando esses dois mecanismos, onde por meio de uma simulação de um ambiente *jobshop*, demonstraram que tanto o controle de entrada quanto o controle de saída podem e deveriam ser aplicados simultaneamente, complementando um ao outro e melhorando significativamente o desempenho do sistema.

Existem na literatura diversos métodos de liberação de ordens (controle de entrada) para o WLC (Henrich *et al.*, (2003); Thürer *et al.*, 2012; Land e Gaalman (1998); Fernandes *et al.* 2013); Renna, (2015)). O método LUMS COR, recentemente apresentado por THÜRER; STEVENSON e LAND (2016), detalhado no tópico 2.5.2 deste trabalho. Já no que diz respeito ao controle de saída, este método consiste no ajuste da capacidade de processamento de carga do chão de fábrica, visando uma melhoria na velocidade do tempo de

atravessamento das ordens nos períodos de altas cargas. Para isso, as soluções lógicas que poderiam ser aplicadas para esse caso seriam:

1. Aumento de capacidade: uso de horas extras, realocação de operadores entre estações de baixa carga para as de altas cargas, etc;
2. Aliviar as capacidades requeridas nas estações de trabalho que possuem altas cargas: por exemplo, redirecionando ordens, terceirizando as operações das estações de trabalho sobrecarregadas, terceirizando ordens inteiras, etc.

Esses ajustes da capacidade não são permanentes, são ajustes temporários que permanecerão durante o período de alta.

O controle de saída, através do ajuste de capacidade, baseia-se em três parâmetros (α , β e γ) definidos por Land *et al.* (2015) e expostos na Tabela 2. É a partir dos valores desses parâmetros que são tomadas as decisões sobre as variações relacionadas ao ajuste de capacidade. Para que se obtenha uma boa precisão quanto a decisão correta desses ajustes, Land et al (2015) explica que primeiramente é importante quantificar qual o nível de carga de trabalho, em unidades de carga agregada corrigida, que irá ativar os ajustes de capacidade nas simulações. Como apresentado na tabela 3, o nível de carga de trabalho é representado pelo parâmetro β , expressado como uma fração da distribuição de frequência da carga agregada corrigida. Assim que a carga em uma determinada estação de trabalho exceder o valor de β , os tempos de processamento executados serão diminuídos pelo valor do parâmetro α nessa estação de trabalho. Para evitar o retorno da capacidade normal depois de apenas uma única operação ter sido completada, a capacidade implementada será reduzida somente quando a carga corrigida for reduzida a um nível γ abaixo do nível do parâmetro β . Como os valores nas configurações principais de β e γ , definidas por Land *et al.* (2015) na tabela 3 são de 85% e 5%, respectivamente, tem-se que somente após a carga retornar a 80% ($\beta - \gamma$) da capacidade é que o ajuste será interrompido. Entretanto esses parâmetros podem assumir valores diferentes de acordo com a análise de sensibilidade realizada.

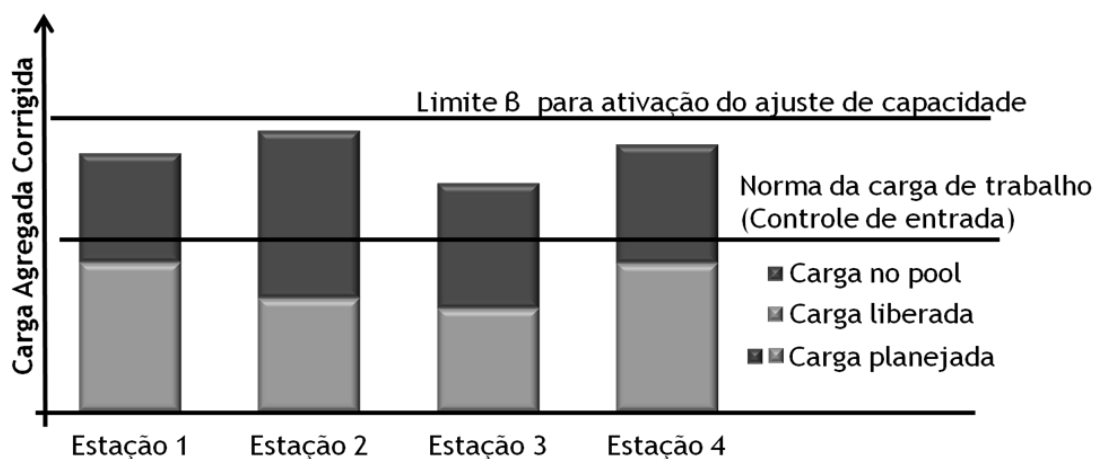
Tabela 2 – Parâmetros do ajuste de capacidade.

Parâmetros	Definição	Principais configurações	Análise de sensibilidade
α	Tamanho do ajuste de capacidade, medido como uma porcentagem de redução no tempo de processamento da operação, ativado na estação de trabalho.	20%	10%, 20%, 30% e 40%
β	Nível da carga de trabalho que ativa o início do ajuste de capacidade	85%	80%, 85%, 90% e 95%
γ	Porcentagem de decréscimo em relação ao nível de acionamento (β) que indica o retorno do nível de capacidade na estação de trabalho as condições normais.	5%	0%, 5%, 10% e 15%

Fonte: Land *et al.* (2015)

Segundo Thürer, Stevenson e Land (2016) a abordagem de liberação de ordem (controle de entrada) com o ajuste de capacidade (controle de saída), quando integrados, parecem se complementar, trazendo um efeito positivo ao sistema. A Figura 9 ilustra a interface gráfica do usuário, combinando essas duas abordagens, normalmente usada para dar suporte a decisão de liberação da ordem.

Figura 9 – Integração entre controles de entrada e saída dentro da liberação de ordens do WLC



Fonte: Thürer; Stevenson; Land. (2016)

A barra clara indica a carga de trabalho liberada W_s para cada estação s que se mantém dentro da norma N_s . A barra escura indica a carga do *pool*, ou seja, a contribuição de carga de todos os trabalhos que aguardam no *pool* para serem liberados, medida em carga corrigida. A soma dessas duas barras, indica a carga planejada, que é usada para orientar ajustes de capacidade em Land *et al.* (2015) e é simplesmente adicionada à carga de trabalho liberada.

2.8. PESQUISAS RECENTES SOBRE O CONTROLE DE CARGA

A partir dos estudos de Plossl Wight (1973) sobre os conceitos de controle de entrada e saída de ordens de trabalho no chão de fábrica, que abriram as pesquisas sobre o controle de carga (WLC), o tema segue sendo muito pesquisado, inclusive nos últimos 5 anos com pesquisas relacionando com outras abordagens e aplicações.

Thürer *et al.* (2012) utilizaram o conceito de liberação de ordens da abordagem do controle de carga, a fim de encontrar uma solução *Lean* para empresas de ambientes *make-to-order* (MTO). Nesse estudo, os autores supracitados buscaram demonstrar que o WLC pode controlar o *lead time*, a capacidade de produção e os níveis de estoques em processo (WIP). Para tal, fez-se uso do método de liberação de ordens LUMS COR (*Lancaster University Management School Corrected Order Release*), que combinava o método original baseado em uma liberação periódica LUMS OR (HENDRY E KINGSMAN, 1991) com uma liberação disparada por um gatilho, ou baseada em eventos, que liberava (disparava) uma ordem ao chão de fábrica sempre que um centro de trabalho estiver com capacidade disponível (ocioso).

Já os estudos de Soepenber, Land e Gaalman (2012) visaram demonstrar como adaptar o conceito do controle de carga para ambientes *jobshop* mais complexos, com grande número de produtos, diversas ordens sendo executadas em paralelo, longas rotas de produção, etc. O intuito dessa pesquisa com a adaptação do conceito do WLC é melhorar a sinergia entre a liberação controlada, para evitar atrasos irresolúveis nas fases iniciais, e o despacho prioritário em estágios posteriores das rotas de longo prazo. Embora o conceito de controle de carga tenha sido desenvolvido para ambientes *jobshop*, o principal foco de liberação do WLC não é suficiente para um ambiente *jobshop* complexo. Com isso, os autores propuseram duas adaptações nos conceitos de liberação de ordem do WLC, inspiradas em estudos empíricos, que utilizam um tempo de *buffer* mais planejado e realista para etapas iniciais do roteiro, permitindo a priorização das ordens próximas às datas de entrega.

Pürgstaller e Missbauer (2012) por meio do estudo de simulação de um caso em uma indústria de fabricação de mídias ópticas, fazem uma comparação entre modelos de otimização de liberação de ordens de produção (controle de entrada e saída com *lead time* fixos) com o modelo tradicional do WLC com controle de carga agregada com a liberação de. Os resultados dessa simulação apontam para a maioria dos casos simulados um melhor desempenho do modelo otimizado frente ao modelo tradicional de liberação de ordens,

mesmo quando a previsibilidade da demanda for baixa, exceto em casos onde a maioria das demandas é constante e em cenários em que há elevada variação do *mix* de produtos.

No Brasil, pesquisas sobre controle de carga obtiveram destaque a partir de Thüerer e Godinho Filho (2012), que buscaram apresentar os conceitos do WLC para a comunidade acadêmica e empresarial com o foco em pequenas e médias empresas (PME). Tal pesquisa vislumbrou a redução do *lead time* e pontualidade nas entregas para as PME que fabricam por encomenda (MTO), apresentando uma revisão dos pontos mais importantes tratados na literatura e realizando uma simulação com o objetivo de avaliar o desempenho do WLC em um ambiente MTO, o que ao final se comprovou como uma opção de grande potencial para o PCP das PME.

Battaglia *et al.* (2016) contribuíram também para as pesquisas brasileiras em WLC, aplicando os seus conceitos para analisar o tempo de atravessamento e o estoque em processo em um sistema flexível de manufatura, fabricante de artefatos plásticos, através de um método quantitativo.

Embora exista um consenso na literatura quanto ao bom desempenho da utilização dos conceitos do controle de carga no PCP das empresas que trabalham em um ambiente MTO, as publicações referentes a casos de sucessos de implantações ainda são bem escassos. Com isso, Hendry, Huang e Stevenson (2013) buscaram descrever um caso bem sucedido de implantação do controle de carga, onde por meio de uma pesquisa-ação longitudinal em uma empresa “Y”, entre os anos de 2007 e 2009, no qual o processo de implementação do WLC foi monitorado a fim de verificar se o sistema era sustentável. Em sua publicação, os autores elaboram um conjunto com 17 questões e respostas a partir da literatura e utilizada na implantação do WLC. Como resultados encontrados, pode-se listar a redução do *lead time*, melhorias significativas em ordens atrasadas, redução dos custos, entre outras. Com isso, os autores conseguiram demonstrar empiricamente as melhorias do desempenho resultantes do controle de carga ao lado de uma discussão detalhada sobre o processo de implantação.

Ainda no que se refere ao estudo das teorias do Controle de Carga associado a sua aplicação prática, destacam-se os estudos de Fernandes e Carmo-Silva (2013) e Silva, Stevenson e Thüerer (2015). A primeira publicação buscou preencher a lacuna existente na literatura entre a teoria e a prática no Controle de Carga, obtendo resultados que levam a importantes diretrizes para o gerenciamento dos sistemas de produção. Já o segundo estudo

trata-se de um caso de sucesso de implantação do WLC em uma empresa de bens de capital produtora de trilhos de alumínio.

Fica evidente na literatura que os conceitos do controle de carga vêm sendo combinados com vários outros conceitos a fim de se obter melhores desempenhos no PCP da empresa. Isso é o que corroboram as pesquisas de Thüerer *et al.* (2014a), que utilizam os conceitos do WLC para determinar quatro novas regras para subcontratação baseadas nas capacidades disponíveis do chão de fábrica, e Thüerer *et al.*, (2014b), que através do conceito de liberação de ordens da abordagem do WLC demonstraram que uma abordagem integrada do controle de carga representa um importante passo para alcançar o *lean* em empresas MTO, contribuindo para o nivelamento da demanda e da produção ao longo do tempo, além de reduzir a porcentagem dos trabalhos atrasados e reduzir os níveis de estoque.

Atualmente, estudos que comparam o desempenho do WLC com outros métodos estão sendo desenvolvidos e publicados. Um caso desses estudos é apresentado por Thüerer *et al.* (2017), que comparou o desempenho da aplicação dos conceitos do WLC com os mecanismos de liberação de ordens contidos nas Teorias das Restrições (TOC) – ou seja, Tambor, pulmão e corda, ou *Drum-Buffer-Rope* (DBR) – apresentados por Goldratt e Cox (1984) e Goldratt (1990), com o intuito de apoiar os gerentes em sua decisão sobre qual abordagem se aplicar em ambiente MTO e *jobshop* com o fluxo de alta variedade e a presença de gargalos.

Por fim, estudos sobre a abordagem do Controle de Carga mais atuais, fazem referência a combinação dos conceitos de liberação de ordens com métodos de controle de saída (através de ajuste de capacidade) apresentados por Land *et al.* (2015) e Thüerer, Stevenson e Land (2106). Estudos que abordam essa combinação citada, fazem esse ajuste de capacidade apenas reduzindo os tempos de processamento das ordens em uma porcentagem predeterminada. No caso de Land *et al.*(2015), os autores estipularam uma redução em porcentagem nos tempos de processamento de $\alpha = 20\%$ nas estações de trabalho.

3 MÉTODO DE PESQUISA

Tendo em vista que este trabalho visa detalhar a construção da estrutura de conversão da abordagem do controle de carga em um *software* comercial de simulação de eventos discretos, esta pesquisa utilizou uma abordagem quantitativa. Segundo Bryman (1989), um dos principais benefícios da abordagem quantitativa é justamente a possibilidade da pesquisa poder ser replicada, ou seja, um pesquisador pode repetir uma pesquisa de outro e encontrar os seus resultados.

Dentre as classificações tipológicas das pesquisas quantitativas, essa dissertação é classificada como uma pesquisa Axiomática Descritiva (PAD). Axiomática pois a construção da estrutura de conversão neste trabalho é dirigida a um modelo de problema já idealizado, ou seja, já existente na literatura (MORABITO E PUREZA, 2012). Descritiva pois o objetivo é descrever o problema modelado (estrutura de conversão do Controle de Carga) permitindo um melhor entendimento de suas características e relacionamentos funcionais do ambiente estudado. O método de pesquisa aplicado nesta dissertação é a modelagem e simulação, uma vez que a conversão proposta pelo trabalho buscou descrever a construção do modelo conceitual ao modelo de simulação com *software* comercial.

O desenvolvimento deste estudo baseou-se nas seguintes etapas:

1. Formular o problema: descrever como construir uma estrutura de conversão da abordagem do controle de carga de trabalho (WLC) para um *software* de simulação de eventos discretos (em detalhe no capítulo 4);
2. Estabelecer os objetivos e o plano de projeto da simulação: descrever todos os elementos que compõem o sistema modelado.
3. Conceitualizar o modelo: o modelo a representar virtualmente é um chão de fábrica que possui um fluxo de ordens com ambiente de produção *jobshop* e que utiliza o mecanismo de liberação de ordens por carga de trabalho (em detalhe na seção 3.1);
4. Coletar dados: promover a coleta dos dados utilizados para um ambiente de produção *jobshop* puro, considerando o modelo da literatura utilizado por Land (2006) (em detalhe na seção 3.1);
5. Verificar: confirmar se o ambiente modelado possui o mesmo comportamento na execução que o modelo da literatura apresentado por Land (2006) e se esse

ambiente representa a realidade da liberação das ordens do WLC (em detalhe no capítulo 4);.

6. Desenhar o experimento: considerar os diferentes cenários que o ambiente modelado precisa ser avaliado na redução de carga de trabalho, a fim de obter desempenho nos novos cenários (em detalhe na seção 3.4);
7. Produzir as simulações e analisar: executar as simulações e comparar os resultados dos diferentes cenários utilizados no experimento (em detalhe no capítulo 5);
8. Documentar: descrever os resultados e a implementação do ambiente modelado.

3.1. COMPONENTES DO SISTEMA A SER SIMULADO NA PESQUISA

Este trabalho apresenta a modelagem de um sistema de produção (chão de fábrica) com uma configuração *jobshop*, composto por seis máquinas, que permite representar os fluxos de produção com diversos roteiros entre essas máquinas. Entretanto, cada máquina realiza somente uma operação por ordem, implicando que uma ordem possui um limite máximo de seis operações. Segundo Oosterman, Land e Gaalman (2000), uma representação válida para um chão de fábrica *jobshop* são roteiros aleatórios possíveis entre seis máquinas, pois um número maior de máquinas pode gerar uma grande quantidade de roteiros dificultando a avaliação dos efeitos das decisões sobre a liberação de ordens. O tempo de processamento de uma operação da ordem em cada máquina desse sistema é aleatório, representando assim uma alta variedade das operações realizadas numa fábrica real.

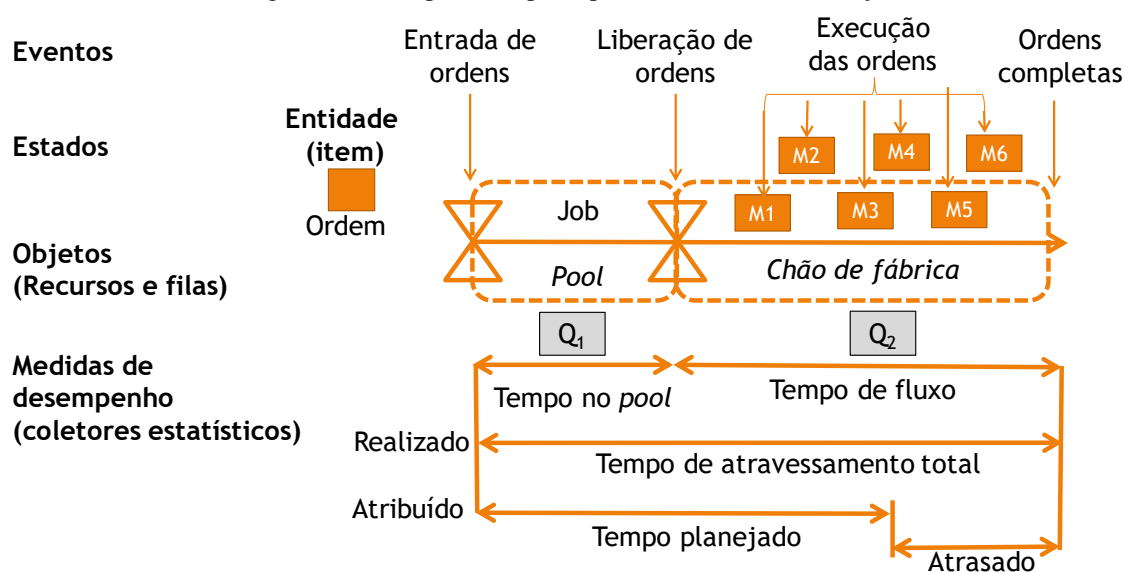
No ambiente simulado, a liberação das ordens é realizada por meio de uma verificação de carga de trabalho a fim de evitar cargas excessivas que incrementem o tempo de espera dentro do chão de fábrica, como aponta a abordagem *Workload Control* (WLC). As ordens são retidas no *job pool* para não incrementarem as filas individuais na frente de cada uma das seis máquinas. O sistema modelado representa um tipo de produção *Make-to-Order*.

Os principais componentes do sistema modelado utilizados e as suas relações, aparecem representados na Figura 10, que é baseado nos conceitos da Figura 7 apresentada no capítulo 2. Figura 10 permite verificar onde serão coletados os dados do sistema modelado a partir dos seus componentes: Eventos; Estados; Objetos (recursos e filas); Entidade (item); Medidas de desempenho. Os Eventos são distribuídos em quatro fases: entrada de ordens;

liberação de ordens; execução das ordens; ordens completas. Os Estados representam as quantidades de ordens Q_1 e Q_2 em cada evento. Os objetos são os recursos/máquinas (M1 à M6) e as filas. A entidade (item) representa a ordem que será processada pelo sistema. As Medidas de desempenho representam os tempos e os dados que serão coletados no sistema.

Destaca-se que a estratégia a desenvolver pelo controle de carga é manter uma quantidade de ordens no *job pool* (Q_1) para evitar que a quantidade das ordens no chão de fábrica (Q_2) se torne muito elevada e crie prazos difíceis de serem cumpridos para os clientes.

Figura 10 – Componentes principais do modelo de simulação



Fonte: Adaptado de Land e Gaalman (1996)

O Quadro 2 apresenta as características e os dados utilizados para estudar os efeitos do controle de carga no sistema modelado neste trabalho, que são tomados do modelo utilizado por Land (2006). Segundo analisado na base de dados da *Web of Science* (2018), no que diz respeito as características do modelo apresentados por Land (2006) e dispostas no Quadro 2, a escolha desse modelo para a pesquisa se justifica por estar presente em mais de 50 citações sobre o tema. Já em respeito ao autor, esse foi escolhido pois está entre os cinco autores que mais publicam trabalhos sobre a abordagem do controle de carga, segundo análise apresentada na base de dados *Scopus* (2018).

Quadro 2 – Características do sistema modelado utilizado como referência

Chão de fábrica	Seis estações, cada uma com capacidade única
Tamanho do roteiro de produção	Distribuição discreta uniforme de [1,6]
Estações do roteiro	Roteiro totalmente aleatório, mas sem permitir um re-ingresso da ordem na mesma máquina
Tempo de intervalo entre chegadas	Distribuição exponencial (média: 0.648 unidades de tempo)
Data de entrega	Distribuição uniforme com valores entre 35 a 60 unidades de tempo
Tempo de processamento da operação	Distribuição 2-Erlang (média: 1 unidades de tempo)
Regra de prioridade de despacho	<i>First come, first served</i> - FCFS (primeiro a chegar é o primeiro a ser servido/liberado)

Fonte: Land (2006)

O sistema modelado considera seis estações de trabalho que são responsáveis por processar as ordens que chegam a essas estações baseadas em roteiros de produção que simulam um ambiente de produção *jobshop*. O tamanho do roteiro de produção (quantidade de operações dentro de cada roteiro da ordem) é aleatório, podendo variar entre uma operação (passando em apenas uma estação de trabalho) ou até seis operações (um roteiro que possui todas as estações do sistema), garantido a maior variedade de fluxo. Dentro do roteiro, não se considera o reingresso na estação de trabalho.

O tempo de intervalos de chegadas é determinado por meio de uma distribuição exponencial com média 0.648 segundos. A data de entrega é a data que a ordem deverá ser entregue e seus valores são determinados por meio de uma distribuição uniforme variando de 35 a 60 segundos. O tempo de processamento da operação é determinado por uma distribuição 2-Erlang com média igual a 1. Segundo Oosterman, Land e Gaalman (2000) a distribuição de Erlang representa melhor a variação real possível das operações que cada máquina realiza em cada ordem, do que uma distribuição exponencial. A regra de prioridade de despacho é aplicada para liberar as ordens que estão aguardando nas filas que antecedem às máquinas, na sequência na qual a primeira ordem que chega à fila, é a primeira a ser liberada à máquina para ser processada.

3.2. CONSTRUÇÃO DO SISTEMA MODELADO NA LINGUAGEM DO SOFTWARE

Para construir sistema modelado funcionando com a abordagem do controle de carga, utilizou-se os conceitos de simulação de eventos discretos. O Quadro 3 apresenta como o modelo é representado e demonstra as definições correspondentes ao *software* FlexSim.

Quadro 3 – Descrição das componentes do sistema modelado

Eventos (Events)	Estados (States variables)	Medidas de desempenho (performance variables)	Objetos fixos (Fixed Resources):	Atividades e esperas (Activities and delays)	Algoritmos associados ao evento (Event Triggering)
1. Entrada da ordem	+Q1: quantidade de ordens no <i>pool</i>	Tempo no <i>pool</i>	Gerador de entidades (<i>Source</i>), <i>job pool</i> (fila - <i>queue</i>)		Definir roteiro da ordem de trabalho. Estabelecer a entrega da ordem
2. Liberação de ordens	-Q1: quantidade de ordens no <i>pool</i> +Q2: quantidade de ordens no chão de fábrica	Tempo de fluxo	Sistema de reingresso na <i>pool</i> (<i>Queue</i> + <i>processor</i>) Contador de tempo TIC-TAC (<i>Source</i> TIC + <i>Sink</i> TAC)	Tempo de espera dentro do <i>pool</i> .	Ordenar as ordens de trabalho. Avaliar a carga de trabalho respeito à norma. Liberar a ordem no chão de fábrica
3. Execução da ordem e atualização da carga	Variação da distribuição Q2 pelas ordens em cada máquina.	Utilização por máquina	Chão de fábrica de 6 máquinas (<i>processor</i>) e 6 filas de cada máquina (<i>queue</i>)	Tempo de processamento de cada máquina Espera em cada fila	Atualizar o valor de carga de trabalho liberada em cada máquina e descontar a carga já terminada por cada máquina.
4. Finalização da ordem	-Q2: quantidade de ordens no chão de fábrica	Tempo de atravessamento total Atraso no sistema	Finalizador da ordem (<i>Sink</i>)		Armazenar os dados de <i>lead time</i> das ordens de produção

Fonte: Elaborado pelo autor

Nesse quadro é possível notar que a cada evento a quantidade das ordens vão se alterando e essas quantidades, bem como os tempos gastos para percorrer de um evento para o

outro, são coletados e utilizados para calcular as medidas de desempenho do sistema. O detalhe da representação dos códigos implementado aparecem no capítulo 4.

3.3. DESENHO DO EXPERIMENTO

Nos testes, foram considerados diferentes cenários variando-se os valores da norma de carga. Os níveis da norma considerados são: a) infinita (um valor de $10e6$ como representação); b) 8; c) 16; d) 24; e) 32. O termo norma infinita é utilizado por Land (2006) para representar um valor de norma muito grande. Neste estudo, utilizou-se para norma infinita o valor igual a $10e6$, o que garante que não haja restrição para liberação da ordem baseado na sua carga. Os demais valores foram escolhidos através de testes e simulações, incrementando a norma de forma crescente para testar o efeito do controle de carga nos tempos de atravessamento e tempo de fluxo.

