UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM

ENGENHARIA DE PRODUÇÃO

Avaliação da Complexidade da Modelagem Híbrida em Processos de Manufatura *Flow Shop*

Tiago Dela Savia

ITAJUBÁ

2019

UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Tiago Dela Savia

Avaliação da Complexidade da Modelagem Híbrida em Processos de Manufatura *Flow Shop*

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Produção como parte dos requisitos para a obtenção do título de Mestre em Engenharia de Produção

Área: Engenharia de Produção

Orientador: Prof. Dr. Alexandre Ferreira de Pinho

ITAJUBÁ

2019

AGRADECIMENTOS

A inteligência, dom inerente ao homem e dado por Deus, é o que move as engrenagens da sociedade e possibilita as transformações no mundo. O uso deste dom divino levou o ser humano à descoberta da Ciência, esta responsável pela iluminação das nossas consciências perante o obscuro e o desconhecido. A minha entrada no mundo da pesquisa acadêmica foi algo conseguido com muito esforço, sendo necessário ainda mais esforço para concluir esta pesquisa. Sou muito grato pela sensação de olhar este documento e perceber que todo meu esforço gerou frutos.

Contudo, sozinho eu nunca seria capaz desta empreitada. Em primeiro lugar, eu não seria o homem que sou hoje sem o apoio de pessoas especiais na minha vida, como minha mãe Margarida e meu pai Orafran, além da minha irmã Juliana. Pelos meus pais foi me dado o dom da vida, o sustento material e a educação moral que me tornaram quem sou. O amor, carinho e compreensão deles me sustentaram nessa experiência de morar em um local até então desconhecido e trabalhar em um projeto significativo e exigente. Nesta vida também fiz muitos amigos, alguns dos quais mantive contato mesmo à distância. Agradeço de coração ao apoio destes durante meu período em outra cidade.

Não menos importante, não poderia deixar de agradecer a todos os professores que contribuíram com o meu crescimento intelectual e acadêmico. Em especial gostaria de agradecer imensamente ao Prof. Dr. Alexandre Ferreira de Pinho por ter sugerido o tema deste trabalho e por ter confiado na minha capacidade de executá-lo. Muito obrigado Pinho pela orientação e pelas diversas sugestões de melhoria para o meu trabalho. Agradeço também a todos os colegas da Unifei que me ajudaram direta ou indiretamente nesta pesquisa, em especial aos professores João Paulo Barbieri por ter me ajudado com sua experiência em trabalhos anteriores e Davi Custódio de Sena pela imensa ajuda para concluir um dos modelos de simulação apresentados nesta dissertação.

Sou muito grato também a todas as experiências e amizades que a cidade de Itajubá me proporcionou. Aos amigos que fiz na Unifei, em especial os colegas de trabalho do NEAAD (que me acompanharam nos cafés da tarde diariamente), psicóloga Liliane que me ouviu nos momentos de angústia, amigos que fiz nos eventos e atividades esportivas da universidade (em especial meus alunos de Ving Tsun e pessoal do sensei Adilson), grupos espíritas, amigos que conheci nas saídas de fim de semana (agradeço ao acolhimento do pessoal do D20 bar) e em demais eventos na cidade. Minha estadia em Itajubá não seria a mesma se não fosse por vocês!

Mudar de cidade nem sempre é fácil, e toda essa caminhada não teria sido possível se não fosse pelo apoio financeiro que recebi através da CAPES. Agradeço às pessoas que compartilharam a moradia comigo e fizeram companhia durante esses dois anos, em especial ao Wilson e seu Lauro e dona Raquel pela hospitalidade. Também não posso me esquecer dos motoristas “caroneiros” que me possibilitaram rever minha família e amigos de tempos em tempos.

Concluindo, gostaria de agradecer a todos da Universidade Federal de Itajubá (UNIFEI), à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e à Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) pelo apoio e incentivo à pesquisa.