Os resultados encontrados dos tempos de atravessamento total (*Total Throughput Time*) e nos tempos de fluxo (tempo de atravessamento do chão de fábrica) no cenário com a norma infinita serviram para comparar o modelo simulado neste trabalho com o modelo apresentado por Land (2006). Os demais resultados dos cenários adjacentes serviram para avaliar os efeitos da norma dentro das medidas de desempenho.

4 MODELAGEM E CONSTRUÇÃO DA ESTRUTURA DE CONVERSÃO

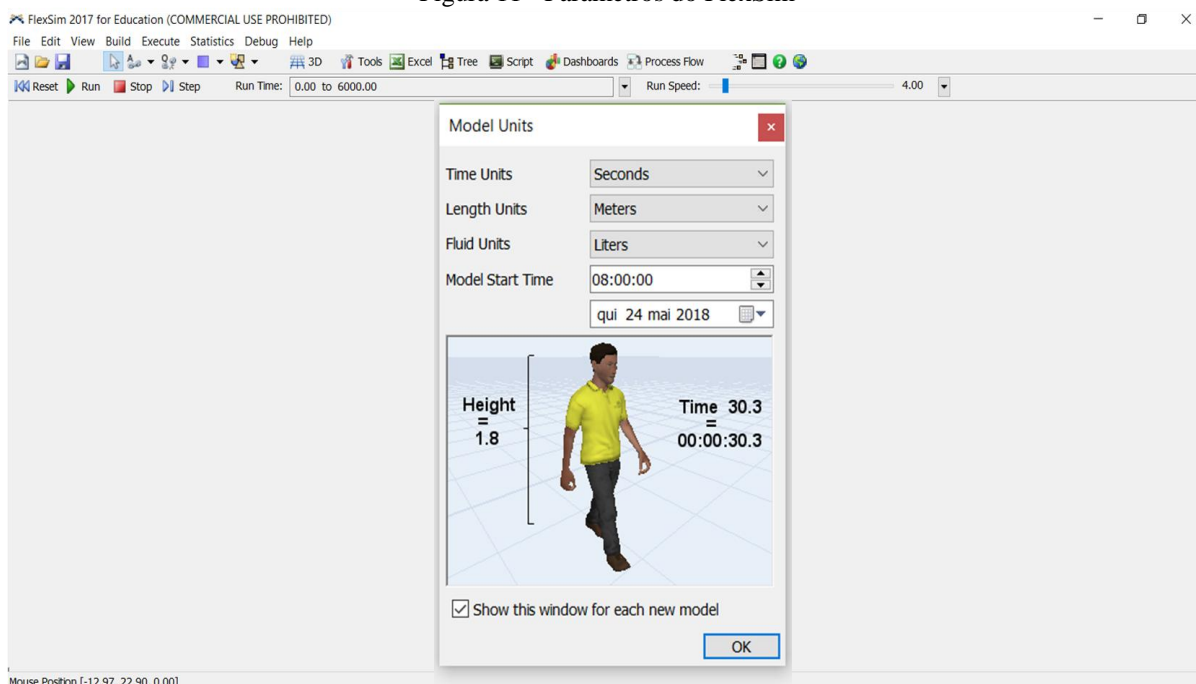
4.1. O SOFTWARE FLEXSIM

O *software* FlexSim trata-se de uma ferramenta de modelagem e simulação, no qual permite a reprodução de um ambiente produtivo para uma plataforma de simulação e emulação. Possui um coletor de dados em diferentes pontos da modelagem, em paralelo com a simulação, o que permite gerar gráficos e realizar análises estatísticas.

A escolha pela utilização deste *software* foi definida pela sua facilidade de acesso e a maneira clara e direta de modelar e simular um ambiente de produção. Utilizou-se a versão 17, na qual permite construir modelos e simulá-los simplesmente arrastando e soltando os objetos, com controles intuitivos e uma visualização em 3D (Figura 12).

A Figura 11 apresenta a tela inicial do FlexSim, onde são definidas as unidades do modelo a ser simulado, como unidades de tempo, tamanho, volume, horário e data do início da simulação.

Figura 11 - Parâmetros do FlexSim

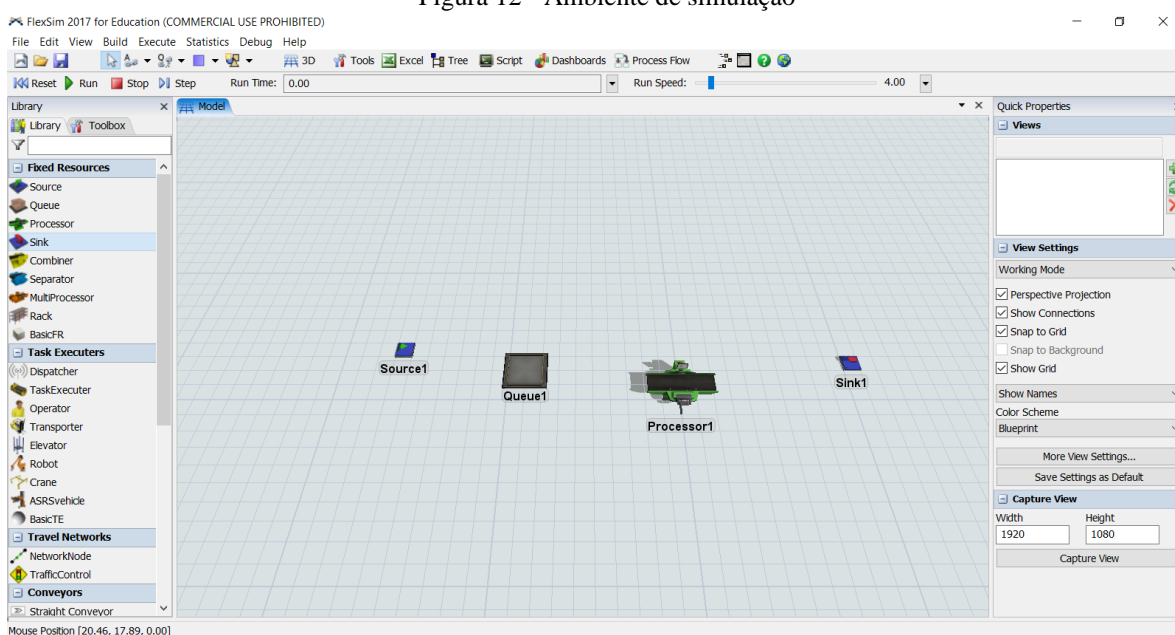


Fonte: Elaborado pelo autor

A Figura 12 apresenta o ambiente de simulação em 3D do *software* onde é possível clicar e arrastar os objetos desejados para a superfície de simulação.

Para a criação do modelo proposto por este trabalho foram utilizados quatro objetos do FlexSim: *source* (entrada), que é utilizado para criar os *flowitens* (itens) que irão percorrer o modelo; *queue* (fila), que é utilizado para armazenar os itens quando os objetos à frente não puderem receber esses itens; *processor* (máquina), que é utilizado para simular o processamento dos itens do modelo; *sink* (saída), que é utilizado para determinar o final da simulação, destruindo os itens criados no *source* ao final do roteiro.

Figura 12 - Ambiente de simulação



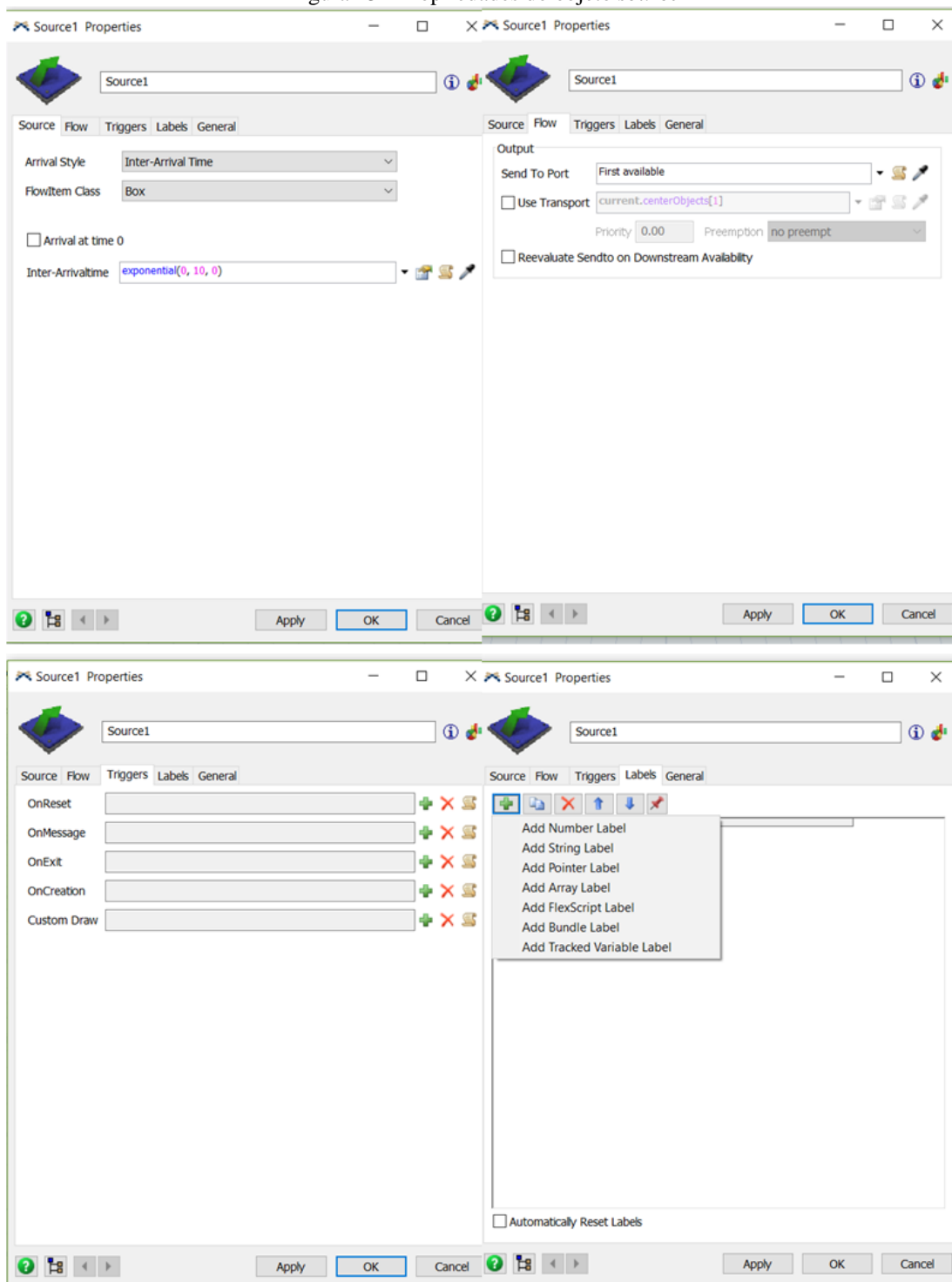
Fonte: Elaborado pelo autor

Com um duplo clique com o botão esquerdo do *mouse* sobre um objeto, é aberta uma janela de propriedades onde são configuradas as características e os parâmetros de ações que cada objeto deve possuir e realizar.

A janela de propriedades é composta por abas que permitem determinadas configurações. Além da aba que contém o próprio nome do objeto, as abas que estão presentes em todas as telas de configuração dos objetos utilizados e que serão configuradas nesse trabalho são: *flow*, *triggers* e *labels*.

A Figura 13 apresenta a tela de propriedades do objeto *source*; na primeira aba, *source*, é possível definir os tipos, os intervalos de chegada e as classes dos itens. A regra de envio dos itens para a próxima porta de entrada é configurada na aba *flow*, podendo configurar o item para ser direcionado para a primeira entrada disponível ou para outras portas com base em alguma regra personalizada. A aba *triggers* é responsável por gerar ações

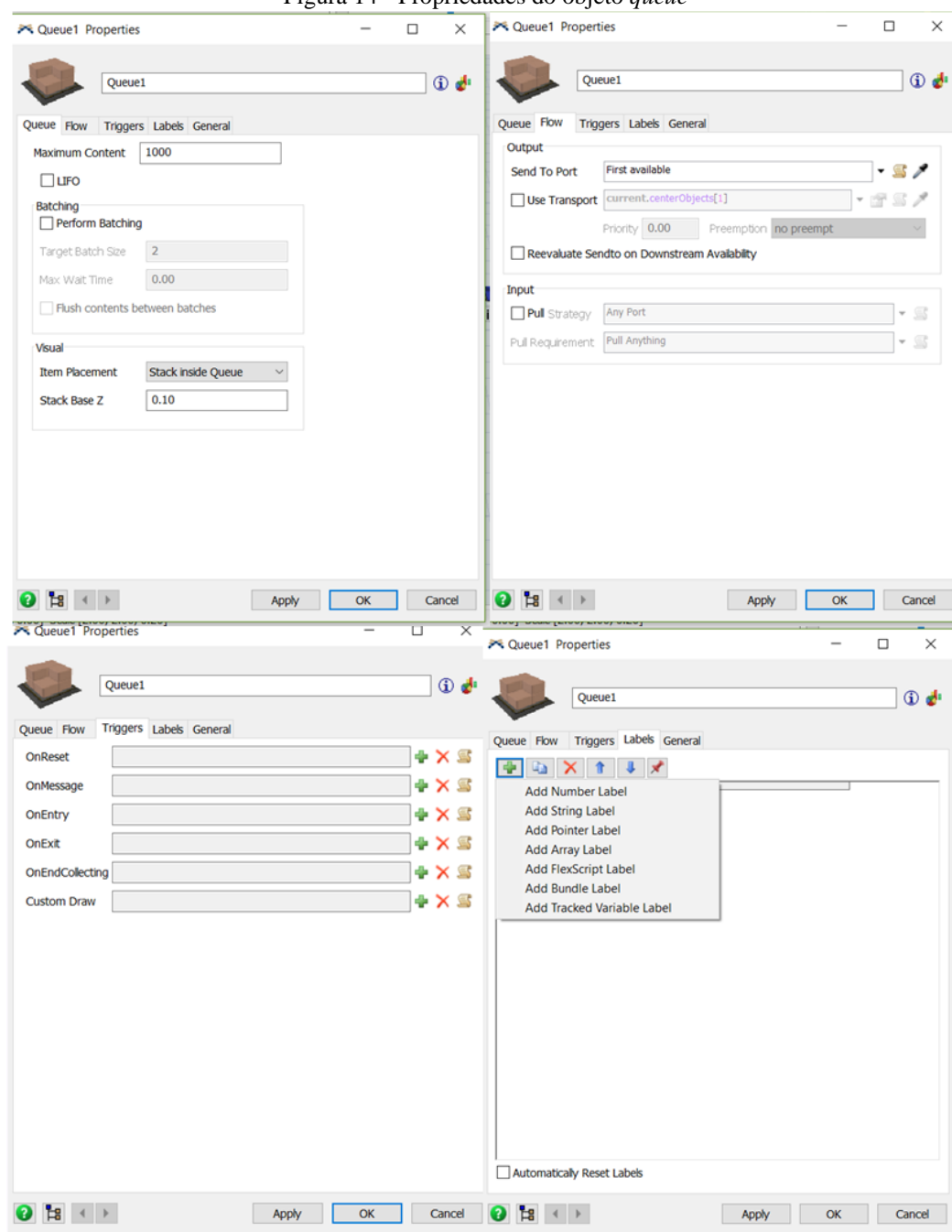
no modelo ou no item. Essas ações podem ser realizadas tanto no momento em que o modelo é reinicializado ou quando o item entra ou sai do objeto. A criação e/ou edição das etiquetas (*labels*) que são utilizadas para armazenar informações dos itens são realizadas na aba *label*.

Figura 13 - Propriedades do objeto *source*

Fonte: Elaborado pelo autor

A Figura 14 apresenta as tela de propriedades do objeto *queue*. Nessa tela é possível clicar nas abas: *queue*, *flow*, *triggers*, *labels*, *general*, e configurar este objeto baseado no número máximo de itens que podem ficar nesse objeto, a regra de despacho do item, as ações de personalização do modelo, criação e edição das etiquetas, entre outros.

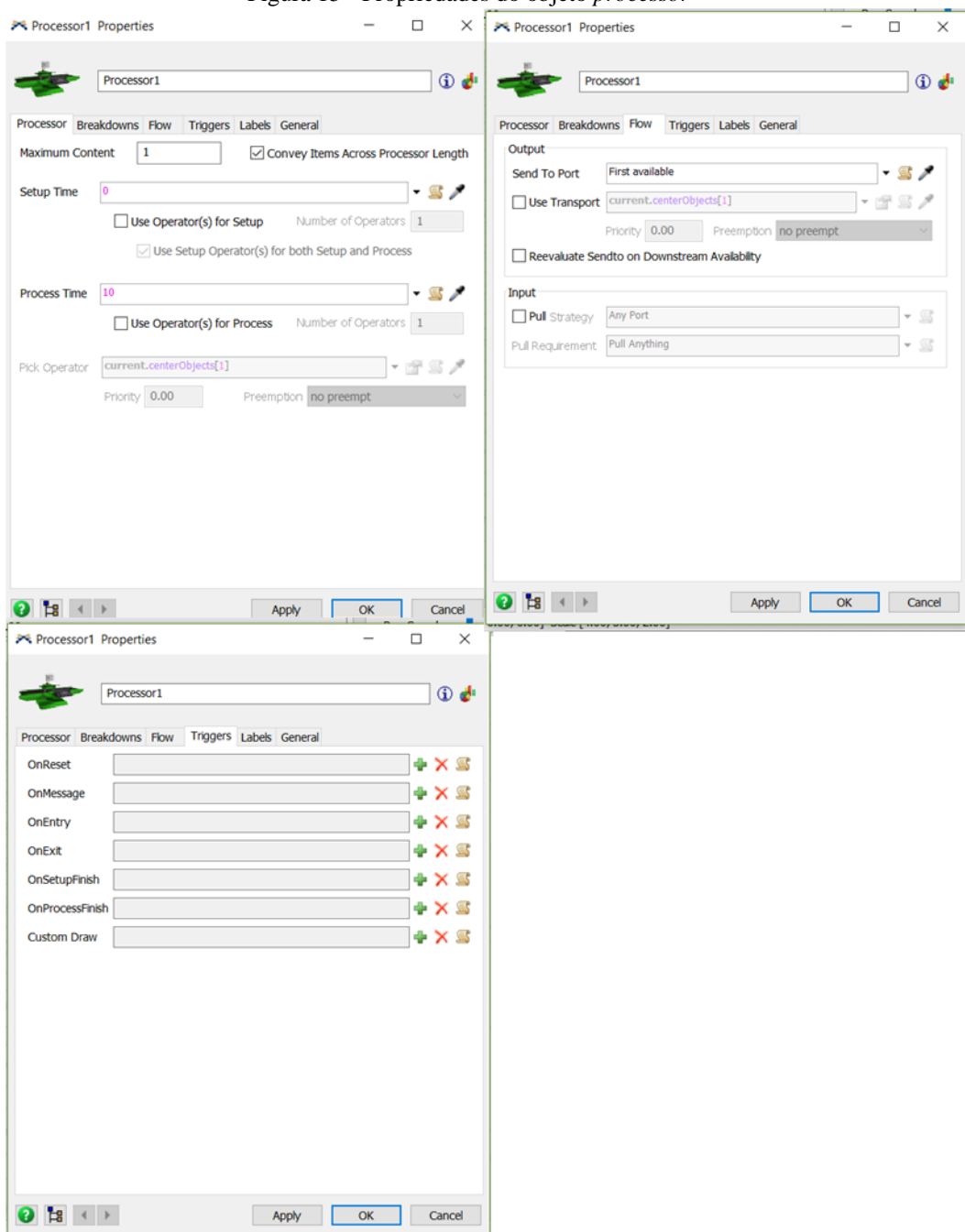
Figura 14 - Propriedades do objeto *queue*



Fonte: Elaborado pelo autor

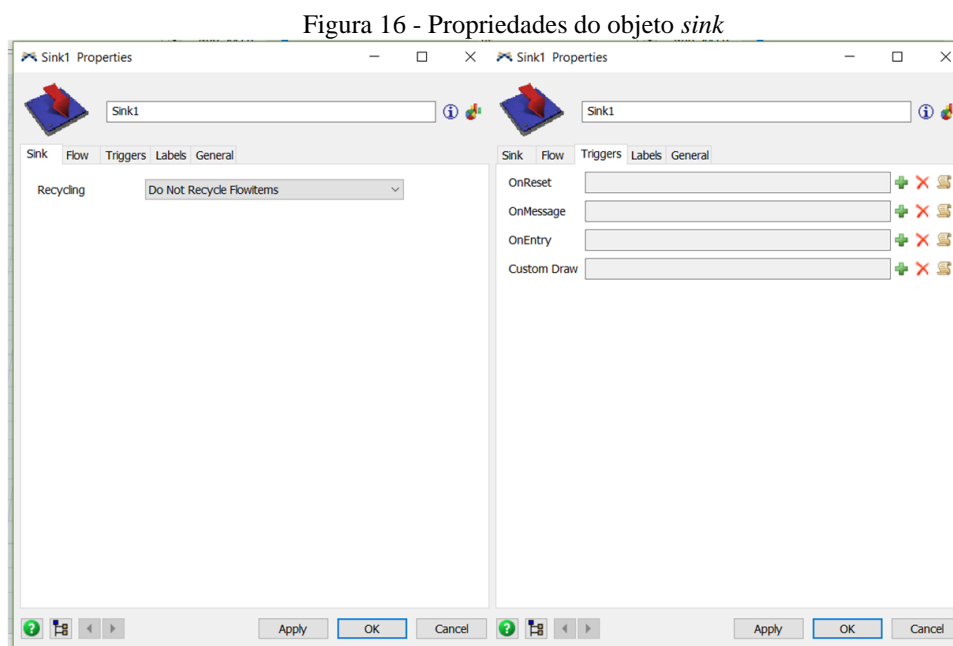
A Figura 15 apresenta a tela de propriedades do objeto *processor*, pelo qual por meio das abas é possível configurar a quantidade máxima de itens que podem ser processados por vez, o tempo de *setup* de máquina, o tempo de processamento por item, a regra de despacho do item, ações de personalização do modelo, criação de etiquetas, entre outros.

Figura 15 - Propriedades do objeto *processor*



Fonte: Elaborado pelo autor

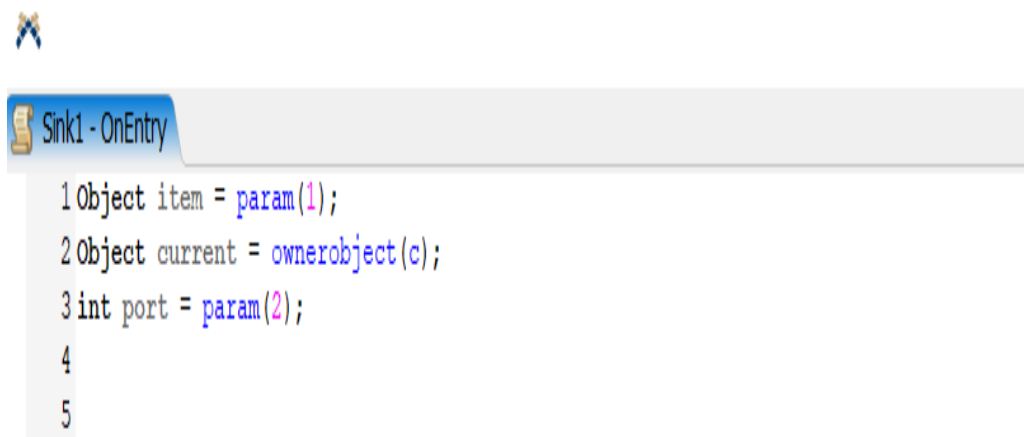
A Figura 16 apresenta a tela de propriedades do objeto *sink*, pelo qual, por meio das abas, é possível configurar as ações que deverão ocorrer sobre os itens, coletar dados e criar ações relacionadas à saída e destruição desses itens quando estes se encontram ao final do processo simulado.



Fonte: Elaborado pelo autor

Em todas as telas de propriedades dos objetos, sempre que aparecer a figura de um pergaminho/documento, é possível editar um código de programação diretamente no modelo, utilizando a linguagem de programação C (Figura 17).

Figura 17 - Ambiente de programação dentro do objeto



Fonte: Elaborado pelo autor

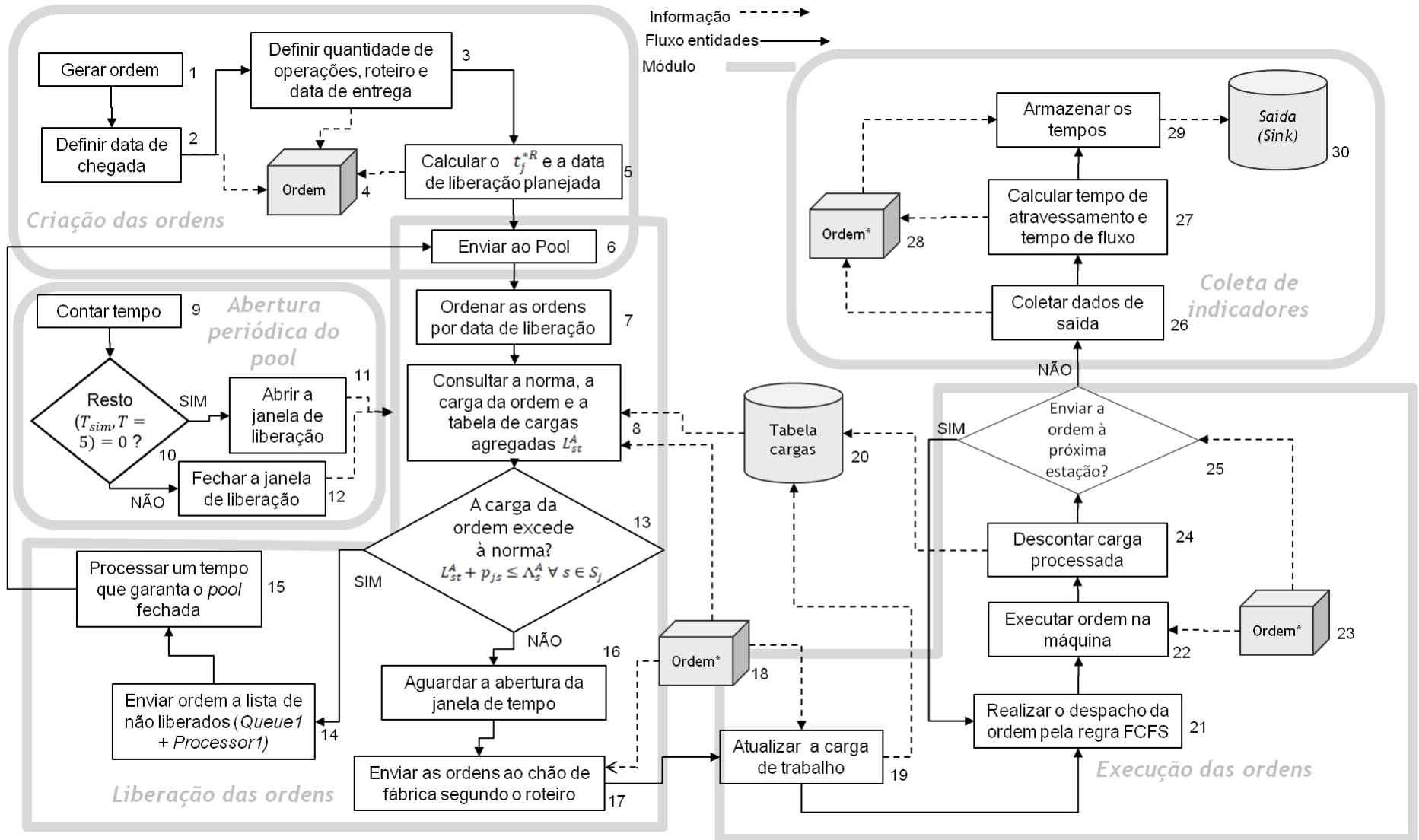
A possibilidade de edição de um código personalizado torna o modelo simulado mais aderente ao modelo real, permitindo o programa executar funções específicas que não estão disponíveis nas configurações padrão do *software*.

4.2. VISÃO MACRO DO MODELO

A Figura 18 apresenta o fluxograma de como é executada a simulação. O fluxograma é dividido em cinco módulos que representam as funções macro do modelo: criação de ordens, abertura periódica *pool*, liberação das ordens, execução das ordens e coleta de indicadores.

Os processos que ocorrem no modelo, contidos dentro dos módulos, são representados por blocos, os quais foram numerados e seguem o fluxo indicado por uma seta contínua. Há também a representação de três objetos do modelo: ordem, tabela de cargas e saída, que são representados para demonstrar o momento em que um determinado processo faz a leitura ou atualiza as informações contidas dentro de cada objeto para as tomadas de decisões. Esse fluxo de informações é indicado por uma seta pontilhada.

Figura 18 - Fluxograma detalhado do funcionamento do modelo traduzido



Fonte: Elaborado pelo autor

O primeiro módulo é o de criação das ordens. É neste módulo que as ordens com seus atributos iniciais serão criados. O bloco 1 é responsável por criar as ordens, no software representadas pelo item *box* (bloco 4) e criadas na entidade *source* (entrada). O bloco 2 define a data de chegada da ordem, que ao final do modelo será utilizada para o cálculo do tempo de atravessamento total. O bloco 3 define a quantidade de operações do roteiro de fabricação, variando de 1 a 6, a sequência de operações/máquinas de cada ordem e a data de entrega da ordem. O bloco 5 calcula os tempos de processamento para cada roteiro de cada ordem e a data de liberação planejada. Feito isso, a ordem é enviada ao *pool*, que é uma entidade do tipo *queue* (fila) no FlexSim, e dá-se início ao segundo módulo de liberação de ordens.

Este segundo módulo, de liberação de ordens, funcionará paralelamente ao módulo de abertura periódica do *pool*. Esse módulo se inicia quando as ordens vão chegando ao *pool* (bloco 6). Elas são ordenadas em ordem crescente pela data de liberação planejada, calculada pela diferença entre a data de entrega e a soma dos tempos de processamento de cada ordem (bloco 7). No bloco 8 o algoritmo consulta na ordem a carga que gerará ao sistema (o tempo de processamento para cada estação de trabalho) e, na tabela de cargas, a norma e as cargas agregadas já alocadas a cada estação de trabalho. Consultado esses dados, o bloco 13 irá verificar, para cada estação de trabalho, se os tempos de processamento da ordem não excedem à norma. Para isso, os tempos de processamento da ordem são somados às cargas agregadas de cada estação de trabalho que fazem parte do seu roteiro de fabricação e o valor dessa soma é comparado com a norma de carga. Caso o valor encontrado exceda à norma, a ordem segue para o bloco 14, que funciona como uma lista de não liberados, no qual passará pelo conjunto *queue* e *processor*, processo um período de tempo, bloco 15, e retorna ao *pool*. Caso não exceda à norma, as ordens aguardam a abertura da janela de tempo para serem liberadas ao chão de fábrica.

O módulo de Abertura periódica do *pool* contém os procedimentos que irão controlar a abertura e fechamento da janela de tempo do *pool*. Este módulo se inicia no momento em que o programa é executado. O primeiro procedimento, bloco 9, irá contar o tempo através da função *time* () do FlexSim. O próximo passo, mostrado no bloco 10, é verificar se o resto da divisão da função *time* () por 5 é igual a zero. Esse cálculo é realizado continuamente e, sempre que o resto dessa operação for zero. Isso indica que o valor contido em naquele instante na função *time*() é múltiplo de 5, intervalo escolhido para abertura da janela de tempo. Assim, toda vez que o valor de *time*() atingir esses múltiplos de 5, a saída do *pool* é aberta e as ordens que não excederam à norma são liberadas ao chão de fábrica. Do mesmo

modo, toda vez que o valor de *time()* não for múltiplo de cinco, a saída do *pool* se manterá fechada (blocos 11 e 12).

Caso o valor encontrado da soma dos tempos de processamento da ordem com a carga agregada de cada estação de trabalho do seu roteiro não ultrapasse à norma, a ordem segue para o próximo procedimento, mostrado no bloco 16, onde irá aguardar a abertura da janela de tempo. Após aberta, o procedimento do bloco 17 enviará as ordens ao módulo de Execução das Ordens, que irá simular o chão de fábrica através do processamento das ordens nas estações de trabalho. Assim, o programa irá ler nas etiquetas (*labels*) de cada ordem o roteiro contendo o número de operações do roteiro e a sequência que essas operações irão ser processadas.

O módulo de Execução das ordens inicia-se no bloco 19 com a leitura dos tempos de processamento das ordens e escrita (atualização) dessa carga na tabela de cargas (procedimento 20). Nesse modelo, a carga de trabalho de uma estação é dada pela soma dos tempos de processamento registrados nas *labels* das ordens. No primeiro momento em que a ordem é liberada ao chão de fábrica e segue para a fila da primeira máquina do seu roteiro, todas as cargas são registradas na tabela de cargas, tanto a carga direta, isto é, o tempo de processamento da primeira estação em que a ordem passará, quanto a carga indireta, ou seja, o tempo de processamento nas máquinas onde a ordem possui roteiro. Cada coluna dessa tabela de cargas recebe o valor acumulado da carga em uma dada máquina.

Uma vez na fila do processamento (bloco 21), as ordens são liberadas de acordo a regra *First Come, First Served*, onde a primeira ordem que chega à fila, é a primeira a ser liberada à máquina (*processor*).

No bloco 22 será feita a leitura do tempo de processamento da ordem para aquela máquina, contido no *label* do item, e executará a ordem baseado nesse tempo de processamento. Após ser processada, o próximo bloco (24) lê a tabela de cargas e desconta nessa tabela o valor da carga processada (tempo de processamento gasto por aquela operação). O último passo do módulo de execução de ordens está contido no bloco 25, no qual é responsável por enviar a ordem à próxima máquina do roteiro ou à saída, caso a última operação tenha sido executada. Para tal, este procedimento irá ler na *label* do roteiro contido na ordem qual roteiro é a próxima operação desse item. Caso ainda haja operações a serem executadas, a ordem seguirá pela saída “sim” e retornará ao bloco 21. Caso contrário, saída “não”, a ordem seguirá para o último módulo: coleta de indicadores.

Este último módulo se inicia com o bloco 26, coleta de dados de saída, no qual são coletados da ordem todos os tempos relativos àquela ordem, bem como o tempo de entrada e

o tempo de saída naquele instante. O bloco 27 realiza o cálculo do tempo de atravessamento baseado nesses tempos de entrada e saída; o bloco 28 é responsável por armazenar esses tempos calculados nas *labels* da saída e, com isso, a ordem é eliminada.

O Quadro 4 apresenta a base dos procedimentos descritos na Figura 18 e descreve quais variáveis são modificadas por cada evento, sua relação com as medidas de desempenho, os objetos principais com as suas atividades ou esperas e a função macro dos algoritmos utilizados. Assim, são descritos em cinco colunas: os módulos e submódulos, objetos, atributos e abas dos objetos, rotinas e comandos implementados e linha do algoritmo geral do WLC. O Quadro 4 também indica entre parêntesis a nomenclatura usada pelo *software* para cada componente.

Quadro 4 – Conversão do algoritmo geral de controle de carga em um modelo implementado em software comercial (*FlexSim*)

Módulos/ submódulos	Objetos	Atributos / abas dos objetos	Rotinas/comandos implementados	Linha do algoritmo geral do WLC (página 31)
Criação das ordens	Gerador de entidades (<i>Source</i> “Chegada”)	Aba <i>source</i>	Definição da data de chegada	A_j
		Aba <i>Triggers, On Creation, Set label + código personalizado</i>	Criação das etiquetas, definição do número de operações, roteiro de produção, tempos de processamento e datas de entrega	S_j p_{js} δ_j
		Aba <i>Triggers, On Creation, código personalizado</i>	Cálculo do tempo de atravessamento planejado, cálculo da data de liberação planejada	$T_s^{*D} = \sum_{s \in S_j} p_{js}$ $t_j^{*R} = A_j + T_s^{*D}$
Abertura periódica do <i>pool</i>	Source TIC	Aba <i>Trigger,</i> Evento: <i>OnExit</i>	Contador de tempo de 5 unidades (segundos) Verificação de liberação aberta ou fechada	T
	Item <i>Cylinder</i>		Item transitório do <i>Source</i> TIC a <i>Sink</i> TAC	
Liberação das ordens	<i>Job</i> <i>pool</i> /fila de ordens (<i>Queue</i>)	Aba <i>Triggers</i> – <i>On entry</i> – <i>Sort by</i> <i>Expression</i>	Ordenação ascendente (<i>Ascending</i> , disponível no software)	Ordenação das tarefas com base na data de liberação planejada: $t_j^{*R} = \delta_j - \sum_{s \in S_j} T_s^{*D}$
		Aba <i>Flow,</i> <i>Send to Port,</i> código personalizado	Consulta das cargas da Tabela Cargas. Avalia se o tempo de processamento da ordem a liberar somado à carga da máquina excede à norma. Caso não exceda à norma, soma à carga existente e direciona ao chão de fábrica. Caso exceda a norma, direciona ao	Leitura de L_{st}^A Avaliar a carga com a norma: $L_{st}^A + p_{js} \leq \Lambda_s^A \forall s \in S_j$

			reingresso da fila.	
	Reingresso das ordens	Aba <i>Processor</i> , <i>Processor1</i>	Espera de um tempo para garantir que a liberação foi encerrada no contador de tempo, depois permite que as ordens não liberadas retornem ao <i>pool</i> .	
Execução das ordens	Fila por máquina "Fila Est i" (Queue)	Aba <i>Trigger</i> , <i>On Entry</i>	Leitura dos tempos de processamento e atualização de carga do chão de fábrica com as ordens ingressantes	$L_{st}^A = L_{st}^A + p_{js} \quad \forall s \in S_j$
	Máquinas (Processor)	Aba <i>Process</i> , em <i>Process time</i>	Execução da operação correspondente	Operação com tempo de processamento p_{js}
		Aba <i>Trigger</i> , Evento <i>On Entry</i>	Contador de objetos processados em cada máquina	
		Aba <i>Trigger</i> , Evento <i>On Exit</i>	Atualização da carga já processada, descontando-se a ordem já processada.	$L_{st}^A = L_{st}^A - p_{js}^+ \quad \forall s \in S_j$
	<i>Sink</i> Saida	Aba <i>Triggers</i> , Evento <i>On Entry</i>	Coleta do momento da saída da ordem Cálculo do tempo de atravessamento total e do tempo de atravessamento no chão de fábrica (tempo de fluxo).	$T'_j = D_j - A_j$
Finalização das ordens e coleta de indicadores	<i>Sink</i> Saida	Aba <i>Triggers</i> , Evento <i>On Entry</i>	Leituras do tempo de atravessamento total, do tempo de fluxo em cada entidade, na fonte (<i>fonte source</i>) e no <i>pool</i> .	$\hat{w}(n) = \sum W_j$

Fonte: Elaborado pelo autor

Na primeira coluna do Quadro 4 estão descritos os cinco módulos/submódulos que também dão nome aos tópicos dessa seção: Criação das ordens (4.4), Abertura periódica do *pool* (4.5), Liberação das ordens (4.6), Execução das ordens (4.7) e Finalização das ordens e coleta de indicadores (4.8).

O módulo de Criação das ordens possui o objeto *source* (chegada), que irá criar os itens *box* que serão processados no modelo. Para isso, são configurados atributos tanto no item como no objeto *source*. Como apresentado no tópico 4.1, a configuração desses atributos é realizada na tela de propriedades do objeto. Na aba *source*, por exemplo, tem-se as definições das datas de chegada, que indicam o momento em que o objeto cria os itens. O tipo de chegada/criação do item pode ser por um dado intervalo de tempo ou por uma determinada programação ou até mesmo respeitando um tipo de sequenciamento. Na última coluna é representada a variável ou a equação do algoritmo de liberação de ordens apresentado na literatura que foi utilizado como base para realizar modelagem no *FlexSim*. Ainda referente ao primeiro módulo, Criação das ordens, na aba *On Creation* é onde serão implementados os

comandos, tanto a criação de etiquetas como o código personalizado. O algoritmo completo desse módulo é apresentado no Apêndice B.

O segundo módulo: Abertura periódica do *pool* é responsável pelo controle da abertura do *pool* para realização da liberação dos itens presentes no objeto *queue (pool)*. Para isso, esse módulo é composto por dois objetos (*source TIC sink TAC*) que estão conectados entre si, sendo que o TIC também está conectado ao *pool*. Como a *source* tem a função de criar um item para ser processado, o TIC cria um item *Cylinder*, que é prontamente destruído no TAC, sem que sofra nenhum tipo de alteração, já que a única função programada nesse conjunto TIC – TAC é o controle da abertura periódica do *pool*. Esse controle é implementado na aba *trigger*, considerando o tempo de abertura periódica do *pool* igual a 5 segundos, programado por meio de um código personalizado, apresentado no Apêndice C e explicado com detalhes no tópico 4.5.

O terceiro módulo, Liberação das ordens, é configurado no objeto *queue (pool)* e no conjunto de objetos *queue 1* e *processor 1* (representando a lista de ordens não liberadas). Nesse módulo definem-se quais itens (ordens) devem ser liberados ao chão de fábrica e quais itens não devem ser liberados, sendo estes enviados ao conjunto dos objetos *queue 1* e *processor 1*, representando neste trabalho a lista de ordens não liberadas, que após um período de tempo retornaram ao *pool* para nova verificação de liberação. O mecanismo de liberação deste módulo considera a data de liberação planejada de cada ordem para ordená-las. Após ordenadas, é verificado se a soma da carga das máquinas com os tempos de processamento da ordem em cada máquina do seu roteiro excede a norma. Caso exceda, a ordem é enviada à lista de ordens não liberadas, que retornará ao *pool* para ser novamente ordenada para a próxima abertura periódica. Caso não exceda, a ordem segue ordenada para ser liberada ao chão de fábrica. Os detalhes dessa implementação está apresentado no tópico 4.6 e o algoritmo completo nos Apêndices D, E e F.

O quarto módulo, Execução das ordens, é responsável por processar os itens que são liberados pelo *pool*. Os objetos utilizados nesse módulo são seis *queues*, representando as filas em frente às máquinas, seis *processors*, representando as máquinas, e um objeto *sink*, representando a saída. As ordens se posicionam nas filas de acordo o momento de chegada de cada uma (FIFO). Liberadas para serem processadas, as ordens chegarão às máquinas e serão processadas com base nos tempos de processamento registrados nas etiquetas de cada ordem, que são lidas pelas máquinas. Neste módulo, o objeto *sink* é responsável pelo cálculo dos tempos de atravessamento total e do tempo de atravessamento do chão de fábrica das ordens.

Os detalhes dessa implementação estão apresentados no tópico 4.7 e o algoritmo completo nos Apêndices G, H e I.

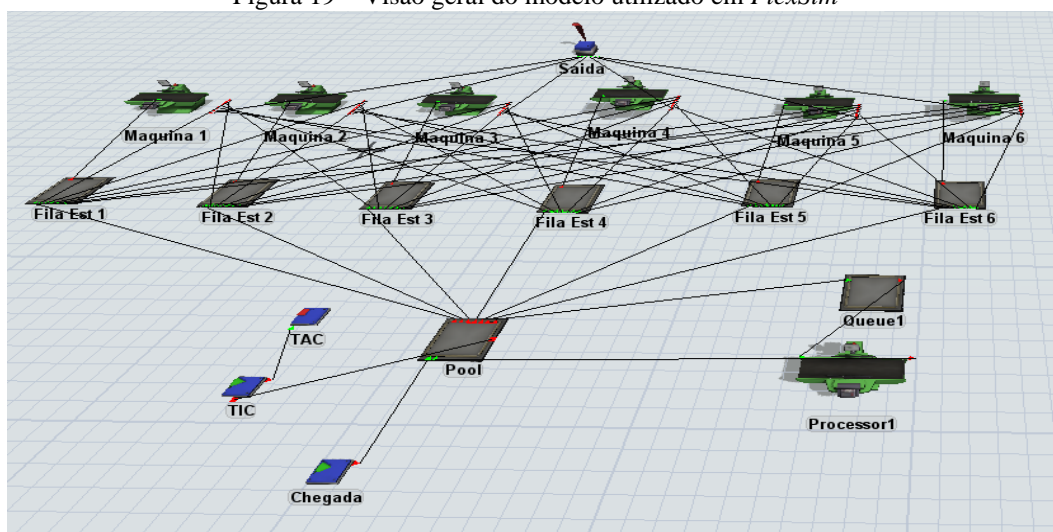
O quinto e último módulo, Finalização das ordens e coleta de dados, é responsável por coletar as informações calculadas e registradas nas ordens, por registrar os tempos de atravessamento total e o tempo de atravessamento do chão de fábrica de cada ordem e por finalizar o processo, destruindo as ordens. O objeto que compõe esse módulo é o *sink* (saída) e as implementações dos comandos são realizadas na janela de propriedades. Os detalhes dessa implementação está apresentado no tópico 4.8 e o algoritmo completo no Apêndice J.

4.3. CONSTRUINDO ESTRUTURA DE CONVERSÃO NA FERRAMENTA *FLEXSIM*

A criação da estrutura de conversão no *software* FlexSim baseou-se nos módulos e etapas do fluxograma da Figura 18 e no Quadro 4. A Figura 19 apresenta os objetos utilizados no *software* do modelo estudado. Nota-se nesta figura a presença de linhas ligando os objetos, essas linhas indicam a relação de entrada e saída de informações, na qual essas informações entram nos objetos, são processadas e saem para um dos objetos ligados que está enviando a informação.

Na Figura 19, os objetos representados para cada evento são: i) o *source* “Chegada” que realiza a entrada da ordem ao sistema; ii) o *pool*, e o sistema de reingresso no *pool* (*Queue1* + *Processor1*), o contador de tempo (*Source TIC* + *Sink TAC*) para efetuar a Abertura periódica do *pool*; iii) Os objetos “Fila Est” 1–6 , “Maquina” 1–6 para a execução da ordem; e iv) e o *Sink* “Saida” para a finalização da ordem e a primeira fase da obtenção de indicadores

Figura 19 – Visão geral do modelo utilizado em *FlexSim*



Fonte: Adaptado de Land (2006)

Os próximos subcapítulos desta pesquisa demonstram a forma como cada módulo da Figura 18 e do Quadro 4 foram construídos.

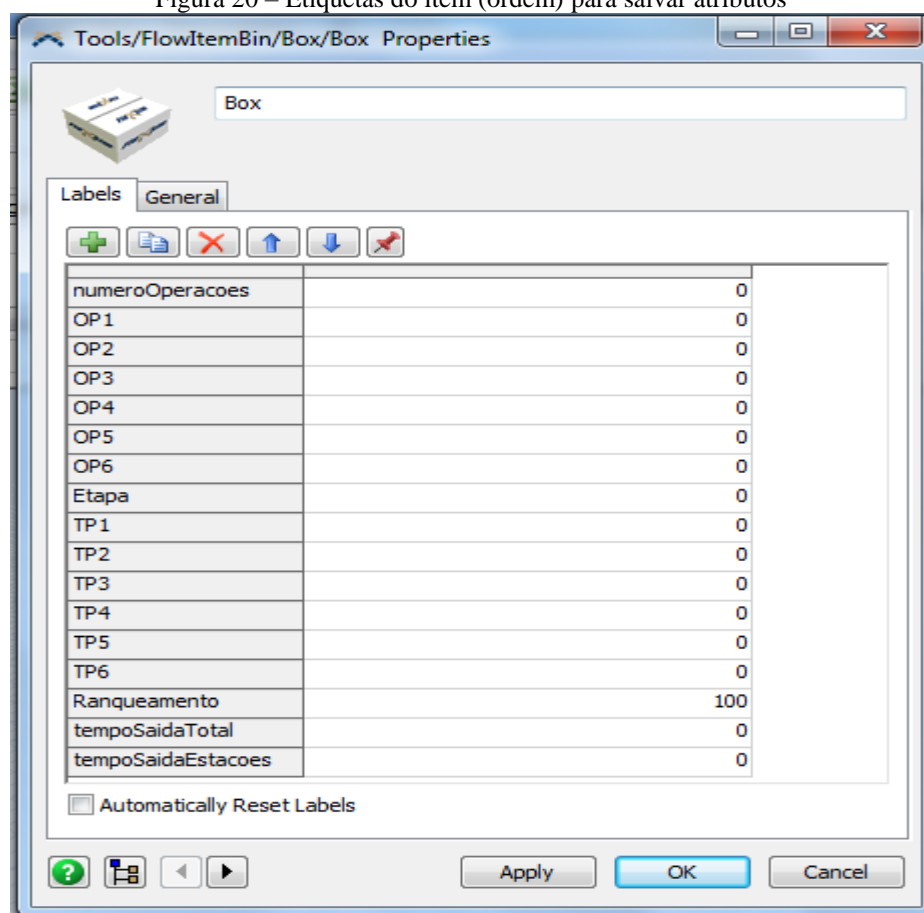
4.4. CRIAÇÃO DAS ORDENS

O modelo considera o item como elemento principal para representar uma ordem de trabalho, isto é, um pedido que já foi aceito para fabricação. No modelo, a ordem de trabalho é representada no FlexSim pelo item “*box*”. Para definir os atributos da ordem de trabalho, usou-se uma etiqueta (*label*). As etiquetas permitem salvar dados nas ordens de trabalho para tomadas decisões no modelo. O detalhe das etiquetas para a ordem de trabalho aparece na Figura 20.

A etiqueta “numeroOperacoes” permite salvar a quantidade de operações no roteiro da ordem. As etiquetas OP1 até OP6 armazenam qual máquina deve passar a ordem, segundo a sua sequência de operações. Caso a etiqueta seja OP3 = 4, indica que a terceira operação de processamento daquela ordem deverá ocorrer na máquina 4. A etiqueta “Etapa” memoriza o número da operação, segundo a sequência de operações do roteiro. Se “Etapa” tem o valor de 2, a ordem de trabalho estará na segunda operação a realizar segundo a sua sequência. As etiquetas TP1 a TP6 armazenam o tempo de processamento da ordem para a sua operação definida. Esses tempos de processamento são utilizados para o cálculo das cargas. Considerando o exemplo da etiqueta OP3, a TP3 = 1.2, implica que a ordem tardará 1.2 unidades de tempo na máquina 4 (por ser a indicada na etiqueta OP3). A etiqueta “Ranqueamento”, que já inicializa com o valor 100, serve para estipular uma preferência à ordem quando ela não pode ser liberada, segue para a lista de espera e retorna ao *pool*. Esta etiqueta será explicada no evento de Liberação de Ordens (seção 4.2). As etiquetas “TempoSaidaTotal” e “TempoSaidaEstacoes” permitem armazenar o momento de finalização da ordem de trabalho e momento de liberação da ordem, respectivamente.

Foram criadas três tabelas gerais para armazenar dados e visualizar alguns resultados: i) a Tabela Cargas (Figura 21), a Tabela do Histórico de Cargas e a Tabela *Throughput* (as duas tabelas restantes serão explicadas na seção de obtenção de indicadores).

Figura 20 – Etiquetas do item (ordem) para salvar atributos



Fonte: Elaborado pelo autor

A Tabela Cargas contém uma fila única e onze colunas. As seis primeiras colunas armazenam a carga temporária das máquinas e são utilizadas para avaliar a carga atual das respectivas máquinas na liberação das ordens. A sétima coluna contém a norma limite de carga de trabalho para todas as máquinas. A oitava coluna serve para salvar o valor do tempo de atravessamento total dos itens, quanto que a décima primeira coluna armazena o valor do tempo de fluxo no chão de fábrica (*Throughput* chão de fábrica). Já a nona coluna é um contador de tempo da execução da simulação que considera cinco unidades de tempo e a décima coluna é um indicador se o *pool* está aberto liberando ordens. Estas duas últimas colunas (9 e 10) são verificadores temporais durante a simulação. As outras duas tabelas apenas são utilizadas para a coleta de dados nos itens.

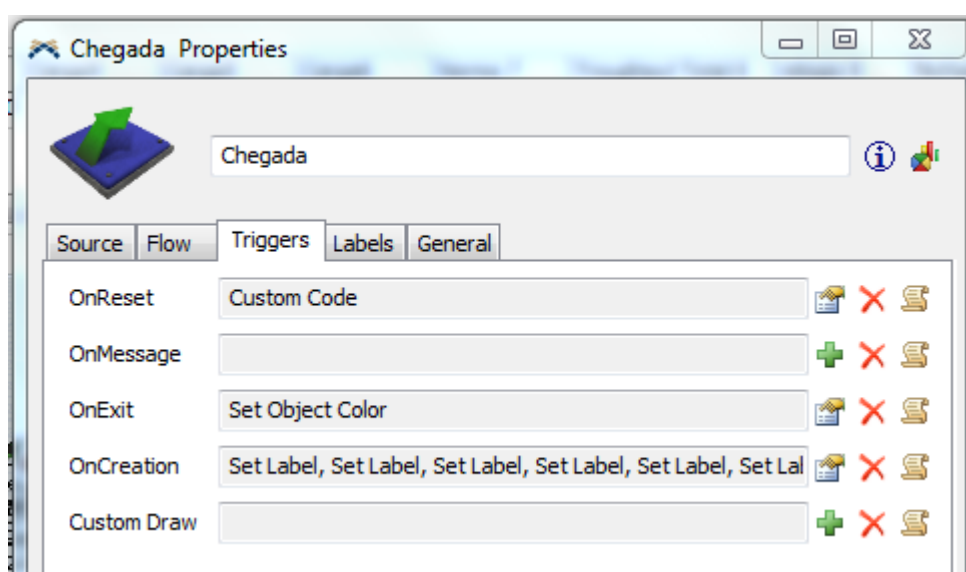
Figura 21 – Detalhe da Tabela cargas

	Carga1	Carga2	Carga3	Carga4	Carga5	Carga6	Norma 7	Troughput Total 8	relógio 9	fechado 10	Throughput Chão de Fábrica 11
Row 1	1.63	6.11	4.73	5.00	4.68	0.00	5	5.79	95	1	5.79

Fonte: Elaborado pelo autor

As ordens saem do *source* “Chegada” considerando o tempo de intervalo entre chegadas da tabela 4 e seguem para o *pool*. A determinação dos atributos dentro do *source* “Chegada” segue uma sequência de eventos internos, apresentada em detalhe na Figura 22. Para o evento de reinício da simulação (“*OnReset*”), determina-se um valor de zero para a tabela que controla as cargas das estações. Para o evento de saída do item da entidade *source*, é determinado uma cor para o item de forma aleatória. No evento de criação do item (*OnCreation*), são escritas etiquetas (*labels*) para criar os atributos que será utilizados na simulação.

Figura 22 – Configuração do source “Chegada”



Fonte: Elaborado pelo autor

Na definição dos atributos, programou-se que o *source* definisse as etiquetas na seguinte ordem: i) quantidade de operações da ordem, ii) data de entrega da ordem, e iii) a sequência de máquinas, ou seja, o roteiro de produção. Para definir a quantidade de operações, a função utiliza a distribuição discreta uniforme entre 1 e 6, como mostrado nas linhas 7 a 10 da Figura 23. A definição da data da entrega da ordem considera o momento em que o item é criado (com a função: *time* ()) e adiciona um intervalo uniforme entre 35 e 60 unidades de tempo, como mostra a linha 14. Para definir o roteiro, o *source* salva o número de operações em OPS (linha 18) e cria um vetor de sequência de operações OPE de seis elementos para salvar o número da máquina, na qual será realizada cada operação (linha 21). As primeiras quatro linhas são parte do código predefinido do *source*.

Figura 23 – Etiquetas do item para salvar atributos

```

Chegada - OnCreation
1 Object item = param(1);
2 Object current = ownerobject(c);
3 int port = param(2);
4 int rownumber = param(2); //row number of the schedule/sequence table
5 { // ***** PickOption Start ***** //
6 /**popup:SetLabel:hasitem=0*/
7 /**Set Label*/
8 Object involved = /** \nObject: /**tag:object/**/item/**/;
9 string labelname = /** \nLabel: /**tag:label/**/"numeroOperacoes"/**/;
10 Variant value = /** \nValue: /**tag:value/**/duniform(1,6)/**/;
11 involved.labels.assert(labelname).value = value;
12 } // ***** PickOption End ***** //
13
14 setlabel(item,"tempoChegada",time()); //adiciona uma label de tempo de chegada (dia de chegada)
15
16 setlabel(item,"diaEntrega",duniform(35,60)+time()); // adiciona uma label com a data de entrega prevista
17
18 int OPS = getlabel(item,"numeroOperacoes"); // Adiciona o número de operações em OPS
19
20
21 intarray OPE = makearray(6); //Cria um vetor OPE que irá receber os valores de OP1 a OP6

```

Fonte: Elaborado pelo autor

A Figura 24 apresenta a definição do tempo de processamento da primeira operação da sequência. A primeira máquina do roteiro se define por meio de uma distribuição discreta uniforme entre 1 e 6 (linhas 29 - 30). Como a primeira operação já foi estabelecida, o contador de operações OPS decresce uma unidade para destacar que faltam $n-1$ operações por definir (linha 36). Programou-se que o *source* escrevesse o tempo de processamento na etiqueta TP1 usando a distribuição *2-Erlang* com parâmetro de escala de 0.5 (linha 38).

Figura 24 – Definição do tempo de processamento no item, operação 1

```

Chegada - OnCreation
22
23 //Comando OP1:
24
25
26 { // ***** PickOption Start ***** //
27 /**popup:SetLabel:hasitem=0*/
28 /**Set Label*/
29 Object involved = /** \nObject: /**tag:object/**/item/**/;
30 string labelname = /** \nLabel: /**tag:label/**/"OP1"/**/;
31 Variant value = /** \nValue: /**tag:value/**/duniform(1,6)/**/;
32
33 involved.labels.assert(labelname).value = value;
34 } // ***** PickOption End ***** //
35
36 OPS = (OPS -1); // Decrementa o número de operações
37
38 setlabel(item,"TP1",erlang(0.0, 0.5, 2.0, 0)); //cria a tempo de processamento para OP1

```

Fonte: Elaborado pelo autor

A definição do tempo de processamento das outras operações do roteiro segue uma lógica semelhante. A Figura 25 apresenta o detalhe da definição da sexta operação; omitiu-se o detalhe entre a segunda e a quinta pelas semelhanças com a sexta. No início, o algoritmo verifica se precisam ser definidas mais operações, verificando se o contador de operações não definidas (OPS) é maior do que zero (linha 167). Então, a etiqueta OP6 recebe um valor discreto uniforme entre 1 e 6 (linhas 169 - 174). A forma de não repetir a máquina no roteiro da ordem, o programa avalia se a máquina definida para OP6 já foi designada em algum passo anterior à sequência (linhas 183). Se não foi designada, o *source* confirma a máquina como parte da sequência. Caso tenha sido designada anteriormente em qualquer operação, determina-se uma nova máquina (linha 184). A definição da máquina finaliza quando uma máquina válida é encontrada para a OP6, (entre linhas 182 - 189). O contador de operações não definidas diminui em uma unidade (linha 191). Para definir o tempo de processamento, o *source* escreve o tempo de processamento na etiqueta correspondente a sua operação, sendo TP6 a partir da distribuição definida (linha 194).

Figura 25 – Definição do tempo de processamento no item, outras operações

```

Chegada - OnCreation
167 if (OPS > 0) { //Comando OP6
168
169 { // ***** PickOption Start ***** //
170 /**popup:SetLabel:hasitem=0*/
171 /**Set Label*/
172 Object involved = /** \nObject: /**tag:object/**/item/**/;
173 string labelname = /** \nLabel: /**tag:label/**/"OP6"/**/;
174 Variant value = /** \nValue: /**tag:value/**/duniform(1,6)/**/;
175
176 involved.labels.assert(labelname).value = value;
177 } // ***** PickOption End ***** //
178
179
180 int status = 0;
181
182 while(status==0){
183     if (getlabel(item,"OP6")==getlabel(item,"OP1") || (getlabel(item,"OP6")==getlabel(item,"OP2")) || (getlabel(item,"OP6")==getlabel(item,"OP3"))
184         setlabel(item,"OP6",duniform(1,6));
185     }
186     else{
187         status = 1;
188     }
189 } //end while
190
191 OPS = (OPS -1); // Decrementa o número de operações
192
193
194 setlabel(item,"TP6",erlang(0.0, 0.5, 2.0, 0)); //cria a tempo de processamento para OP6
195
196 } //end comando 6.
197

```

Fonte: Elaborado pelo autor

Os tempos de processamento definidos para cada máquina do roteiro são transferidos ao vetor TP (linhas 207-212 da Figura 26). A partir dos valores de tempo transferidos ao vetor, calcula-se o tempo de atravessamento planejado da ordem. Ele considera o tempo de criação da ordem e a somatória de todos os tempos de processamento (linha 215), sendo equivalente ao tempo desconsiderando os tempos de espera nas filas das máquinas e no *pool*. A liberação planejada da ordem é armazenada em uma etiqueta que estima a folga existente para terminar a ordem. Essa folga considera a data de entrega menos o tempo de atravessamento. Para conferir que não faltam operações, o contador é zerado (linha 225 - 226).

Figura 26 – Transferência do tempo de processamento e cálculo da data de liberação planejada

```

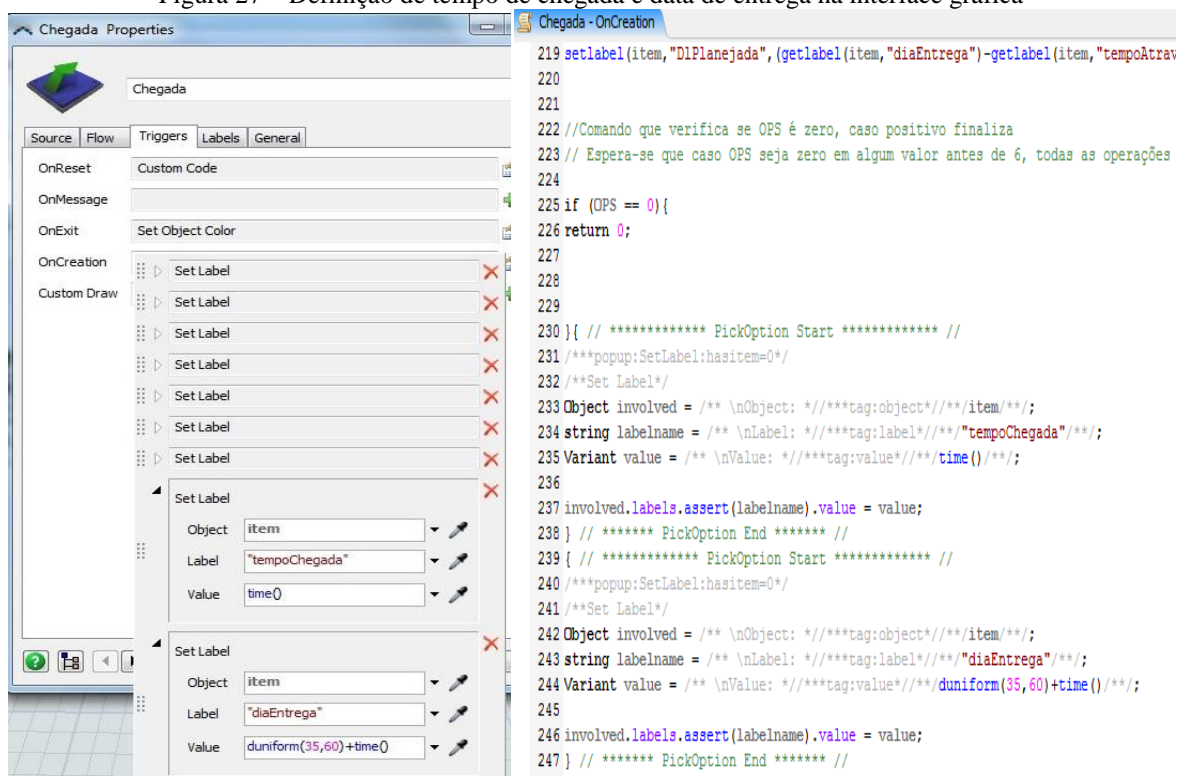
Chegada - OnCreation
203
204
205 intarray TP = makearray(6);
206
207 TP[1] = getlabel(item,"TP1");
208 TP[2] = getlabel(item,"TP2");
209 TP[3] = getlabel(item,"TP3");
210 TP[4] = getlabel(item,"TP4");
211 TP[5] = getlabel(item,"TP5");
212 TP[6] = getlabel(item,"TP6");
213
214
215 setlabel(item,"tempoAtravessamento",((TP[1]+TP[2]+TP[3]+TP[4]+TP[5]+TP[6])+(getlabel(item,"tempoChegada"))));
216
217
218 //cria lebal data de liberação planejada
219 setlabel(item,"DlPlanejada", (getlabel(item,"diaEntrega")-getlabel(item,"tempoAtravessamento")));
220
221
222 //Comando que verifica se OPS é zero, caso positivo finaliza
223 // Espera-se que caso OPS seja zero em algum valor antes de 6, todas as operações não preenchidas recebam zero.
224
225 if (OPS == 0){
226 return 0;

```

Fonte: Elaborado pelo autor

A Figura 27 apresenta o detalhe da interface gráfica e a linha de comandos para definir o tempo de chegada (linhas 230 - 237), assim como a data de entrega (linhas 239 - 247).

Figura 27 – Definição de tempo de chegada e data de entrega na interface gráfica



Fonte: Elaborado pelo autor

4.5. ABERTURA PERIÓDICA DO POOL

O mecanismo criado para controlar a abertura periódica do *pool* para a liberação das ordens é o TIC-TAC (Figura 28), mencionado na seção 4.1. Este mecanismo é composto por dois objetos (*source* e *sink*) ligados ao objeto *queue* que representa o *pool*. O TIC-TAC é programado para abrir e fechar a porta (saída) do *pool* em um intervalo de tempo, liberando as ordens por meio dos comandos: *openoutput* e *closeoutput*, como se pode verificar pelas linhas dos comandos extraídas do modelo, na aba *OnExit* do objeto TIC.

Figura 28 – Implementação da Abertura periódica do *pool*

```

TIC - OnExit*
1 /**Custom Code*/
2 Object item = param(1);
3 Object current = ownerobject(c);
4 int port = param(2);
5
6 if (fmod(time(),5) == 0) {
7
8     openoutput(centerobject(current,1));
9     setlabel(current,"fechado",0);
10    settablenum("Cargas",1,9,time());
11    // escreve se a pool esta aberta ou fechada
12    settablenum("Cargas",1,10,getlabel(current,"fechado"));
13
14 }
15 if (fmod(time(),5) != 0) {
16
17     closeoutput(centerobject(current,1));
18     setlabel(current,"fechado",1);
19     // escreve se a pool esta aberta ou fechada
20     settablenum("Cargas",1,10,getlabel(current,"fechado"));
21

```

Fonte: Elaborado pelo autor

Embora este algoritmo crie m item a cada 1 unidade de tempo, este item logo é destruído no *sink* TIC pois o intuito deste mecanismo é a realização do processo de abertura e fechamento do *pool*. O *source* TIC avalia o resto da divisão do tempo por cinco unidades (Linha 6). Se a condição é cumprida, ou o resto da divisão é zero, o *source* abre a sua saída para os itens criados (Linha 8); cria a etiqueta de fechado dentro dele mesmo (Linha 9); transfere o tempo atual simulado para tabela carga, na coluna relógio (Linha 10); e escreve um valor de fechado na tabela carga (Linha 11). Esses dois últimos passos são para o controle de verificação da simulação. Caso o tempo seja diferente do múltiplo de 5 unidades (Linha 14), a *source* fecha a saída de ordens (Linha 16), modifica o valor da sua etiqueta de fechado (Linha 17); e reescreve na tabela de cargas que o *pool* está fechado (Linha 19). Já os itens criados no TIC são destruídos no *sink* TAC, sem nenhuma condição. Com isso, as colunas da Tabela cargas chamadas “Relógio 9” e “Fechado 10” permitem verificar o funcionamento da abertura periódica do *pool* para liberação das ordens.

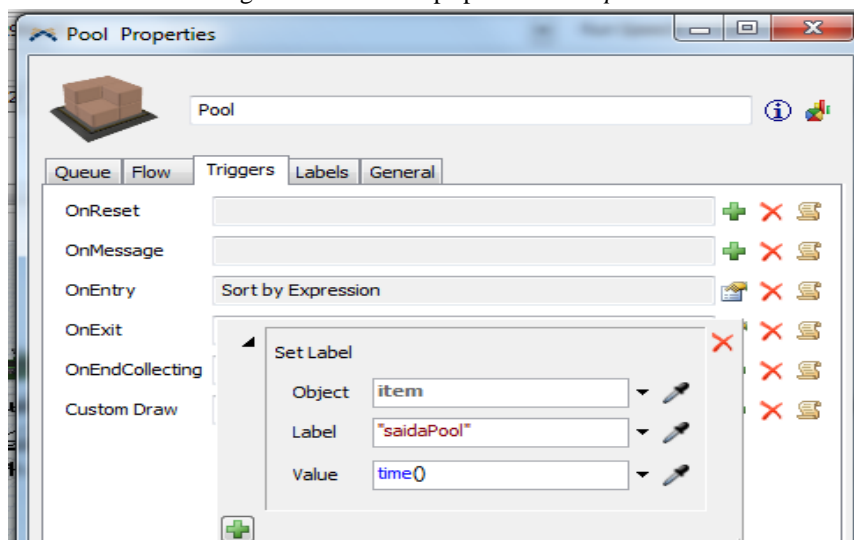
As ordens liberadas são enviadas para o fluxo de estações (chão de fábrica) e seguem para as máquinas de trabalho definidas em seu roteiro. Para cada máquina há uma fila de espera onde as ordens aguardam para serem processadas. No caso em que uma ordem não seja liberada, esta segue para a estação de ordens não liberadas (*Queue1*) e só retornam ao *pool* após o fechamento da janela de tempo, utilizando para isso uma entidade *processor* com o tempo de processamento de 3 unidades de tempo, garantido que no retorno da ordem a janela de tempo do *pool* já esteja fechada.

4.6. LIBERAÇÃO DAS ORDENS

A avaliação das cargas de trabalho para a liberação das ordens acontece no objeto *pool*. Para controlar e verificar a simulação manteve-se o conteúdo máximo de entidades na fila em 1000 unidades. Dentro da fila, existem passos a serem executados: i) a ordenação das ordens de trabalho para definir a sequência em que as mesmas são consideradas, ii) a leitura da carga atual (direta e indireta) em cada máquina, iii) a decisão de liberação de ordens com base na carga atual existente, iv) o armazenamento do tempo no momento de liberação, e v) a definição da porta de saída para que a ordem siga o roteiro desejado. A Figura 29 apresenta a tela de propriedades do *pool*, na aba *triggers*, demonstrando onde foram declaradas as funções (i) e (iv), dando um destaque para quarto passo (armazenamento do tempo no momento da liberação) que foi configurado na linha do *OnExit*, no qual por meio do comando *Set Label*,

armazena o valor do tempo de saída da ordem na *label* “saidaPool”. Já o item (iii) se apresenta em detalhe na aba *Flow* dessa fila.

Figura 29 – Tela de propriedades do *pool*



Fonte: Elaborado pelo autor

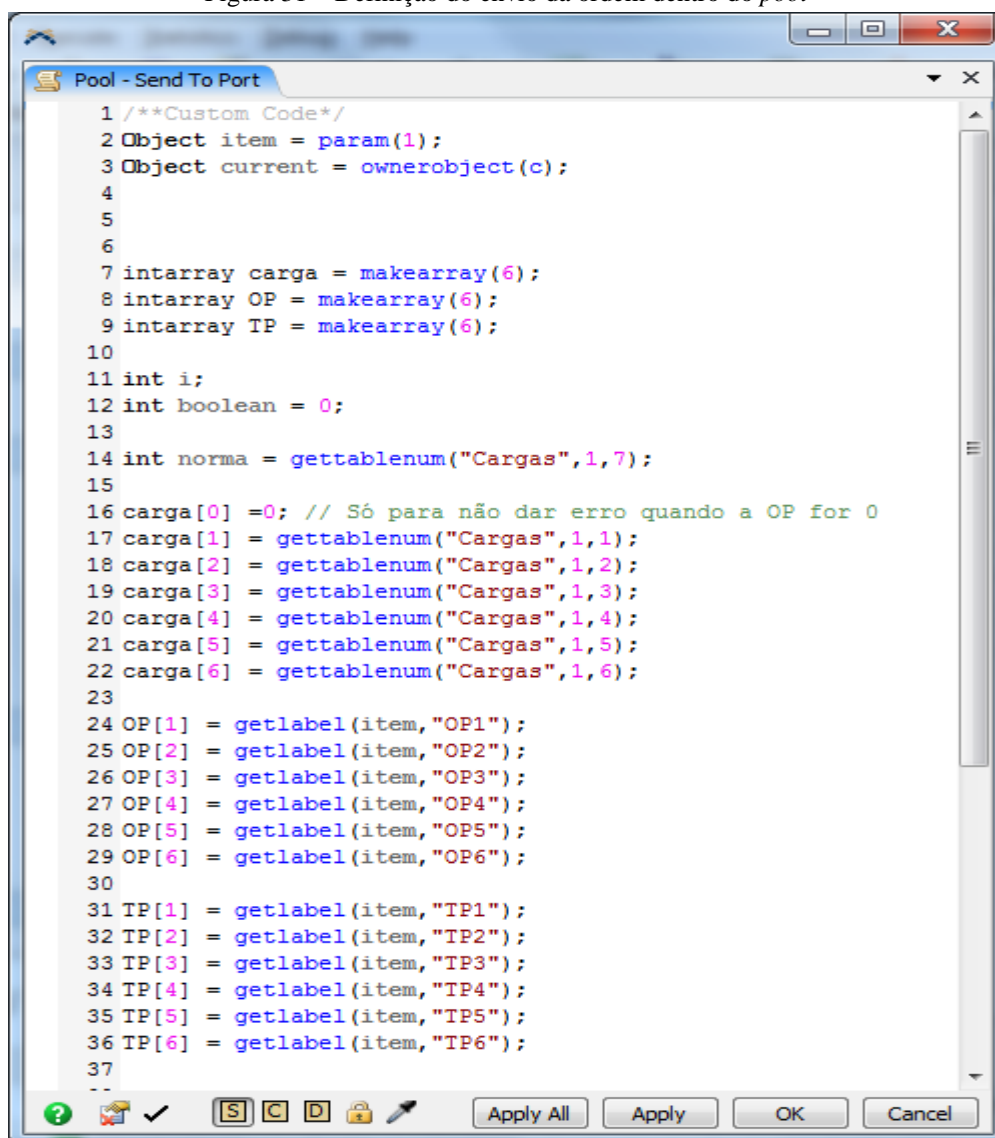
A Figura 30 detalha a ordenação das ordens dentro do *pool* para serem liberadas. O critério de ordenação é do menor ao maior valor de data de liberação planejada. A ordenação ocorre logo que o item entra no *pool*, na linha do *OnEntry*, por meio do comando *sort by expression*, que irá ordenar as ordens de forma ascendente baseados na expressão $(\text{getlabel}(\text{item}, "DIPlanejada") + \text{getlabel}(\text{item}, "Ranqueamento"))$.

Na Figura 30, percebe-se que a data de liberação planejada é somada a uma etiqueta chamada ranqueamento. A etiqueta ranqueamento é utilizada para avaliação de um segundo cenário, não o algoritmo original do WLC, que visa gerar uma prioridade de liberação para as ordens que no primeiro momento não foram liberadas. Ao retornarem ao *pool*, estas recebem uma prioridade de liberação em relação às novas ordens criadas. A lógica do ranqueamento é que caso seja a primeira vez que a ordem passa no *pool*, todas terão o mesmo valor de ranqueamento (100) e, portanto, a classificação dependerá da data liberação planejada. Assim que uma ordem é enviada para a lista de não liberadas, o valor do ranqueamento torna-se zero e, ao retornar ao *pool*, garante-se que essa ordem possua um somatório menor do que as novas ordens que chegam ao *pool* com o valor de ranqueamento igual a 100.

Figura 30 – Classificação das ordens dentro do *pool*

Fonte: Elaborado pelo autor

Para a determinação da liberação da ordem no *pool*, se estabelecem dois passos executados no momento de definir a porta de saída para cada entidade: i) consultar os dados de carga atual de trabalho nas máquinas, bem como o roteiro e os tempos de processamento da ordem a ser considerada para liberação (Figura 31); ii) comparar a carga de trabalho nas máquinas (já somadas aos tempos de processamento da ordem considerada) em relação à norma de carga (Figura 32). Para o primeiro passo, na Figura 31, são criados os vetores de carga de trabalho, roteiro e tempo de processamento (linhas 7 - 9) para o uso do *pool*. Depois, define-se uma variável inteira para armazenar o valor da norma de carga máxima permitida nas máquinas (Linhas 11 - 14). A seguir, o valor da carga atual é consultado na tabela carga para cada máquina (Linhas 16 - 22), seguido pela consulta do roteiro de produção da ordem, armazenado nas etiquetas do item “*box*” (Linha 24 - 29) e pela consulta do seu tempo de processamento (Linha 31 - 36).

Figura 31 – Definição do envio da ordem dentro do *pool*


```

1 /**Custom Code*/
2 Object item = param(1);
3 Object current = ownerobject(c);
4
5
6
7 intarray carga = makearray(6);
8 intarray OP = makearray(6);
9 intarray TP = makearray(6);
10
11 int i;
12 int boolean = 0;
13
14 int norma = gettablenum("Cargas",1,7);
15
16 carga[0] = 0; // Só para não dar erro quando a OP for 0
17 carga[1] = gettablenum("Cargas",1,1);
18 carga[2] = gettablenum("Cargas",1,2);
19 carga[3] = gettablenum("Cargas",1,3);
20 carga[4] = gettablenum("Cargas",1,4);
21 carga[5] = gettablenum("Cargas",1,5);
22 carga[6] = gettablenum("Cargas",1,6);
23
24 OP[1] = getlabel(item,"OP1");
25 OP[2] = getlabel(item,"OP2");
26 OP[3] = getlabel(item,"OP3");
27 OP[4] = getlabel(item,"OP4");
28 OP[5] = getlabel(item,"OP5");
29 OP[6] = getlabel(item,"OP6");
30
31 TP[1] = getlabel(item,"TP1");
32 TP[2] = getlabel(item,"TP2");
33 TP[3] = getlabel(item,"TP3");
34 TP[4] = getlabel(item,"TP4");
35 TP[5] = getlabel(item,"TP5");
36 TP[6] = getlabel(item,"TP6");
37
--

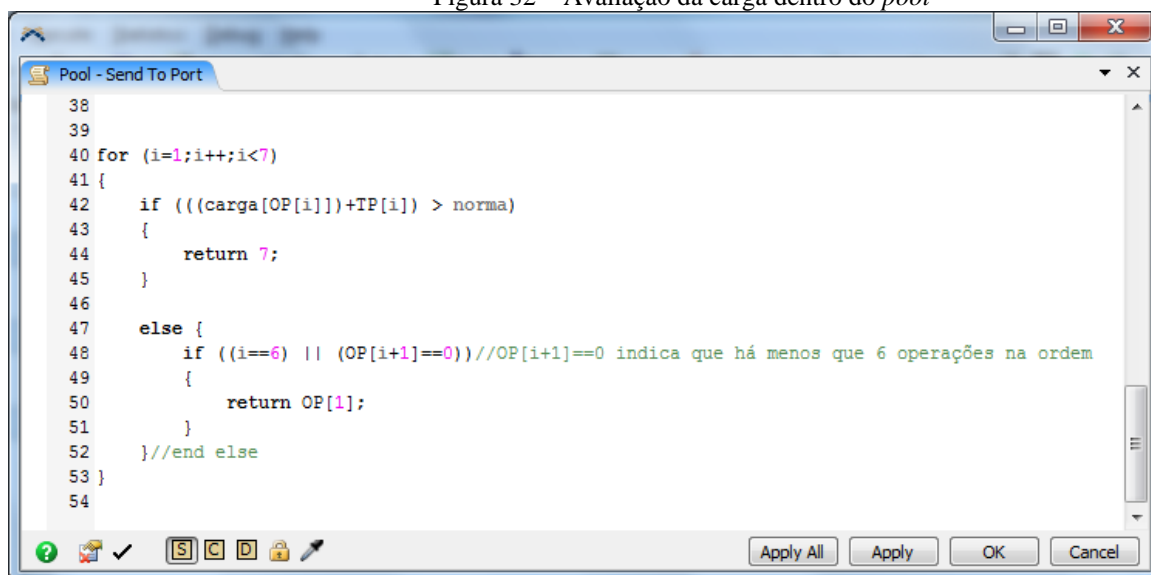
```

Fonte: Elaborado pelo autor

A partir dos dados obtidos, o programa avalia se a carga de trabalho total (carga da máquina somada ao tempo de processamento da ordem) não ultrapassa a norma limite (Figura 32). A avaliação é realizada para as seis máquinas de forma independente (Linha 40). A carga avaliada é a carga atual já alocada nas máquinas (considerando carga direta e indireta) (carga[OP[i]]) mais o tempo de processamento da ordem avaliada (TP[i]). Se a carga for maior que a norma, então a ordem de trabalho é definida para sair pela porta 7, uma porta que se conecta para reingressar no *pool* pelo “*Processor1*”. Se não, ela é encaminhada para a porta da máquina na qual sua primeira operação deve ser executada (linhas 42 - 45). Caso o contador atinja a máquina 6 ou exista algum valor de 0 como máquina na sequência das operações, então indica que a carga não foi excedida em nenhuma máquina e a ordem é encaminhada para a máquina que deve executar a sua primeira operação, cujo número está

armazenado em OP[1] e assim, se termina a rotina (Linhas 48 - 51). Esse último passo é implementado para que o programa não fique sem instruções para terminar a rotina de avaliação.

Figura 32 – Avaliação da carga dentro do *pool*

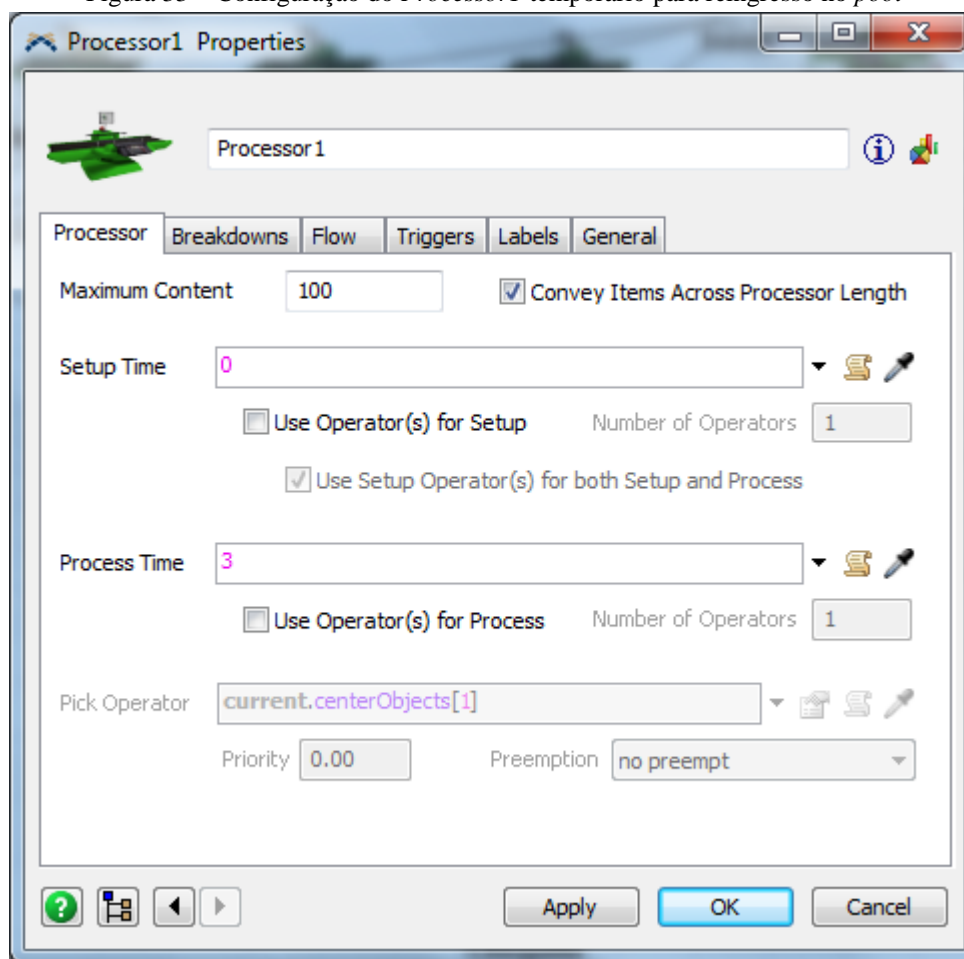


```
38
39
40 for (i=1;i++;i<7)
41 {
42     if (((carga[OP[i]])+TP[i]) > norma)
43     {
44         return 7;
45     }
46
47     else {
48         if ((i==6) || (OP[i+1]==0))//OP[i+1]==0 indica que há menos que 6 operações na ordem
49         {
50             return OP[1];
51         }
52     }//end else
53 }
54
```

Fonte: Elaborado pelo autor

Quando uma ordem não é liberada no *pool*, porque a carga excede a norma, a ordem passa pelo *Queue1* e o *Processor1*. Eles, em sequência, representam apenas atividades temporais na simulação para que as ordens possam ingressar novamente no *pool* na próxima abertura da janela de tempo. A classificação do *pool* aparece já citada na Figura 30. A ordem continua seu passo ao *Processor1*, que apenas demora 3 unidades de tempo para liberar a ordem, apresentado na Figura. O tempo do *Processor1* apenas foi um valor arbitrário menor ao tempo de abertura periódica do *pool*.

Figura 33 – Configuração do *Processor1* temporário para reingresso no *pool*



Fonte: Elaborado pelo autor

4.7. EXECUÇÃO DA ORDEM E ATUALIZAÇÃO DA CARGA DE CADA MÁQUINA

Uma vez que a ordem é liberada, ela passa pelas máquinas segundo seu roteiro até sair do chão de fábrica. Cada máquina possui sua própria fila de espera. A Figura 34 detalha os passos executados dentro dessa fila de espera de uma dada máquina. Como exemplo, tomou-se a máquina 1. O primeiro passo é declarar que a ordem completa mais uma etapa da sequência, adicionando um valor unitário na etiqueta temporal de Etapa (Linhas 6 - 8). Em seguida, a fila consulta se a ordem está na primeira etapa/operação do seu roteiro. Essa consulta é importante pois a atualização das cargas só deverá ser realizada na primeira operação de cada ordem. Logo, caso a ordem esteja na primeira operação, são criados dois vetores para armazenamento dos seis tempos de processamento e dos valores das seis operações da ordem (linha 18 – 30). Após isso, é criado um laço “for”, que irá registrar na tabela geral de cargas todos os tempos de processamento de uma determinada ordem nas respectivas máquinas que essa ordem passará, segundo o roteiro de fabricação, somando com

os tempos de processamento já existentes. (Linhas 34 – 40). Nesse contexto, o tempo de processamento da ordem referente à máquina da fila na qual ela se encontra é chamado de carga direta. Já os demais tempos de processamento da ordem referentes às outras máquinas, são chamado de carga indireta. Pois mesmo a ordem não estando diretamente ligada à máquina que está no seu roteiro de fabricação, o seu tempo de processamento já é acrescentando à essa máquina para a avaliação das cargas. A Figura 34 apresenta a fila 1 como modelo.

Caso a ordem que chegar na fila não esteja na primeira etapa, ou seja, não seja a primeira operação, (Linha 9), nada é adicionado na tabela geral de cargas e a ordem segue o seu roteiro programado. As outras filas das outras máquinas se seguem o mesmo padrão.

Figura 34 – Configuração da fila referente a cada máquina

```

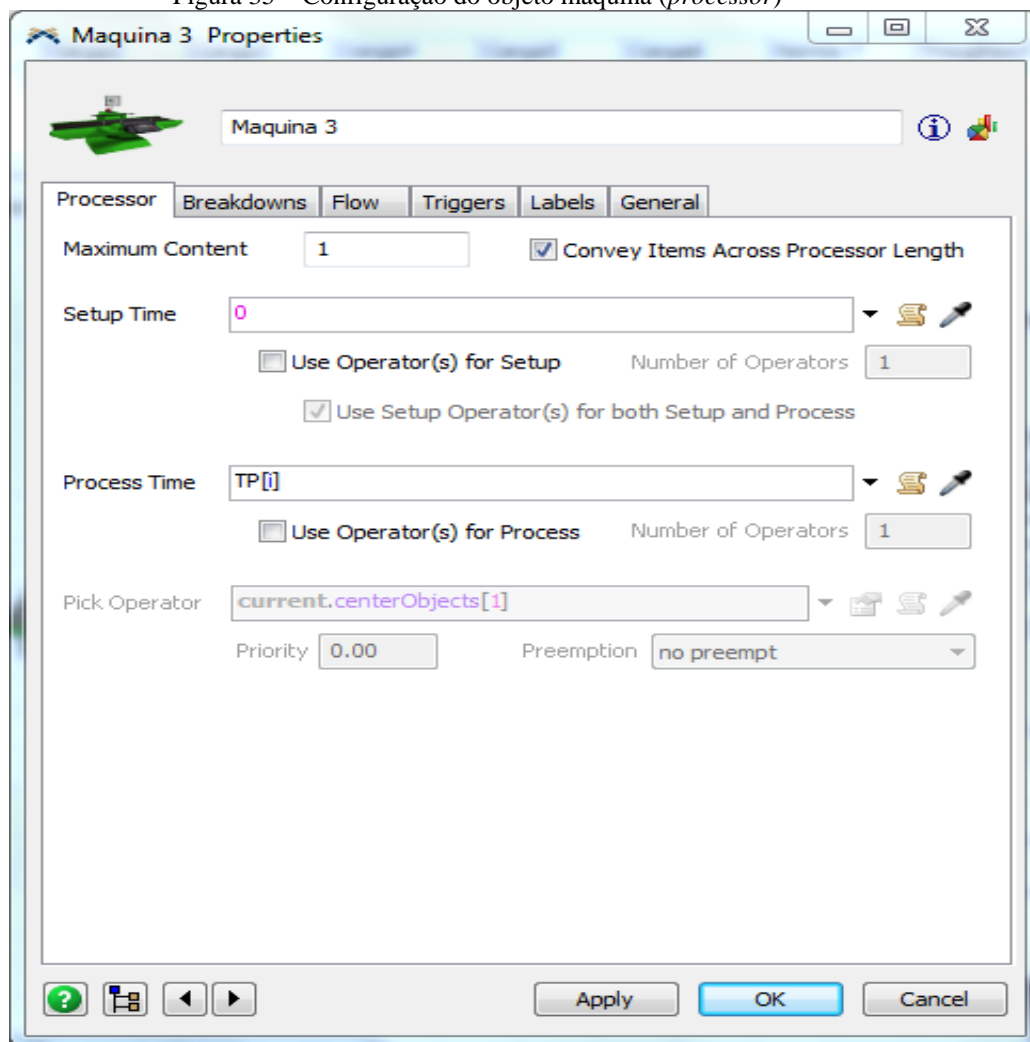
1 /**Custom Code*/
2 Object item = param(1);
3 Object current = ownerobject(c);
4 int port = param(2);
5
6 setlabel(item, "Etapa", getlabel(item, "Etapa")+1);
7 // Na entrada da fila, soma o label "Etapa" do item em 1.
8
9 if (getlabel(item, "Etapa")==1)
10 {
11
12     doublearray TP = makearray(6);
13     intarray OP = makearray(6);
14     //for (int j=1;j++;j<7)
15     //{ OP[j]=0;}
16
17
18     TP[1] = getlabel(item, "TP1");
19     TP[2] = getlabel(item, "TP2");
20     TP[3] = getlabel(item, "TP3");
21     TP[4] = getlabel(item, "TP4");
22     TP[5] = getlabel(item, "TP5");
23     TP[6] = getlabel(item, "TP6");
24
25     OP[1] = getlabel(item, "OP1");
26     OP[2] = getlabel(item, "OP2");
27     OP[3] = getlabel(item, "OP3");
28     OP[4] = getlabel(item, "OP4");
29     OP[5] = getlabel(item, "OP5");
30     OP[6] = getlabel(item, "OP6");
31
32     int w = 1 ; //Define var pivote desde 1
33
34     for (w=1;w<=getlabel(item, "numeroOperacoes");w++)
35     //ex: numeroOperacoes = 6, i<7.
36
37     {
38         settablenum("Cargas", 1, OP[w], (gettablenum("Cargas", 1, OP[w]))+TP[w]);
39         //escreve na tabela cargas o valor correspondente de Q1
40         //Faz uma função de incremento com o valor já existente em cargas
41     }
42 }

```

Fonte: Elaborado pelo autor

Cada máquina é representada por um *processor*. O tempo de processamento de cada ordem de trabalho é obtido consultando-se a etiqueta TP da ordem, segundo a etapa que esteja sendo processada nessa máquina, como apresenta a Figura 35.

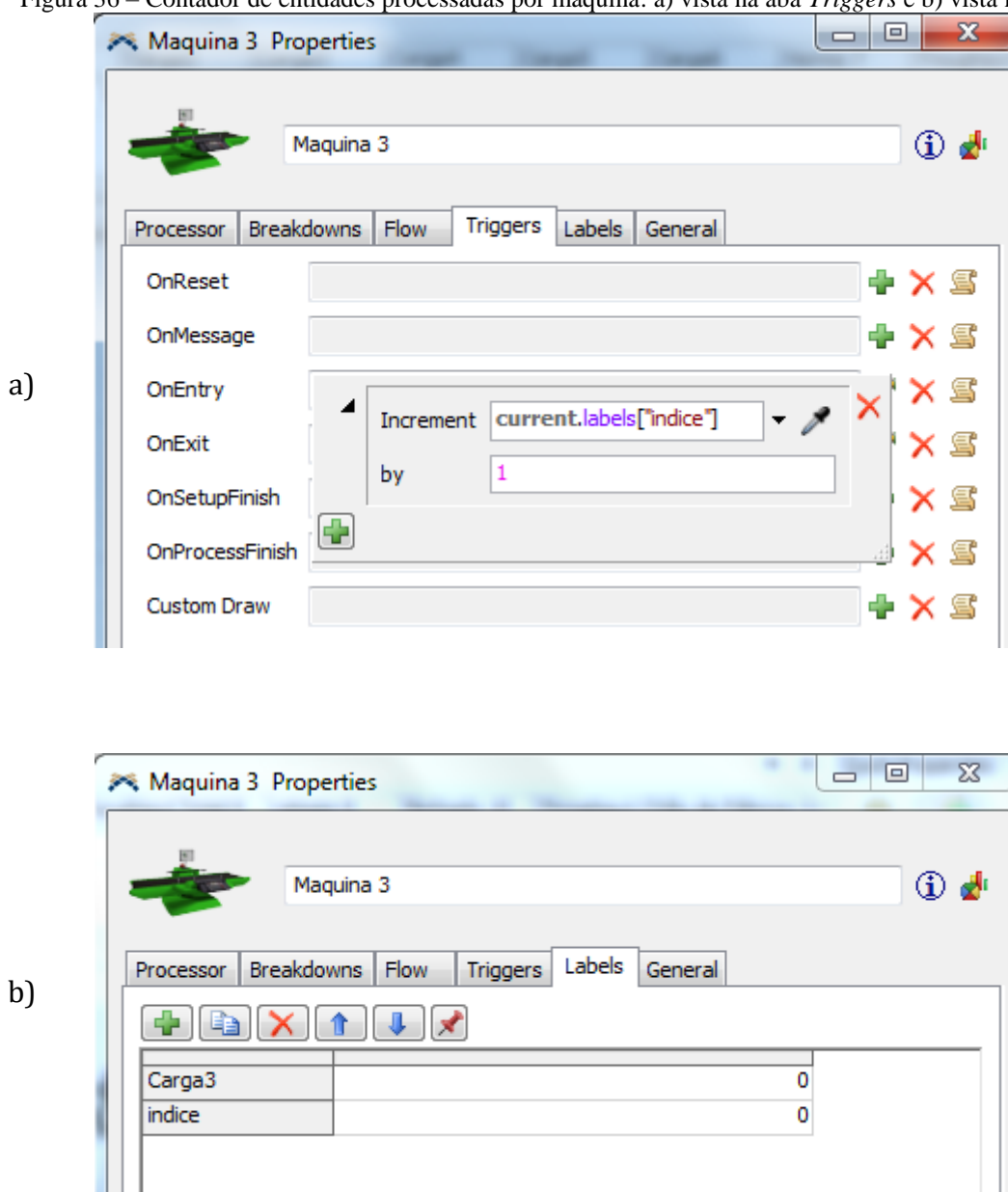
Figura 35 – Configuração do objeto máquina (*processor*)



Fonte: Elaborado pelo autor

Na entrada da ordem na máquina, incrementa-se em uma unidade a etiqueta “índice” (Figura 36a). Essa etiqueta indica a quantidade de ordens processadas pela máquina. A etiqueta “carga3” (Figura 36b) serve como variável temporária para a máquina 3 armazenar a carga de trabalho.

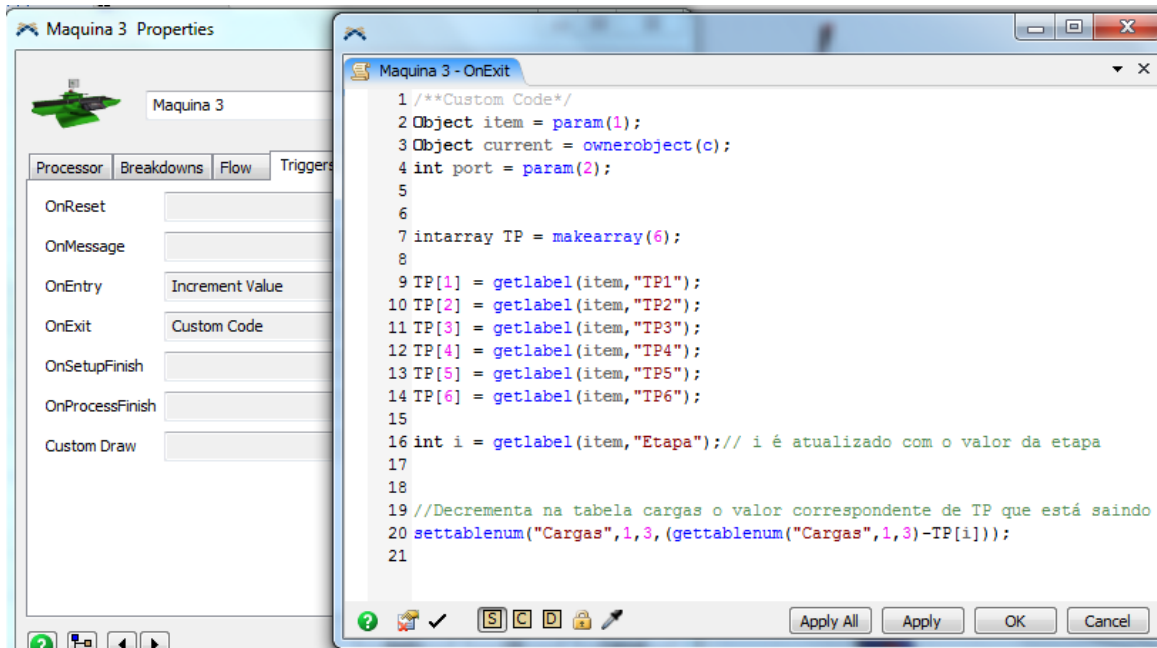
Figura 36 – Contador de entidades processadas por máquina: a) vista na aba *Triggers* e b) vista na aba *Labels*



Fonte: Elaborado pelo autor

O último passo da máquina é retirar a carga da ordem já processada da tabela de carga de trabalho. Esse passo é realizado na saída da máquina (*OnExit*), como indica a Figura 37. A máquina faz uma consulta dos tempos de processamento da ordem recém processada. Isso é feito consultando o valor das etiquetas da ordem de trabalho (Linhas 7 - 14). Realiza-se também uma leitura da etapa atual da ordem (Linha 16). Com esses dados, a máquina diminui a carga atual, armazenada na tabela geral de carga, com o valor do tempo de processamento que correspondeu a essa etapa (Linha 19 - 20). Depois, a ordem segue seu roteiro estabelecido.

Figura 37 – Subtração da carga já processada pela máquina



Fonte: Elaborado pelo autor.

4.8. FINALIZAÇÃO DAS ORDENS E COLETA DE INDICADORES

Quando a ordem termina seu roteiro, ela segue para o *sink* “Saida” para finalizar seu processo dentro da simulação. No momento que uma ordem ingressa no *sink*, armazena-se o valor atual do tempo e simulação para a etiqueta “TempoSaidaTotal”, em cada entidade (Figura 38, Linhas 5 - 13). O objetivo é salvar o tempo de finalização total da ordem dentro das etiquetas do item “*box*”. Depois, o *sink* calcula o tempo total de atravessamento da ordem e armazena o valor na coluna 8 da Tabela Cargas, como apresentado na Figura 38 (Linha 15), transfere esse valor dentro do item “*box*” (Linhas 16), e transfere o valor nas próprias etiquetas do *sink* (Linhas 17 – 24). O *sink* armazena o último valor do tempo de fluxo no chão de fábrica (Linhas 25 – 33), e atualiza o mesmo valor na coluna 11 da Tabela Cargas (Linha 36). As colunas 8 e 11 da Tabela de Cargas permitem verificar os valores do *throughput* total e do *throughput* do chão de fábrica no instante em que ocorre a simulação. As etiquetas do *sink* são usadas como indicadores de desempenho relevantes na teoria de controle de carga para verificação do comportamento do modelo, confrontando com os resultados obtidos na literatura.

Figura 38 – Configuração do *sink* saída para suas próprias etiquetas

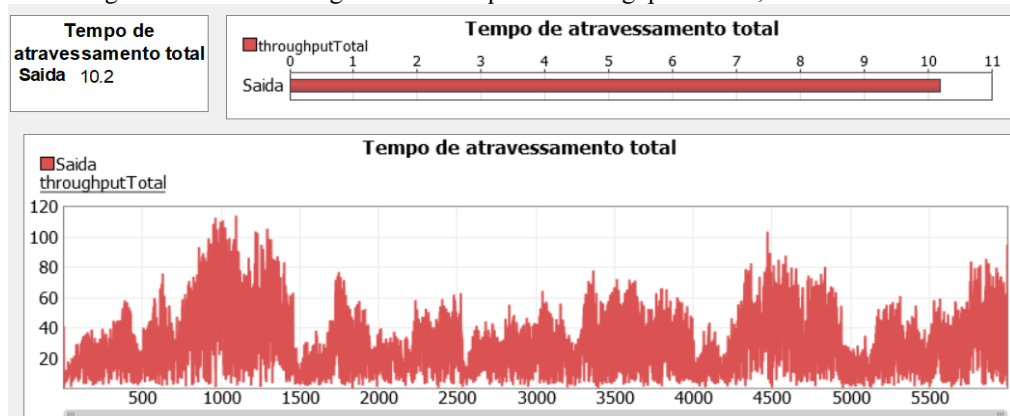
```

Saida - OnEntry
1 Object item = param(1);
2 Object current = ownerobject(c);
3 int port = param(2);
4
5 { // ***** PickOption Start ***** //
6 /**popup:SetLabel:hasitem=0*/
7 /**Set Label*/
8 Object involved = /** \nObject: *//****tag:object*//***/item/**/;
9 string labelname = /** \nLabel: *//****tag:label*//***/"tempoSaidaTotal"/**/;
10 Variant value = /** \nValue: *//****tag:value*//***/time()/**/;
11
12 involved.labels.assert(labelname).value = value;
13 } // ***** PickOption End ***** //
14
15 settablenum("Cargas",1,8,(getlabel(item,"tempoSaidaTotal")-(getlabel(item,"tempoChegada"))));
16 setlabel(item,"throughputTotal",gettablenum("Cargas",1,8));{ // ***** PickOption Start
17 /**popup:SetLabel:hasitem=0*/
18 /**Set Label*/
19 Object involved = /** \nObject: *//****tag:object*//***/current/**/;
20 string labelname = /** \nLabel: *//****tag:label*//***/"throughputTotal"/**/;
21 Variant value = /** \nValue: *//****tag:value*//***/gettablenum("Cargas",1,8)/**/;
22
23 involved.labels.assert(labelname).value = value;
24 } // ***** PickOption End ***** //
25 { // ***** PickOption Start ***** //
26 /**popup:SetLabel:hasitem=0*/
27 /**Set Label*/
28 Object involved = /** \nObject: *//****tag:object*//***/current/**/;
29 string labelname = /** \nLabel: *//****tag:label*//***/"throughputChao"/**/;
30 Variant value = /** \nValue: *//****tag:value*//***/gettablenum("Cargas",1,11)/**/;
31
32 involved.labels.assert(labelname).value = value;
33 } // ***** PickOption End ***** //
34
35
36 settablenum("Cargas",1,11,(getlabel(item,"tempoSaidaTotal")-(getlabel(item,"saidaPool"))));
37

```

Fonte: Elaborado pelo autor

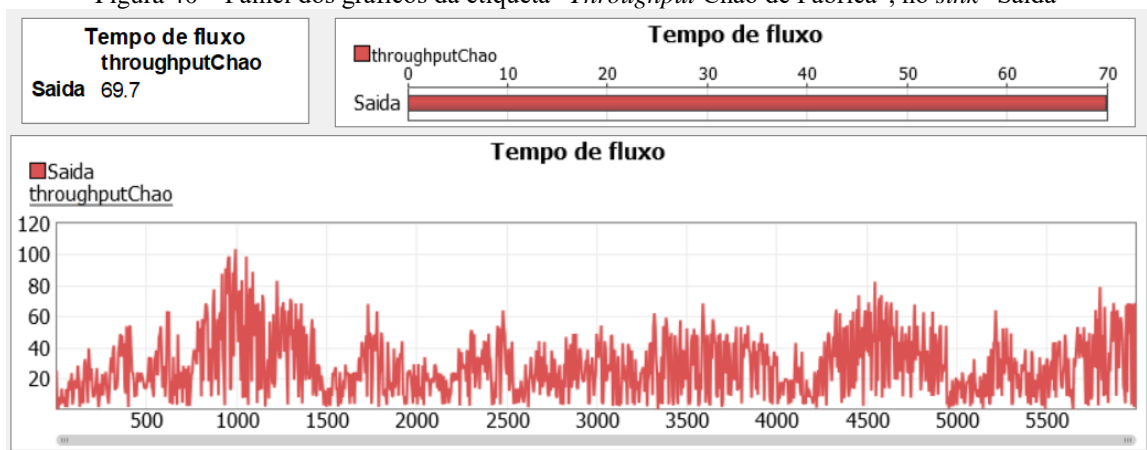
Das etiquetas do *sink* “saida”, montou-se duas séries de gráficos para verificar o comportamento do modelo. A Figura 39 apresenta os gráficos da etiqueta de Tempo de atravessamento total.

Figura 39 – Painel dos gráficos da etiqueta “Throughput Total”, no *Sink* “Saida”

Fonte: Elaborado pelo autor

Figura 40 apresenta o gráfico do Tempo de fluxo (tempo de atravessamento do chão de fábrica). As duas figuras apresentam os dados que são armazenados no *sink* para utilizá-los na verificação e nos resultados.

Figura 40 – Painel dos gráficos da etiqueta “*Throughput Chão de Fábrica*”, no *sink* “Saida”



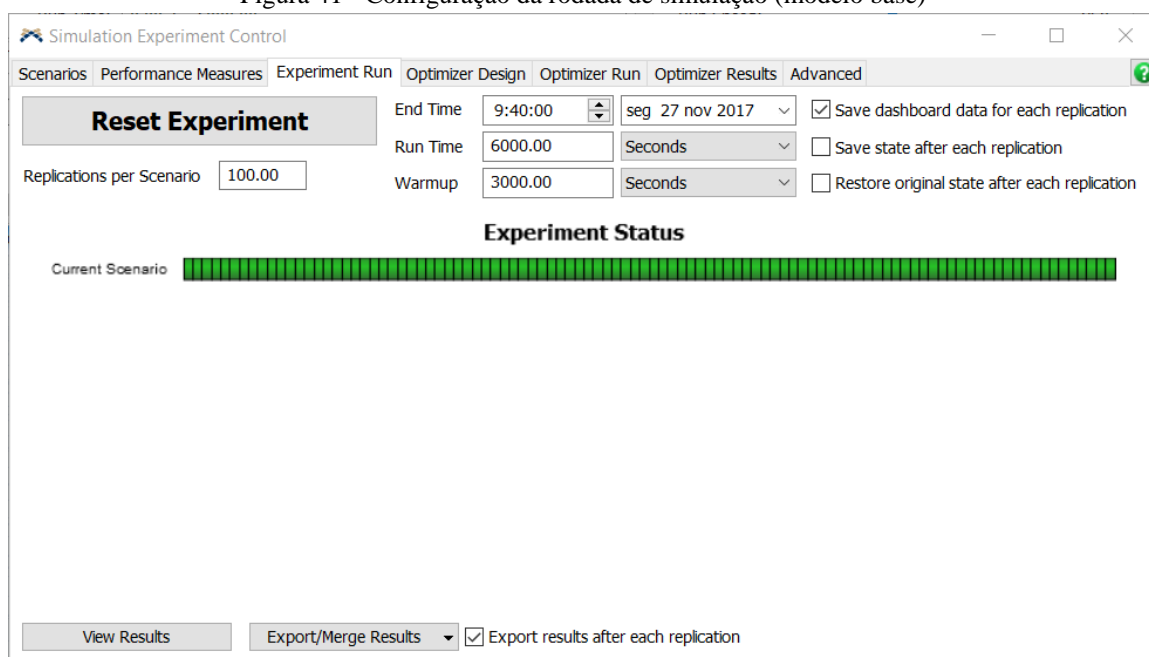
Fonte: Elaborado pelo autor

O valor exato e a barra horizontal foram utilizados para a verificação do modelo. Já o gráfico da série de tempo, foi coletado para comparar a simulação em relação ao comportamento do modelo de Land (2006). O *software* tem um próprio mecanismo para calcular a porcentagem de utilização das máquinas, segundo os estados que elas representem.

5 RESULTADOS

Os resultados do modelo foram analisados no estado permanente da simulação. Considerou-se estado permanente para que a projeção dos resultados obtidos com o modelo simulado fosse consistente para representar uma decisão que permaneça ao longo do tempo de execução. Como se apresenta no modelo de Land (2006), realizaram-se 100 replicações com o valor da norma relativamente alto, a ponto de não obstruir a liberação das ordens e ser considerada como uma norma infinita. O tempo de simulação de 6.000 unidades de tempo (definidos na modelagem em segundos) e um tempo para o sistema para o sistema se estabilizar (*warming-up*) de 3.000 unidades de tempo (Figura 41). A simulação foi executada em um computador pessoal, com um processador Intel i7, com 6GB de RAM, em um sistema operacional MS® Windows 10.

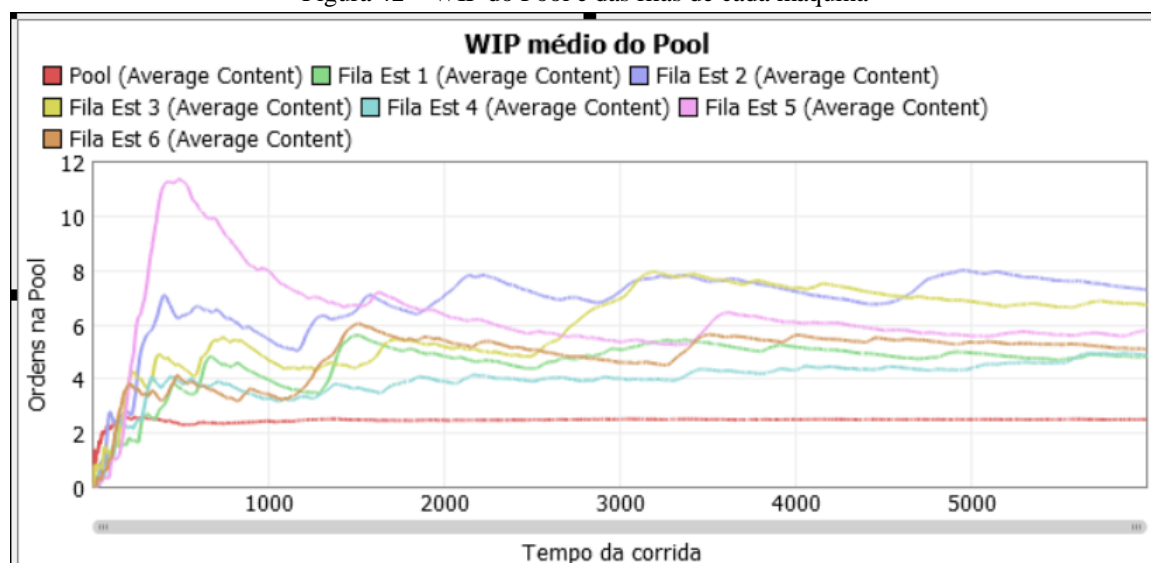
Figura 41 - Configuração da rodada de simulação (modelo base)



Fonte: Elaborado pelo autor

Para verificar o tempo *warming-up*, utilizou-se o método gráfico. Para todas as análises o tempo em questão é dado em segundos. A Figura 42 apresenta como a quantidade média de ordens fica mais estável a partir de 3.000 unidades de tempo de simulação.

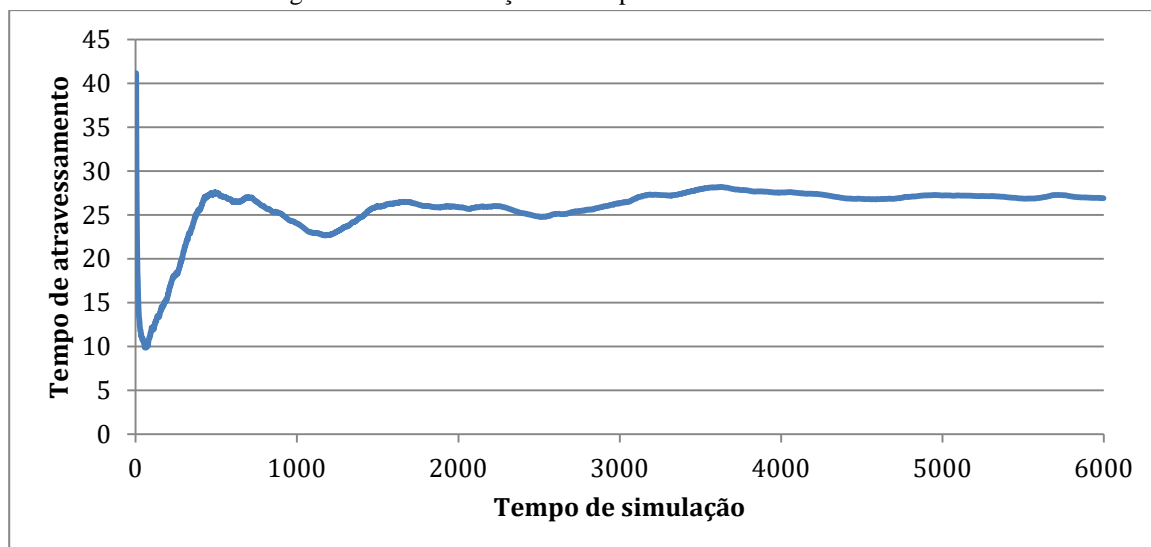
Figura 42 – WIP do Pool e das filas de cada máquina



Fonte: Elaborado pelo autor

A Figura 43 apresenta a estabilização do tempo de atravessamento total no decorrer da simulação. É possível notar no gráfico que a partir do tempo de *warming up*, o tempo de atravessamento total se mantém mais estável, com valores próximos entre si.

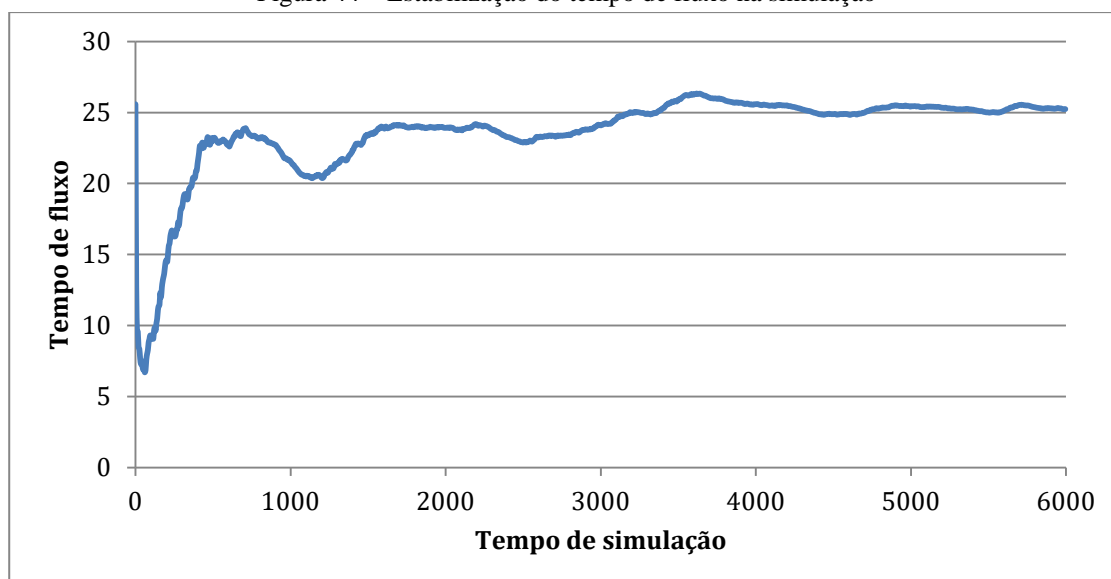
Figura 43 – Estabilização do tempo de atravessamento total



Fonte: Elaborado pelo autor

A Figura 44 refere-se ao gráfico: Estabilização do tempo de fluxo (tempo de atravessamento do chão de fábrica) na simulação, no qual demonstra que o tempo de *warming-up* adotado é suficiente para o sistema ficar em estado permanente.

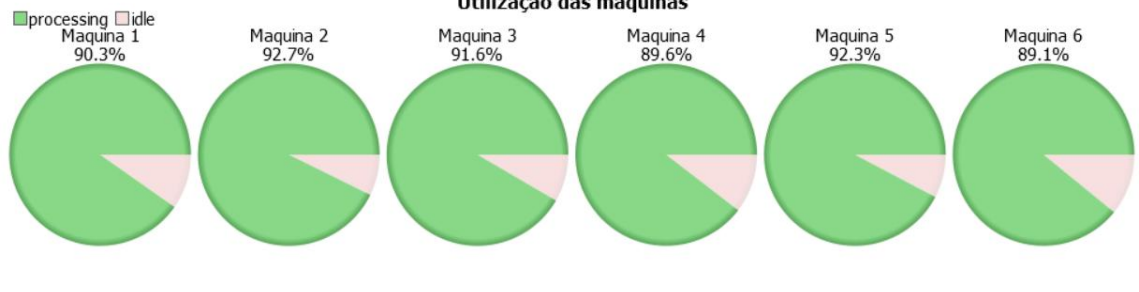
Figura 44 – Estabilização do tempo de fluxo na simulação



Fonte: Elaborado pelo autor

Para analisar o modelo base, utilizaram-se as seguintes variáveis, por ordem de relevância: a) Tempo de atravessamento total da ordem; b) Tempo de fluxo (tempo de atravessamento do chão de fábrica); c) Utilização das máquinas. O modelo base é aquele que utiliza um valor de norma infinita, ou seja, não restringe a liberação das ordens, que são enviadas ao chão de fábrica assim que chegam. Segundo o modelo de Land (2006), a taxa de utilização das máquinas encontrada deve ser de aproximadamente 90%. A Figura 45 demonstra que nas seis máquinas, o modelo construído atinge valores próximos a taxa determinada na literatura.

Figura 45 – Utilização das máquinas no modelo

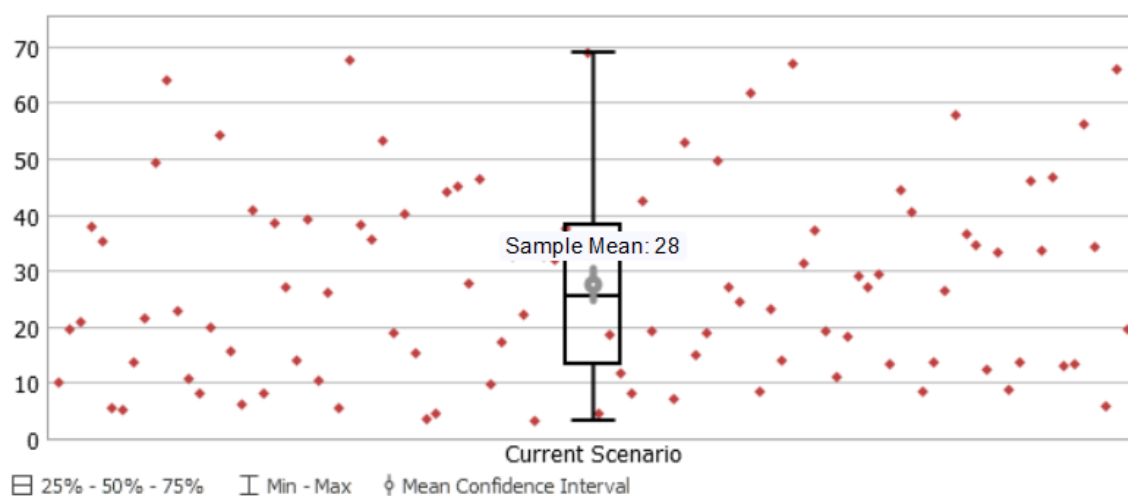


Fonte: Elaborado pelo autor

A partir dos resultados obtidos com 100 replicações, demonstrados no gráfico da Figura 46 com o eixo “x” indicando as ordens e o eixo “y” o tempo de atravessamento,

percebe-se que a média do tempo de atravessamento total foi de 28 unidades de tempo, contra um tempo de 28,2 unidades reportado em Land (2006), considerando as mesmas condições. Para o modelo base (sem restrição de liberação por norma), observa-se que 50% dos valores ao redor da mediana (25.0 unidades de tempo) estão entre os valores de 13.0 unidades de tempo (Percentil 25) e 38.0 unidades de tempo (Percentil 75). Os percentis 25 e 75 no gráfico do *Box plot* representam o primeiro e o terceiro quartil da caixa do gráfico. A mediana representa o segundo quartil (Figura 46).

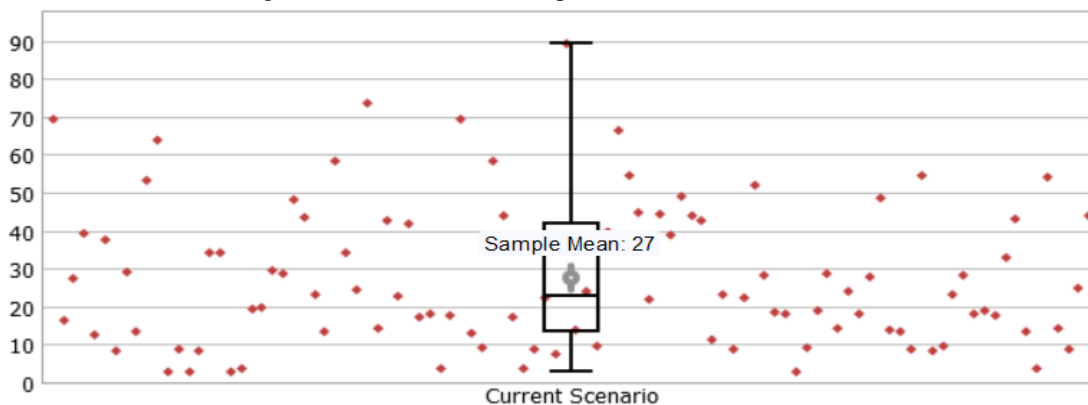
Figura 46 – Resultado do tempo de atravessamento total do modelo base



Fonte: Elaborado pelo autor

Para a média do tempo de fluxo (Figura 47), o valor obtido na simulação foi de 27.0 unidades de tempo, contra 25,7 unidades de tempo reportadas por Land (2006). O valor da mediana encontrada na simulação foi de 25 e os valores do primeiro (25%) e terceiro quartil (75%), que limitam a caixa do gráfico *box plot* foram de 14.0 e 42.0 unidades de tempo, respectivamente.

Figura 47 – Resultado do tempo de fluxo do modelo base



Fonte: Elaborado pelo autor

A Tabela 3 faz referência aos resultados encontrados na literatura em comparação ao obtidos por esse modelo.

Tabela 3 - Tempo de atravessamento da literatura versus autor

DESCRIÇÃO	LITERATURA (LAND, 2006)	ENCONTRADO
Média do tempo de fluxo do chão de fábrica (Norma infinita)	25,7	27
Média do tempo de atravessamento total do chão de fábrica (Norma infinita)	28,2	28

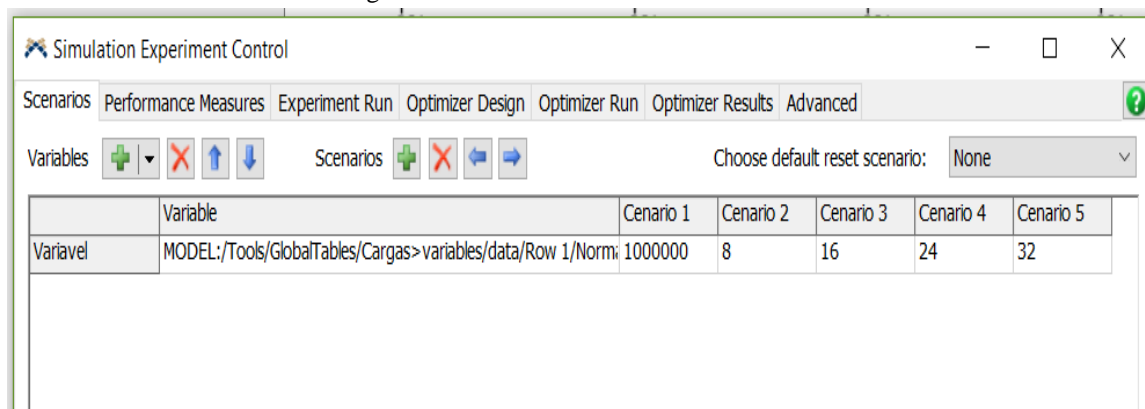
Fonte: Elaborado pelo autor

Uma vez que o modelo base simulado, após entrar em regime permanente, apresentou resultados de taxa de utilização de máquinas, tempo de fluxo e média de tempo de atravessamento total próximos ao apresentado na literatura por Land (2006), é possível fazer uso desse modelo para simular diversos cenários diferentes, ajustando os parâmetros do modelo de acordo as necessidades desejadas, com o intuito de obter melhores resultados do sistema modelado.

Como exemplo de parâmetro que pode ser ajustado está a norma. Land (2006) demonstrou que ajustando os valores da norma é possível obter resultados na diminuição dos tempos de atravessamento total e no tempo de atravessamento de chão de fábrica.

Este trabalho criou cenários com diferentes valores da norma para ser possível analisar quais impactos os diferentes valores da norma causam no modelo. Sendo assim, foram gerados quatro cenários com diferentes valores de normas: i) 8; ii) 16; iii) 24; iv) 32; (Figura 48).

Figura 48 – Cenários com diferentes normas



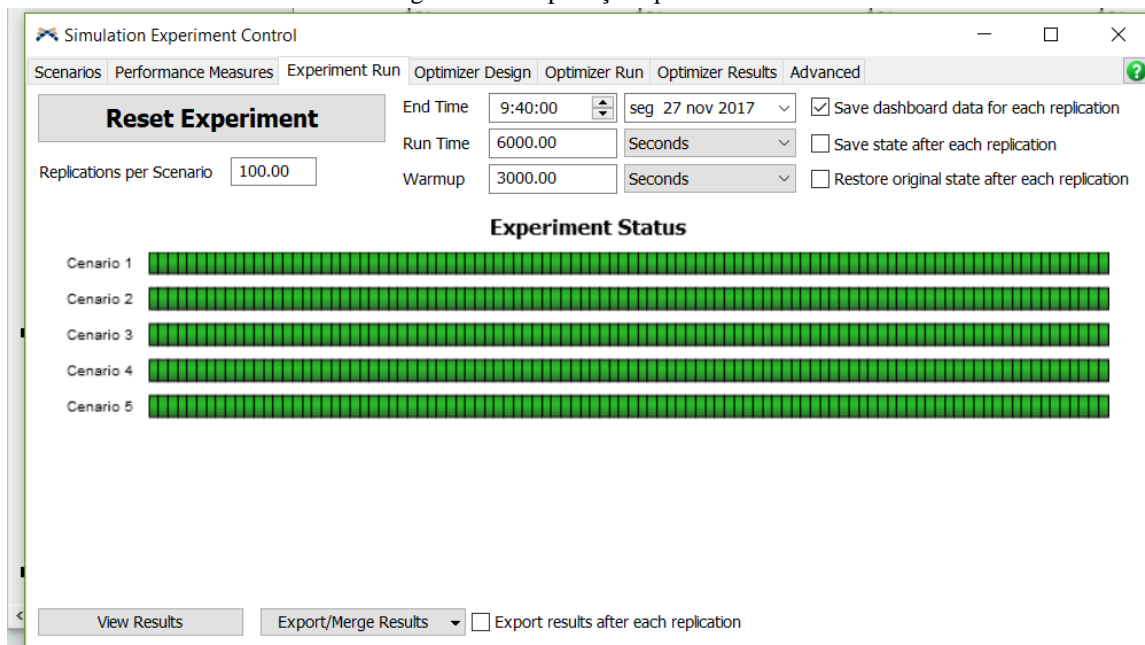
The screenshot shows the 'Simulation Experiment Control' window. It has several tabs: 'Scenarios', 'Performance Measures', 'Experiment Run', 'Optimizer Design', 'Optimizer Run', 'Optimizer Results', and 'Advanced'. The 'Scenarios' tab is active. Below the tabs, there are controls for 'Variables' and 'Scenarios', including a 'Choose default reset scenario:' dropdown set to 'None'. A table is displayed with the following data:

Variable	Cenário 1	Cenário 2	Cenário 3	Cenário 4	Cenário 5
MODEL:/Tools/GlobalTables/Cargas>variables/data/Row 1/Norma	1000000	8	16	24	32

Fonte: Elaborado pelo autor

Esses cenários foram replicados 100 vezes cada um (Figura 49) e comparados com os resultados encontrados dos tempos de atravessamento com os resultados do cenário de norma infinita.

Figura 49 – Replicações por cenário

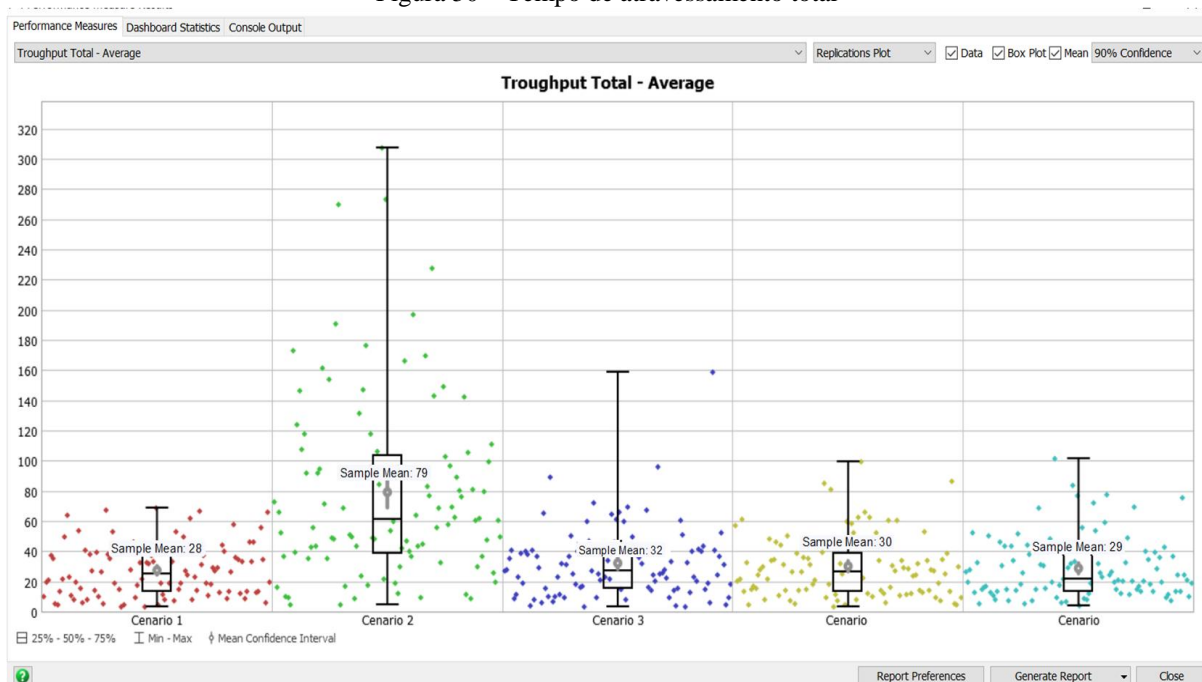


Fonte: Elaborado pelo autor

Nesses quatro cenários criados, foram considerados o modelo clássico do WLC, ou seja: a ordem sempre é classificada pela data de liberação planejada, não há prioridade caso ela esteja retornando do *loop* (lista de ordens não liberadas), e só é liberada ao chão de fábrica se a soma da carga das máquinas com o tempo de processamento da ordem, não ultrapassar a norma estipulada.

Quanto ao tempo de atravessamento total (Figura 50), quando a norma é muito estreita, neste caso, no segundo cenário (norma igual a 8), a média desse tempo de atravessamento é muito longa (média igual a 79 unidades de tempo) em relação ao primeiro cenário (média igual a 27 unidades de tempo) com norma infinita. Já para os cenários 3,4 e 5 obteve-se uma média de tempo de atravessamento total igual a 32, 30 e 29 unidades de tempo, respectivamente.

Figura 50 – Tempo de atravessamento total

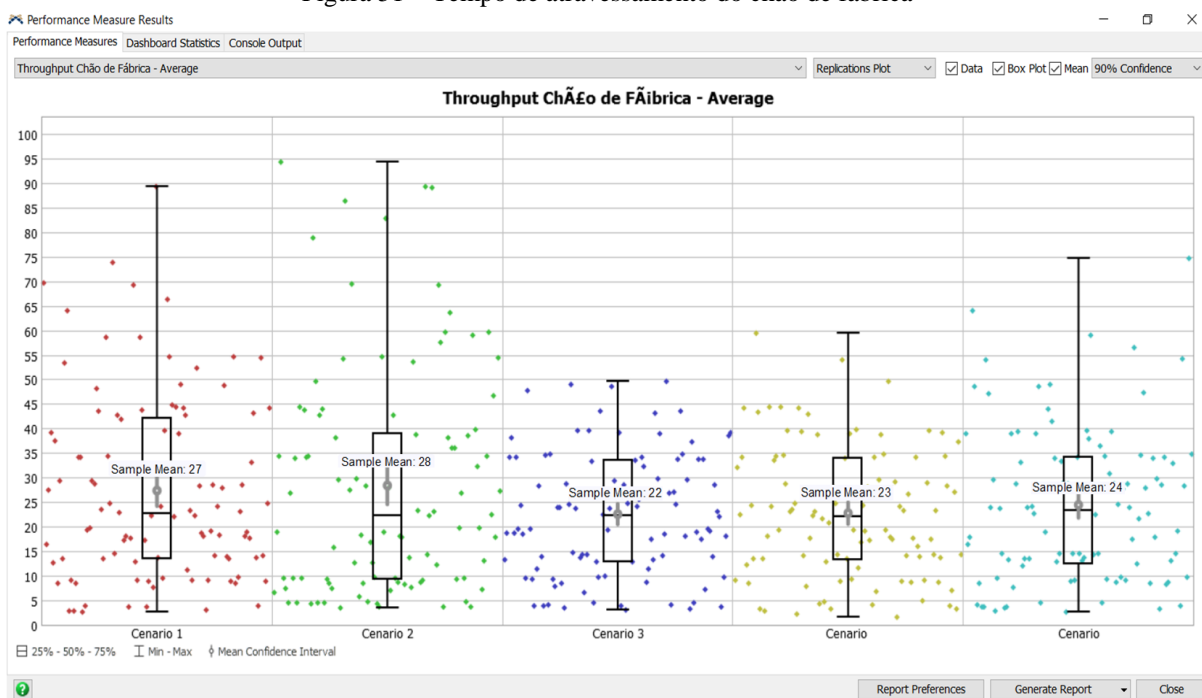


Fonte: Elaborado pelo autor

Quanto ao tempo de atravessamento de chão de fábrica (Figura 51), o segundo cenário apresentou uma média de 28 unidades de tempo, registrando um aumento de 1 unidade de tempo em relação ao primeiro cenário (27 unidades de tempo). Já os cenários 3, 4 e 5, apresentam valores de média do tempo de atravessamento de chão fábrica menores do que o primeiro cenário, que possui norma infinita. Foram obtidas médias de 22, 23 e 24 unidades de tempo, respectivamente.

Os resultados das médias dos tempos de atravessamento do chão de fábrica encontradas nos cenários 3, 4 e 5, em relação ao primeiro e ao segundo cenário, apresentam diferenças de valores de médias que podem ser representativas para uma situação real. Ao comparar o cenário 3 com o cenário 1, por exemplo, a diferença do valor da média para cada ordem é de 5 unidades de tempo, considerando que foram simuladas 100 ordens, implica concluir que o cenário 3 processou essas 100 ordens em 500 unidades de tempo a menos que o cenário 1. O resultado dessa comparação leva a concluir que o chão de fábrica do cenário 3 é mais produtivo que o cenário 1, pois consegue processar uma mesma quantidade de ordens em menor tempo.

Figura 51 – Tempo de atravessamento do chão de fábrica



Fonte: Elaborado pelo autor

Outra característica importante a ser analisada é a variabilidade dos tempos de atravessamento do chão de fábrica. É possível observar alta variabilidade nos valores de tempo de atravessamento do chão de fábrica para os cenários 1, 2 e 5 e menor variabilidade para o cenário 3, o que presume ser um cenário mais estável do que os demais (Figura 51).

A criação desses cenários contribui para compreender e analisar os resultados gerados pela estrutura de conversão do controle de carga construída. Neste trabalho variou-se a norma, que ao adquirir diferentes valores, impactou diretamente na variação do tempo de atravessamento de uma ordem.

A variação da norma em um ambiente de simulação de eventos discretos para a liberação das ordens permite analisar os efeitos que essa variação causa nos tempos de atravessamento total e de chão de fábrica. É possível também analisar para qual o valor de norma obtém-se menor variabilidade nesses tempos calculados, permitindo a escolha de um cenário mais estável.

6 CONSIDERAÇÕES FINAIS

Este trabalho apresentou uma aplicação da abordagem do controle de carga (WLC) em um *software* comercial de simulação de eventos discretos. Na literatura, os modelos computacionais utilizados para simular esta abordagem são em geral dedicados, e pouca ênfase se dá à implementação do modelo em si. Assim, detalhou-se essa conversão do algoritmo básico do controle de carga com liberação periódica em um programa de simulação desenvolvido em *Flexsim*.

Este trabalho mostrou que a implementação do algoritmo do WLC em um *software* comercial de simulação de eventos discretos é possível e viável. Entretanto, essa conversão, passando-se de um programa dedicado e de linguagem estruturada, para um ambiente de modelagem que utiliza objetos e programação não centralizada, não é trivial. Os comandos e as sub-rotinas que implementam as linhas do algoritmo base do controle de carga ficam distribuídos em diferentes objetos e é necessário criar mecanismos de coordenação entre eles, para que sejam executados na ordem correta e nos momentos corretos. Exemplos desses mecanismos de interação e integração entre os objetos são o uso de etiquetas (*labels*) e tabelas, que são lidos em diferentes momentos da simulação, por diferentes objetos, para captar informações atualizadas para a execução de sub-rotinas ou decisões sobre o fluxo das ordens.

Para o funcionamento correto da abordagem do controle de carga no sistema modelado, é necessário que sejam implementados tanto mecanismos que controlam a periodicidade da abertura e fechamento do *pool* como mecanismos que representam a lista de ordens não liberadas. Esses mecanismos, quando se escreve um programa dedicado, exigem poucas linhas de código em comparação a um *software* de simulação de eventos discretos, que além de linhas de códigos, requer a criação de conexões entre os objetos, por meio de uma programação não centralizada.

Embora não seja trivial a implementação do WLC no *software* de simulação de eventos discretos, como exposto acima, é importante destacar algumas vantagens que essa implementação proporciona, como: a capacidade descritiva que esse tipo de *software* possui, possibilitando a modelagem e simulação de um ambiente fabril bem próximo ao real; a possibilidade de visualização da dinâmica da simulação e do funcionamento da abordagem do WLC no ambiente modelado, uma vez que é possível visualizar o caminho que as ordens percorrem desde a criação até a saída do sistema; e a possibilidade de clicar nos itens e objetos

e visualizar as informações contidas neles no exato momento da simulação e as alterações que esses sofrem ao passar do tempo de simulação.

Como contribuições para pesquisas futuras, o modelo pode servir de base para representar a implantação do WLC em outras configurações de chão de fábrica, contribuindo para diminuir o esforço gasto pelo desenvolvedor na modelagem e programação dessa estrutura, uma vez que permite ao desenvolvedor partir de um modelo inicial bem estruturado para a criação e modelagem de novos cenários fabris, que utilizam a abordagem do WLC.

Entre as limitações deste trabalho, está a utilização de somente uma pesquisa como referência da literatura, o estudo de Land (2006). Embora este seja um modelo muito citado, expandir a comparação dessa estrutura para outros autores geraria uma análise mais aprofundada, possibilitando encontrar resultados diferentes, o que fica de recomendação para pesquisas futuras.

O uso de somente um *software* de simulação de eventos discretos, nesse caso o *FlexSim*, é outra limitação desta pesquisa, já que existem outros *softwares* similares (*Arena*, *ProModel*, etc.) que requerem outras estratégias e recursos para se gerar um modelo adequado à representação do WLC.

A descrição da modelagem apresentada contribui com o ambiente empresarial e acadêmico, pois facilita a replicação e aplicação dessa estrutura em outros ambientes fabris, permitindo reduzir o tempo do projeto para desenvolver o modelo base, permitindo focar diretamente nas comparações e adequações ao ambiente avaliado, por parte de gestores e/ou pesquisadores.

REFERÊNCIAS

- AKILLIOGLU, H.; DIAS-FERREIRA, J.; ONORI, M. Characterization of continuous precise workload control and analysis of idleness penalty. **Computers & Industrial Engineering**, v. 102, p. 351-358, 2016.
- ALVES, R. *Filosofia da ciência: introdução ao jogo e suas regras*. 21. ed. São Paulo: Brasiliense, 1995.
- BATTAGLIA, D. *et al.* Controle pela carga de trabalho de um sistema flexível de manufatura. **Revista Ingeniería Industrial**, v. 15, n. 1, 2016.
- BANKS, J. **Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice**. John Wiley and Sons, 1998.
- BECHTE, W. Theory and practice of load-oriented manufacturing control. **The International Journal of Production Research**, v. 26, n. 3, p. 375-395, 1988.
- BECHTE, W. Load-oriented manufacturing control just-in-time production for *jobshops*. **Production Planning & Control**, v. 5, n. 3, p. 292-307, 1994.
- BERTRAND, J. W. M.; WORTMANN, J. C. **Production control and information systems for component-manufacturing shops**. Amsterdam: Elsevier Scientific Publishing Company, 1981.
- BRYMAN, A. *Research methods and organization studies*. Londres: Unwin Hyman, 1989.
- CARVALHO, M. C.M. A construção do saber científico: algumas proposições. In: CARVALHO, M. C.M. (Org.). *Construindo o saber*. 2. ed. Campinas: Papirus, 2000.
p. 63-86.
- CHALMERS, A. F. *O que é ciência, afinal?* São Paulo: Brasiliense, 1995.
- FERNANDES, N. O. Contribuições para o controlo da actividade de produção no sector de produção por encomenda. **Contribuições para o controlo da actividade de produção no sector de produção por encomenda**, 2007.
- FERNANDES, N. O.; CARMO-SILVA, S. Workload control under continuous order release. **International Journal of Production Economics**, v. 131, n. 1, p. 257-262, 2011.
- FERNANDES, N. O.; CARMO-SILVA, S. Workload Control and Order Release: The Influence of the Location of Protective Capacity. In: **Proceedings of the World Congress on Engineering**. 2013.
- FERNANDES, N. O. *et al.* Improving workload control order release: incorporating a starvation avoidance trigger into continuous release. **International Journal of Production Economics**, 2016.
- GOLDRATT, E.M. **What is This Thing Called Theory of Constraints and How Should it be Implemented?**. North River Press, New York, 1990.
- GOLDRATT, E.M., COX, J. **The Goal: Excellence in Manufacturing**. North River Press, New York, 1989.
- HENDRY, L.C.; **A decision support system to manage delivery and manufacturing lead times in make-to-order companies**. Ph.D. thesis, University of Lancaster (UK), 1989.
- HENDRY, L. C.; KINGSMAN, B. G. Production planning systems and their applicability to make-to-order companies. **European Journal of Operational Research**, v. 40, n. 1, p. 1-15, 1989.
- HENDRY, L.; KINGSMAN, B. A decision support system for job release in make-to-order companies. **International Journal of Operations & Production Management**, v. 11, n. 6, p. 6-16, 1991.
- HENDRY, L. C.; KINGSMAN, B. G.; CHEUNG, P. The effect of workload control (WLC) on performance in make-to-order companies. **Journal of Operations Management**, v. 16, n. 1, p. 63-75, 1998.

- HENDRY, L.; HUANG, Y.; STEVENSON, M.. Workload control: Successful implementation taking a contingency-based view of production planning and control. **International Journal of Operations & Production Management**, v. 33, n. 1, p. 69-103, 2013.
- HENRICH, P. *et al.* The effect of information on workload control. In: **First Joint EUROMA POMS Conference Proceedings**. 2003. p. 611-620.
- JOHNSON, L.A; MONTEGOMERY, D.C. **Operations research in production planning, scheduling and inventory control**. New York: Johnson Wiley, 1974.
- KINGSMAN, B. G.; TATSIOPOULOS, I. P.; HENDRY, L. C. A structural methodology for managing manufacturing lead times in make-to-order companies. **European journal of operational research**, v. 40, n. 2, p. 196-209, 1989.
- KOMESU, A., S. **Heurística Evolutiva para minimização do atraso total em ambiente de produção flowshop com buffer zero**. Dissertação de mestrado – EESC/USP. São Carlos, p.84. 2015.
- LAND, M.; GAALMAN, G.. Workload control concepts in *jobshops* a critical assessment. **International journal of production economics**, v. 46, p. 535-548, 1996.
- LAND, M. J.; GAALMAN, G. J.C. The performance of workload control concepts in *jobshops*: Improving the release method. **International Journal of Production Economics**, v. 56, p. 347-364, 1998.
- LAND, M.. Parameters and sensitivity in workload control. **International journal of production economics**, v. 104, n. 2, p. 625-638, 2006.
- LAND, M. J. *et al.* *Jobshop* control: In search of the key to delivery improvements. **International Journal of Production Economics**, v. 168, p. 257-266, 2015.
- LIDDELL, Mike. **O pequeno livro azul da programação da produção**. 2. ed. Vitória: Tecmaran, 2009. 127 p.
- MARTINS, R.A.; Princípios da Pesquisa Científica. In: **Metodologia de Pesquisa em Engenharia de Produção e Gestão de Operações**. Rio de Janeiro: Elsevier: ABEPRO, 2012.
- MORABITO NETO, R.; PUREZA, V.; Modelagem e Simulação. In: **Metodologia de Pesquisa em Engenharia de Produção e Gestão de Operações**. Rio de Janeiro: Elsevier: ABEPRO, 2012.
- OOSTERMAN, B.; LAND, M.; GAALMAN, G.. The influence of shop characteristics on workload control. **International journal of production economics**, v. 68, n. 1, p. 107-119, 2000.
- PLOSSL, G. W. - **Production and Inventory Control: principles and techniques**. 2nd ed. New Jersey: Prentice-hall, Englewood Cliffs, 1985.
- PÜRGSTALLER, P.; MISSBAUER, H. Rule-based vs. optimisation-based order release in workload control: A simulation study of a MTO manufacturer. **International Journal of Production Economics**, v. 140, n. 2, p. 670-680, 2012.
- PUTNAM, A. O. MRP for repetitive manufacturing shops: a flexible kanban system for America. **Production and Inventory Management**, p. 60-88, Third Quarter, 1983.
- RENNA, P. Workload Control Policies under continuous order release. **Production Engineering**, v. 9, n. 5-6, p. 655-664, 2015.
- SILVA, C.; STEVENSON, M.; THÜRER, M. A case study of the successful implementation of workload control: a practitioner-led approach. **Journal of Manufacturing Technology Management**, v. 26, n. 2, p. 280-296, 2015.
- SIPPER, D.; BULFIN JR. R. L. **Production: Planning, Control and Integration**. New York: McGraw Hill, 1997.
- SLACK, N.; CHAMBERS, S.; JOHNSTON, R. **Administração da Produção**. 3. ed. São Paulo: Atlas, 2009.
- SPEARMAN, M. L. *et al.*, CONWIP: a pull alternative to kanban. **The International Journal of Production Research**, v. 28, n. 5, p. 879-894, 1990.

- SIPPER, D.; BULFIN JR. R. L. **Production: Planning, Control and Integration**. New York: McGraw Hill, 1997.
- SOEPENBERG, G. D.; LAND, M. J.; GAALMAN, G. J.C. Adapting workload control for *jobshops* with high routing complexity. **International Journal of Production Economics**, v. 140, n. 2, p. 681-690, 2012.
- TATSIPOULOS, I. P. **A micro-computer based interactive system for managing production and marketing in small component-manufacturing firms: using a hierarchical backlog control and lead time management methodology**. 1983. Tese de Doutorado. University of Lancaster.
- THÜRER, M. *et al.* Concerning Workload Control and Order Release: The Pre-Shop Pool Sequencing Decision. **Production and Operations Management**, v. 24, n. 7, p. 1179-1192, 2015.
- THÜRER, M. *et al.* The design of simple subcontracting rules for make-to-order shops: An assessment by simulation. **European Journal of Operational Research**, v. 239, n. 3, p. 854-864, 2014^a
- THÜRER, M. *et al.* Lean control for make-to-order companies: Integrating customer enquiry management and order release. **Production and Operations Management**, v. 23, n. 3, p. 463-476, 2014b.
- THÜRER, M.; SILVA, C.; STEVENSON, M. Optimising workload norms: the influence of shop floor characteristics on setting workload norms for the workload control concept. **International Journal of Production Research**, v. 49, n. 4, p. 1151-1171, 2011.
- THÜRER, M.; GODINHO FILHO, M. Redução do lead time e entregas no prazo em pequenas e médias empresas que fabricam sob encomenda: a abordagem Workload Control (WLC) para o Planejamento e Controle da Produção (PCP). **Gestão e Produção [online]**, v. 19, n. 1, p. 43-58, 2012.
- THÜRER, M. *et al.* Workload Control and Order Release: A Lean Solution for Make-to-Order Companies. **Production and Operations Management**, v. 21, n. 5, p. 939-953, 2012.
- THÜRER, M. *et al.* Drum-buffer-rope and workload control in High-variety flow and *jobshops* with bottlenecks: An assessment by simulation. **International Journal of Production Economics**, v. 188, p. 116-127, 2017.
- THÜRER, M.; STEVENSON, M.; LAND, M. J. On the integration of input and output control: Workload Control order release. **International Journal of Production Economics**, v. 174, p. 43-53, 2016.
- TUBINO, D. F. **Planejamento e controle da produção: teoria e prática** . Editora Atlas SA, 2009.
- WIENDAHL, H. P. *et al.* **Load-oriented manufacturing control**. Berlin etc.: Springer, 1995.

APÊNDICE A.1: Algoritmo básico de liberação de ordens do WLC em um ambiente de programação simples, com linguagem estruturada e de alto nível (parte 1).

```

1  % Programa que roda todas as funções do preenchimento da tabela
2  %Sequencia: ID, Intervalos de chegadas, Tempo de atravessamento de cada
3  %estação, Soma do tempo de cada estação, TDs que é a soma de cada estação +
4  %o intervalo de chegadas, data de entrega, tempo de liberação planejada
5  %(data de entrega - TDs)
6
7  L=10; %Linhas
8  c=6;  % Colunas
9  min=1; %Quantidade mínima de máquinas
10 max=6; %Quantidade máxima de máquinas
11
12 %Função de ordens de 1 a L. Gera os Id's que serão usados adiante na
13 %Psj2
14 for i=1:L
15     vt_ordens(i,1)=i;
16 end
17
18 % Função que gera os intervalos de chegada acumulados pela dist.
19 % uniforme(0,1) para cada ordem
20
21 inter_chegada=zeros;
22 for i=1:L
23     valor= unifrnd(0,1);
24     if i==1
25         inter_chegada(i)=valor;
26     else
27         inter_chegada(i)=inter_chegada(i-1)+valor;
28     end
29 end
30
31 %Função cria a quantidade de estações em cada ordem
32 qt_estacoes = randi([min max],L,1);
33
34
35 %Preenchimento da matriz com as estações que cada ordem passará baseado em
36 %uma dist randômica de 1 a 6
37 %Tem-se que implementar uma condicional para que uma estação não se
38 %repita em uma mesma ordem
39
40 roteiros=zeros(L,c);
41 for i=1:L
42     for j=1:qt_estacoes(i)
43         status=true;
44         while status
45             roteiros(i,j)=randi([1 6]);
46             status=false;
47             %Função que verifica se há uma estação igual a outra na mesma ordem
48             %Caso haja, o status recebe true e retorna ao loop para receber outro valor
49             for k=1:j-1
50                 if roteiros(i,j)==roteiros(i,k)
51                     status=true;
52                 end
53             end
54         end
55     end
56 end

```

APÊNDICE A.2: Algoritmo básico de liberação de ordens do WLC em um ambiente de programação simples, com linguagem estruturada e de alto nível (parte 2).

```

57
58 %Função que atribui o tempo de atravessamento planejado (tpa), calculado por
59 uma uniforme(5,10), para cada ordem à uma matriz Psj.
60 % No final a função calcula a soma do tpa de cada ordem: tpa_planejado
61
62 Psj=zeros(L,c); %Cria uma matriz Lxc de zeros
63 tpa_planejado(L)=zeros; % Cria um vetor de dimensão L de zeros
64 ord=0;
65 for i =1:L
66     for j=1:c
67         if roteiros(i,j)==0
68             break
69         else
70             ord=unifrnd(5,10);
71             Psj(i,j)=ord;
72         end
73     %Função de soma do tempo de atravessamento planejado de todas as
74     estações para cada ordem i
75     if j==1
76         tpa_planejado(i)=ord;
77     else
78         tpa_planejado(i)=tpa_planejado(i)+ord;
79     end
80 end
81
82 %Função calcula Tds que é a data de chegada + tempo de atravessamento
83 %planejado
84
85 for i=1:L
86     Tds(i)=inter_chegada(i)+tpa_planejado(i);
87 end
88
89 %Função que calcula a data de entrega através de uma uniforme(35,40)
90
91 for i=1:L
92     data_entrega(i,1)=unifrnd(35,40); %(i,1) é para data_entrega ficar na
93     forma de vetor
94 end
95
96 %Função que atribui Psj à matriz Psj2 e acrescenta a coluna da soma dos
97 tempos nas estações
98
99 Psj2=vt_ordens; %Atribui o ID na primeira coluna de Psj2
100
101 %Função que coloca na coluna 2 de Psj2 o intervalo de chegadas
102 %Adiciona também nas coluna 3 a 8 os tpa_planejados (matriz Psj)
103
104 for i=1:L
105     Psj2(i,2)=inter_chegada(i);% Coloca na coluna 2 o intervalo de chegada
106     for j=1:c
107         Psj2(i,j+2)=Psj(i,j);
108         %Psj2(i,j+2) a partir da coluna 3 os valores de Psj serão add
109     end
110 end

```

APÊNDICE A.3: Algoritmo básico de liberação de ordens do WLC em um ambiente de programação simples, com linguagem estruturada e de alto nível (parte 3).

```

112 %Função que calcula a diferença entre a data de chegada(Tds) e a Data de
113 entrega (data_entrega)
114 % Chamar de tempo de liberação planejada : TRj
115 for i=1:L
116     TRj(i)=data_entrega(i)-Tds(i);
117 end
118
119 % Função que adiciona tempo de atravessamento planejado a matriz Psj2
120 (coluna 9)
121 % Função que adiciona a soma do tempo de atravessamento planejado a matriz
122 Psj2 (coluna 10)
123 % Função que adiciona a data de entrega a matriz Psj2 (coluna 11)
124 % Função que adiciona o tempo de liberação planejada (TRj) a matriz Psj2
125 (coluna 12)
126 for i=1:L
127     Psj2(i,c+3)=tpa_planejado(i);
128     %Por conta da função anterior, a Psj2 passou a ter dimensão 8
129     %Então c+3 (=9), faz com que tpa_planejado seja implementado
130     %na coluna 9
131     Psj2(i,c+4)=Tds(i);
132     % O mesmo acontece aqui, onde a Tds é implementado na coluna 10
133
134     Psj2(i,c+5)=data_entrega(i);
135     % O mesmo acontece aqui, onde a data de entrega é implementada
136     % na coluna 11
137
138     Psj2(i,c+6)=TRj(i);
139     % O mesmo acontece aqui, onde TRJ é implementado
140     % na coluna 12
141 end
142
143 %Função zerar os valores do tempo de atravessamento planejado das estações
144 de Psj3
145
146 for i=1:L
147     for j=1:c
148         Psj3(i,j+2)=zeros;
149     end
150 end
151
152 %Função que adiciona à matriz Psj3 os valores do tempo de atravessamento
153 planejado da matriz Psj2, já ordenados de 1 a 6.
154
155 for i=1:L
156     for j=1:c
157         switch roteiros(i,j)
158             case(1)
159                 Psj3(i,1+2)=Psj2(i,j+2);
160             case(2)
161                 Psj3(i,2+2)=Psj2(i,j+2);
162             case(3)
163                 Psj3(i,3+2)=Psj2(i,j+2);
164             case(4)
165                 Psj3(i,4+2)=Psj2(i,j+2);
166             case(5)
167                 Psj3(i,5+2)=Psj2(i,j+2);
168             case(6)
169                 Psj3(i,6+2)=Psj2(i,j+2);
170             otherwise
171                 break
172         end
173     end
174 end
175
176 end
177
178 end

```


APÊNDICE A.4: Algoritmo básico de liberação de ordens do WLC em um ambiente de programação simples, com linguagem estruturada e de alto nível (parte 4).

```
179
180 %Função ordena a matriz Psj3 pelo tempo de liberação planejado (ordena por
181 %linhas)
182
183 %Primeiro ordena vetor de tempo de liberação planejado (TRj)
184
185 TRj_ordenado=sort(TRj); %Ordena TRj do menor para o maior
186
187 %Cria uma matriz Psj3_ordenado
188
189 Psj3_ordenado=zeros;
190
191 %Calcula a quantidade de colunas de Psj3 e sala em tamanho
192
193 tamanho=length(Psj3(1,:));
194
195 %Função que adiciona os valores de Psj3 ordenadamente na matriz
196 %Psj3_ordenado
197
198 for i=1:L
199     for j=1:L
200         if TRj_ordenado(i)== TRj(j)
201             for k=1:tamanho
202                 Psj3_ordenado(i,k)=Psj3(j,k);%Adiciona na linha i, em cada k
                %coluna da matriz Psj3_ordenado, os valores de cada coluna k
                %da linha j da matriz Psj3
203             end
204             break
205         end
206     end
207 end
```

APÊNDICE B: Algoritmo na entidade “Chegada”, aba “Triggers”, campo “OnCreation”

```

Object item = param(1);
Object current = ownerobject(c);
int port = param(2);
int rownumber = param(2); //row number of the schedule/sequence table
{ // ***** PickOption Start ***** //
/**popup:SetLabel:hasitem=0*/
/**Set Label*/
Object involved = /** \nObject: *//**tag:object*//**/item/**/;
string labelname = /** \nLabel: *//**tag:label*//**/"numeroOperacoes"/**/;
Variant value = /** \nValue: *//**tag:value*//**/duniform(1,6)/**/;
involved.labels.assert(labelname).value = value;
} // ***** PickOption End ***** //

setlabel(item,"tempoChegada",time()); //adiciona uma label de tempo de chegada (dia
de chegada)

setlabel(item,"diaEntrega",duniform(35,60)+time()); // adiciona uma label com a
data de entrega prevista

int OPS = getlabel(item,"numeroOperacoes"); // Adiciona o n mero de opera es em
OPS

intarray OPE = makearray(6); //Cria um vetor OPE que ir  receber os valores de OP1
a OP6

//Comando OP1:

{ // ***** PickOption Start ***** //
/**popup:SetLabel:hasitem=0*/
/**Set Label*/
Object involved = /** \nObject: *//**tag:object*//**/item/**/;
string labelname = /** \nLabel: *//**tag:label*//**/"OP1"/**/;
Variant value = /** \nValue: *//**tag:value*//**/duniform(1,6)/**/;

involved.labels.assert(labelname).value = value;
} // ***** PickOption End ***** //

OPS = (OPS -1); // Decrementa o n mero de opera es

setlabel(item,"TP1",erlang(0.0, 0.5, 2.0, 0)); //cria a tempo de processamento para
OP1

//Comando OP2:
if (OPS >0){

{ // ***** PickOption Start ***** //
/**popup:SetLabel:hasitem=0*/
/**Set Label*/
Object involved = /** \nObject: *//**tag:object*//**/item/**/;
string labelname = /** \nLabel: *//**tag:label*//**/"OP2"/**/;
Variant value = /** \nValue: *//**tag:value*//**/duniform(1,6)/**/;

involved.labels.assert(labelname).value = value;
} // ***** PickOption End ***** //

    int status = 0;

    while(status==0){
        if (getlabel(item,"OP2")==getlabel(item,"OP1)){
            setlabel(item,"OP2",duniform(1,6));
        }
    }
}

```

```

    }
    else{
        status = 1;
    } //end while

OPS = (OPS -1); // Decrementa o número de operações

setlabel(item,"TP2",erlang(0.0, 0.5, 2.0, 0)); //cria a tempo de processamento para
OP2

} //end comando 2

if (OPS >0){ //Comando OP3

{ // ***** PickOption Start ***** //
/**popup:SetLabel:hasitem=0*/
/**Set Label*/
Object involved = /** \nObject: /**tag:object***/item/**/;
string labelname = /** \nLabel: /**tag:label***/"OP3"/**/;
Variant value = /** \nValue: /**tag:value***/duniform(1,6)/**/;

involved.labels.assert(labelname).value = value;
} // ***** PickOption End ***** //

    int status = 0;

    while(status==0){
        if (getlabel(item,"OP3")==getlabel(item,"OP1") ||
(getlabel(item,"OP3")==getlabel(item,"OP2"))){
            setlabel(item,"OP3",duniform(1,6));
        }
        else{
            status = 1;
        }
    } //end while

OPS = (OPS -1); // Decrementa o número de operações

setlabel(item,"TP3",erlang(0.0, 0.5, 2.0, 0)); //cria a tempo de processamento para
OP3

} //end comando 3.

if (OPS > 0){ //Comando OP4

{ // ***** PickOption Start ***** //
/**popup:SetLabel:hasitem=0*/
/**Set Label*/
Object involved = /** \nObject: /**tag:object***/item/**/;
string labelname = /** \nLabel: /**tag:label***/"OP4"/**/;
Variant value = /** \nValue: /**tag:value***/duniform(1,6)/**/;

involved.labels.assert(labelname).value = value;
} // ***** PickOption End ***** //

    int status = 0;

    while(status==0){
        if (getlabel(item,"OP4")==getlabel(item,"OP1") ||
(getlabel(item,"OP4")==getlabel(item,"OP2")) || (getlabel(item,"OP4")==getlabel(item,
"OP3"))){
            setlabel(item,"OP4",duniform(1,6));
        }
        else{
            status = 1;
        }
    }
}

```

```

    }
    } //end while

OPS = (OPS -1); // Decrementa o número de operações

setlabel(item,"TP4",erlang(0.0, 0.5, 2.0, 0)); //cria a tempo de processamento para
OP4

} // end comando 4.

if (OPS > 0){ //Comando OP5

{ // ***** PickOption Start ***** //
/**popup:SetLabel:hasitem=0*/
/**Set Label*/
Object involved = /** \nObject: /**tag:object/**/item/**/;
string labelname = /** \nLabel: /**tag:label/**/"OP5"/**/;
Variant value = /** \nValue: /**tag:value/**/duniform(1,6)/**/;

involved.labels.assert(labelname).value = value;
} // ***** PickOption End ***** //

int status = 0;

while(status==0){
    if (getlabel(item,"OP5")==getlabel(item,"OP1") ||
(getlabel(item,"OP5")==getlabel(item,"OP2")) || (getlabel(item,"OP5")==getlabel(item,
"OP3")) || (getlabel(item,"OP5")==getlabel(item,"OP4"))){
        setlabel(item,"OP5",duniform(1,6));
    }
    else{
        status = 1;
    }
} //end while

OPS = (OPS -1); // Decrementa o número de operações

setlabel(item,"TP5",erlang(0.0, 0.5, 2.0, 0)); //cria a tempo de processamento para
OP5

} //end comando 5.

if (OPS > 0){ //Comando OP6

{ // ***** PickOption Start ***** //
/**popup:SetLabel:hasitem=0*/
/**Set Label*/
Object involved = /** \nObject: /**tag:object/**/item/**/;
string labelname = /** \nLabel: /**tag:label/**/"OP6"/**/;
Variant value = /** \nValue: /**tag:value/**/duniform(1,6)/**/;

involved.labels.assert(labelname).value = value;
} // ***** PickOption End ***** //

int status = 0;

while(status==0){
    if (getlabel(item,"OP6")==getlabel(item,"OP1") ||
(getlabel(item,"OP6")==getlabel(item,"OP2")) || (getlabel(item,"OP6")==getlabel(item,
"OP3")) || (getlabel(item,"OP6")==getlabel(item,"OP4")) || (getlabel(item,"OP6")==getla
bel(item,"OP4")) || (getlabel(item,"OP6")==getlabel(item,"OP5"))){
        setlabel(item,"OP6",duniform(1,6));
    }
} //end while

```

```

    }
    else{
        status = 1;
    }
} //end while

OPS = (OPS -1); // Decrementa o nmero de operaes

setlabel(item,"TP6",erlang(0.0, 0.5, 2.0, 0)); //cria a tempo de processamento para
OP6

} //end comando 6.

//cria uma label de data de entrega prevista baseada no nmero de operaes
multiplicado pela soma do dia de chegada com a distribuio do tempo de
processamento
//setlabel(item,"tempoAtravessamento",((getlabel(item,"numeroOperacoes")*(erlang(0.
0, 0.5, 2.0, 0)))+(getlabel(item,"tempoChegada"))));

intarray TP = makearray(6);

TP[1] = getlabel(item,"TP1");
TP[2] = getlabel(item,"TP2");
TP[3] = getlabel(item,"TP3");
TP[4] = getlabel(item,"TP4");
TP[5] = getlabel(item,"TP5");
TP[6] = getlabel(item,"TP6");

setlabel(item,"tempoAtravessamento",((TP[1]+TP[2]+TP[3]+TP[4]+TP[5]+TP[6])+(getlabel
l(item,"tempoChegada"))));

//cria lebal data de liberao planejada
setlabel(item,"DlPlanejada", (getlabel(item,"diaEntrega")-
getlabel(item,"tempoAtravessamento")));

//Comando que verifica se OPS  zero, caso positivo finaliza
// Espera-se que caso OPS seja zero em algum valor antes de 6, todas as operaes
no preenchidas recebam zero.

if (OPS == 0){
return 0;

}{ // ***** PickOption Start ***** //
/**popup:SetLabel:hasitem=0*/
/**Set Label*/
Object involved = /** \nObject: ****tag:object****/item/**/;
string labelname = /** \nLabel: ****tag:label****/"tempoChegada"****/;
Variant value = /** \nValue: ****tag:value****/time()****/;

involved.labels.assert(labelname).value = value;
} // ***** PickOption End ***** //
{ // ***** PickOption Start ***** //
/**popup:SetLabel:hasitem=0*/
/**Set Label*/
Object involved = /** \nObject: ****tag:object****/item/**/;
string labelname = /** \nLabel: ****tag:label****/"diaEntrega"****/;
Variant value = /** \nValue: ****tag:value****/duniform(35,60)+time()****/;

```

```
involved.labels.assert(labelname).value = value;  
} // ***** PickOption End ***** //
```

APÊNDICE C: Algoritmo entidade “TIC”, aba “Triggers”, campo “OnExit”

```
/**Custom Code*/
Object item = param(1);
Object current = ownerobject(c);
int port = param(2);

if (fmod(time(),5) == 0) {

    openoutput(centerobject(current,1));
    setlabel(current,"fechado",0);
    settablenum("Cargas",1,9,time());
    settablenum("Cargas",1,10,getlabel(current,"fechado")); // escreve se a pool
esta aberta ou fechada

}

if (fmod(time(),5) != 0) {

    closeoutput(centerobject(current,1));
    setlabel(current,"fechado",1);
    settablenum("Cargas",1,10,getlabel(current,"fechado")); // escreve se
a pool esta aberta ou fechada

}
```

APÊNDICE D: Algoritmo entidade “Pool”, aba “Triggers”, campo “OnEntry

```

Object item = param(1);
Object current = ownerobject(c);
int port = param(2);

{ // ***** PickOption Start ***** //
/**popup:SortByExpression*/
/**Sort by Expression*/
int Ascending = 1;
int Descending = 2;
int order = /** \nSort Order:
****tag:choice****/Ascending/**list:Ascending~Descending*/;
/** \n\nSort entering flowitems in the given order according to the value*/
treenode olditem = param(1);
double curitemtype = /** \nExpression:
****tag:expression****/(getlabel(item, "DlPlanejada")+getlabel(item, "Ranqueamento
"))/**/;
int maxrank = 1;
for (int i = 1; i <= current.subnodes.length; i++) {
    Object item = current.subnodes[i];
    double compare = /** \nCompare To:
****tag:compareto****/(getlabel(item, "DlPlanejada")+getlabel(item, "Ranqueamento
"))/**/;
    if (order == Ascending) {
        if (compare > curitemtype)
            break;
    } else if (compare < curitemtype)
        break;
    maxrank++;
}
olditem.rank = min(maxrank, current.subnodes.length);
/**\n*/
} // ***** PickOption End ***** //

```


APÊNDICE E: Algoritmo entidade “Pool”, aba “Triggers”, campo “OnExit

```
Object item = param(1);
Object current = ownerobject(c);
int port = param(2);
{ // ***** PickOption Start ***** //
  /**popup:SetLabel:hasitem=0*/
  /**Set Label*/
  Object involved = /** \nObject: *//**tag:object**/**/item/**/;
  string labelname = /** \nLabel: *//**tag:label**/**/"saidaPool"/**/;
  Variant value = /** \nValue: *//**tag:value**/**/time()/**/;

  involved.labels.assert(labelname).value = value;
} // ***** PickOption End ***** //
```

APÊNDICE F: Algoritmo entidade “Pool”, aba “Flow”, campo “Send To Port”

```

**Custom Code*/
Object item = param(1);
Object current = ownerobject(c);

intarray carga = makearray(6);
intarray OP = makearray(6);
intarray TP = makearray(6);

int i;
int boolean = 0;

int norma = gettablenum("Cargas",1,7);

carga[0] =0; // SÃ³ para nÃ£o dar erro quando a OP for 0
carga[1] = gettablenum("Cargas",1,1);
carga[2] = gettablenum("Cargas",1,2);
carga[3] = gettablenum("Cargas",1,3);
carga[4] = gettablenum("Cargas",1,4);
carga[5] = gettablenum("Cargas",1,5);
carga[6] = gettablenum("Cargas",1,6);

OP[1] = getlabel(item,"OP1");
OP[2] = getlabel(item,"OP2");
OP[3] = getlabel(item,"OP3");
OP[4] = getlabel(item,"OP4");
OP[5] = getlabel(item,"OP5");
OP[6] = getlabel(item,"OP6");

TP[1] = getlabel(item,"TP1");
TP[2] = getlabel(item,"TP2");
TP[3] = getlabel(item,"TP3");
TP[4] = getlabel(item,"TP4");
TP[5] = getlabel(item,"TP5");
TP[6] = getlabel(item,"TP6");

for (i=1;i<7;i++)
{
    if (((carga[OP[i]])+TP[i]) > norma)
    { //Excede a norma estabelecida para estaÃ§Ã£o i
        return 7; //Saida para Queue1+Processor1
    }

    else {
        if ((i==6) || (OP[i+1]==0)) //OP[i+1]==0 indica que hÃ¡ menos que 6
operaÃ§Ãµes na ordem
        {
            return OP[1]; //1era porta do roteiro
        }
    } //end else
}

```

APÊNDICE G: Algoritmo entidade “Fila Est 1 (Queue)”, aba “Triggers”, campo “OnEntry”

```

/**Custom Code*/
Object item = param(1);
Object current = ownerobject(c);
int port = param(2);

setlabel(item,"Etapa",getlabel(item,"Etapa")+1);
// Na entrada da fila, soma o label "Etapa" do item em 1.

if (getlabel(item,"Etapa")==1)
{

    doublearray TP = makearray(6);
    intarray OP = makearray(6);
    //for (int j=1;j++;j<7)
    //{ OP[j]=0;}

    TP[1] = getlabel(item,"TP1");
    TP[2] = getlabel(item,"TP2");
    TP[3] = getlabel(item,"TP3");
    TP[4] = getlabel(item,"TP4");
    TP[5] = getlabel(item,"TP5");
    TP[6] = getlabel(item,"TP6");

    OP[1] = getlabel(item,"OP1");
    OP[2] = getlabel(item,"OP2");
    OP[3] = getlabel(item,"OP3");
    OP[4] = getlabel(item,"OP4");
    OP[5] = getlabel(item,"OP5");
    OP[6] = getlabel(item,"OP6");

    int w = 1 ; //Define var pivote desde 1

    for (w=1;w<=getlabel(item,"numeroOperacoes");w++)
    //ex: numeroOperacoes = 6, i<7.

    {

        settablenum("Cargas",1,OP[w], (gettablenum("Cargas",1,OP[w]))+TP[w]);
        //escreve na tabela cargas o valor correspondente de Q1
        //Faz uma funÃ§Ã£o de incremento com o valor jÃ¡ existente em cargas
    }

    //settablenum("Cargas",1,1, (gettablenum("Cargas",1,1)+TP[i]));

```

APÊNDICE H: Algoritmo entidade “Máquina 3 (Processor)”, aba “Triggers”, campo “OnEntry”

```

/**Custom Code*/
Object item = param(1);
Object current = ownerobject(c);
int port = param(2);

setlabel(item,"numeroOperacoes",getlabel(item,"numeroOperacoes")-1);
// Na entrada na estaÃ§Ã£o, diminui o label "numeroOperacoes" do item em 1.

if (getlabel(item,"OP1") > 0) { //Caso OP1 seja <> 0, esta Ã© a primeira operacao da
ordem.
    setlabel(item,"OP1",0); //Indica que a primeira operaÃ§Ã£o jÃ¡ foi realizada.
    setlabel(item,"proximaPorta",getlabel(item,"OP2")); } // cria o label
"proximaPorta", o faz assumir o valor do label "OP2".
else {
    //caso OP1 = 0
    if (getlabel(item,"OP2") > 0) { //caso OP2 ainda nÃ£o tenha sido zerado (esta
Ã© a segunda operaÃ§Ã£o)
        setlabel(item,"OP2",0); //Indica que a seunda operaÃ§Ã£o vai ser
realizada.
        setlabel(item,"proximaPorta",getlabel(item,"OP3")); } //Faz o label
"proximaPorta" assumir o valor de "OP3".
    else {
        //caso OP1 = 0 e OP2 = 0
        if (getlabel(item,"OP3") > 0) { //caso OP3 ainda nÃ£o tenha sido zerado
(esta Ã© a terceira operaÃ§Ã£o)
            setlabel(item,"OP3",0); //Indica que a terceira operaÃ§Ã£o vai
ser realizada.
            setlabel(item,"proximaPorta",getlabel(item,"OP4")); } //Faz o
label "proximaPorta" assumir o valor de "OP4".
        else {
            //caso OP1 = 0 e OP2 = 0 e OP3 = 0

            if (getlabel(item,"OP4") > 0) { //caso OP4 ainda nÃ£o tenha sido
zerado (esta Ã© a quarta operaÃ§Ã£o)
                setlabel(item,"OP4",0); //Indica que a quarta operaÃ§Ã£o
vai ser realizada.
                setlabel(item,"proximaPorta",getlabel(item,"OP5")); } }
//Faz o label "proximaPorta" assumir o valor de "OP5".
        else {
            //caso OP1 = 0 e OP2 = 0 e OP3 = 0 e OP4 = 0

            if (getlabel(item,"OP5") > 0) { //caso OP5 ainda nÃ£o
tenha sido zerado (esta Ã© a quinta operaÃ§Ã£o)
                setlabel(item,"OP5",0);

                setlabel(item,"proximaPorta",getlabel(item,"OP6")); } //Faz o label
"proximaPorta" assumir o valor de "OP6".
                //Indica que a quinta operaÃ§Ã£o vai ser
realizada.

                else{
                    if (getlabel(item,"OP6") > 0) {
                        setlabel(item,"proximaPorta",getlabel(item,7));
                    }
                }
            }
        }
    }
}

int tempo = time();
if (tempo >= 3000) { //condiciona o registro do histÃ³rico de carga ao tempo 3000.
Tempo de warm up

```

```
{ // ***** PickOption Start ***** //
/**popup:IncrementValue*/
/**Increment Value*/
treenode thenode = /** \nNode: */ /**tag:node*//**/current.labels["indice"]/**/;
double value = /** \nIncrement By: */ /**tag:value*//**/1/**/;
inc(thenode,value);
} // ***** PickOption End ***** //

//Coloca o valor da carga 1 na label da máquina
setlabel(current,"Carga3",gettablenum("Cargas",1,3));

//carrega a global table de histórico
settablenum("Historico
Cargas",getlabel(current,"indice"),3,(gettablenum("Cargas",1,3)));
}
```

**APÊNDICE I: Algoritmo entidade “Máquina 3 (Processor)”, aba “Triggers”, campo
“OnExit”**

```
/**Custom Code*/
Object item = param(1);
Object current = ownerobject(c);
int port = param(2);

intarray TP = makearray(6);

TP[1] = getlabel(item,"TP1");
TP[2] = getlabel(item,"TP2");
TP[3] = getlabel(item,"TP3");
TP[4] = getlabel(item,"TP4");
TP[5] = getlabel(item,"TP5");
TP[6] = getlabel(item,"TP6");

int i = getlabel(item,"Etapa");// i é atualizado com o valor da etapa

//Decrementa na tabela cargas o valor correspondente de TP que está; saindo
settablenum("Cargas",1,3,(gettablenum("Cargas",1,3)-TP[i]));
```

APÊNDICE J: Algoritmo entidade “Saída (Sink)”, aba “Triggers”, campo “OnEntry”

```

Object item = param(1);
Object current = ownerobject(c);
int port = param(2);

{ // ***** PickOption Start ***** //
/**popup:SetLabel:hasitem=0*/
/**Set Label*/
Object involved = /** \nObject: *//**tag:object*//**item/**/;
string labelname = /** \nLabel: *//**tag:label*//**"tempoSaidaTotal"/**/;
Variant value = /** \nValue: *//**tag:value*//**time()/**/;

involved.labels.assert(labelname).value = value;
} // ***** PickOption End ***** //

settablenum("Cargas",1,8,(getlabel(item,"tempoSaidaTotal")-
(getlabel(item,"tempoChegada"))));
setlabel(item,"throughputTotal",gettablenum("Cargas",1,8));{ // *****
PickOption Start ***** //
/**popup:SetLabel:hasitem=0*/
/**Set Label*/
Object involved = /** \nObject: *//**tag:object*//**current/**/;
string labelname = /** \nLabel: *//**tag:label*//**"throughputTotal"/**/;
Variant value = /** \nValue: *//**tag:value*//**gettablenum("Cargas",1,8)/**/;

involved.labels.assert(labelname).value = value;
} // ***** PickOption End ***** //
{ // ***** PickOption Start ***** //
/**popup:SetLabel:hasitem=0*/
/**Set Label*/
Object involved = /** \nObject: *//**tag:object*//**current/**/;
string labelname = /** \nLabel: *//**tag:label*//**"throughputChao"/**/;
Variant value = /** \nValue: *//**tag:value*//**gettablenum("Cargas",1,11)/**/;

involved.labels.assert(labelname).value = value;
} // ***** PickOption End ***** //

settablenum("Cargas",1,11,(getlabel(item,"tempoSaidaTotal")-
(getlabel(item,"saidaPool"))));

int tempo = time();
if (tempo >= 0){ //condiciona o registro dos throughput total e chÃ£o de fÃ;brica
ao tempo 3000.

{ // ***** PickOption Start ***** //
/**popup:IncrementValue*/
/**Increment Value*/
treenode thenode = /** \nNode: */ /**tag:node*//**current.labels["indice"]/**/;
double value = /** \nIncrement By: */ /**tag:value*//**1/**/;
inc(thenode,value);
} // ***** PickOption End ***** //

//carrega a global table Throughput chÃ£o de fÃ;brica
settablenum("Throughput",getlabel(current,"indice"),2,(getlabel(current,"throughput
Chao")));

//carrega a global table Throughput total
settablenum("Throughput",getlabel(current,"indice"),1,(getlabel(current,"throughput
Total")));

}

```