RESUMO

A simulação computacional é considerada uma das mais poderosas ferramentas de auxílio à tomada de decisões, podendo ser utilizada para analisar a operação de sistemas complexos, testando hipóteses e prevendo seu comportamento futuro. Uma das características da simulação é a capacidade de reduzir a complexidade dos sistemas reais através da sua representação nos modelos computacionais. Contudo, com o aumento da complexidade dos processos de manufatura em si, os modelos computacionais têm se tornado cada vez mais complexos, o que pode impactar na acessibilidade e funcionalidade destes. No contexto da simulação aplicada em gestão de operações existe uma discussão a respeito da utilização da simulação baseada em agentes (SBA), que começou a ganhar destaque a partir dos anos 2000, em contraponto (às vezes complementando uma a outra) com a simulação a eventos discretos (SED), que há mais de 40 anos vem sendo a prática mais comum de simulação dentro da pesquisa operacional. Este trabalho consiste em um estudo de caso com o objetivo de se comparar o impacto da simulação híbrida (que utiliza elementos da SBA em conjunto com a SED) na complexidade dos modelos de simulação, em processos de manufatura do tipo *flow shop*, desenvolvidos através do *software* Anylogic®. Um *framework* foi elaborado para possibilitar a mensuração da complexidade dos modelos, utilizado para comparar pares de modelos que consistem, cada par, em um modelo construído a partir da simulação a eventos discretos pura e outro através da simulação híbrida. Por esta análise foi possível perceber que os modelos construídos através da simulação híbrida apresentaram maior complexidade em três dos quatro casos analisados, sendo que a diferença de complexidade entre modelos variou em proporção segundo cada caso.

Palavras-chave: Simulação a Eventos Discretos; Simulação Baseada em Agentes; Simulação Híbrida; Complexidade.

ABSTRACT

Computer simulation is considered one of the most powerful tools for decision making, as it can be used to analyze the operation of complex systems by testing hyphoteses and predicting its future behavior. One of simulation's characteristics is its ability to reduce the complexity of real systems by representing them in the form of computer models. However, with the rise of manufacture processes themselves, computer models have become more and more complex with time, which can cause an impact on their accessibility and functionality. In the context of operations management there is a discussion regarding the applicability of agent based simulation (ABS), which started to gain popularity in the early 2000s, in contrast (and sometimes complementing each other) to discrete-event simulation (DES), which has been the most common practice in operations research for more than 40 years. This paper consists in a case study comparing the impact of hybrid simulation, called HS for short, (utilizing elements of "agents" in a discrete event model) on the complexity of simulation models, which represents flow shop manufacturing processes, developed in the simulation package Anylogic®. A framework was developed to measure the complexity of pairs of models, which consists, each pair, in a DES and a HS based model respectively, representing the same process by the two different approaches. The analysis showed that most models developed in hybrid simulation presented a bigger complexity, with the exception of one. The difference in complexity varied pair by pair, according to each case.

Keywords: Discrete Event Simulation; Agent Based Simulation; Hybrid Simulation; Complexity.

Lista de Figuras

Lista de Quadros

Lista de Tabelas

Lista de Siglas

SED – Simulação a Eventos Discretos

SBA – Simulação Baseada em Agentes

SH – Simulação Híbrida

Sumário

1.INTRODUÇÃO

A crescente complexidade dos sistemas de manufatura, caracterizados por um comportamento imprevisível, dinâmico e não-linear, apresenta desafios para as empresas que estão em busca da constante otimização de seus processos (EFTHYMIOU *et al*., 2014). Essa complexidade deriva de vários fatores, dos quais pode-se incluir a integração de componentes físicos e *software*, um grande número de dados a se processar, entre outros (VAN MIERLO & VANGHELUWE, 2018). Para se manterem competitivas no mercado, as organizações precisam adaptar seus processos de tomada de decisões ao ambiente complexo e imprevisível deste (HAO & SHEN, 2008). Neste contexto, Sharma (2015) aponta a simulação como uma ferramenta de grande importância para prevenir decisões catastróficas tomadas pela inobservância de mudanças externas.

A simulação computacional é uma ferramenta que permite representar um sistema real através de um modelo computacional, para assim analisar o seu comportamento sob diversas condições, como para realizar projeções futuras ou saber qual será o impacto causado por alguma mudança, sem precisar interferir no sistema real. Segundo Jacob (2013) e Sharma (2015), os computadores podem ser usados para simular a operação de sistemas complexos, estudar sua performance, fazer experimentos, testar a influência de diferentes *inputs* no resultado final, entre outras coisas.

Sabe-se que a simulação pode ser utilizada para se representar sistemas complexos. Ahmed *et al.* (2016) apontam que a modelagem e simulação é a atividade que visa reduzir a complexidade encontrada no mundo real, e que deve-se tentar manter os modelos de simulação mais simples e pequenos quanto possível. No entanto, o que se percebe na realidade é um aumento da complexidade dos próprios modelos de simulação com o passar do tempo (CHWIF *et al.* 2000; AHMED *et al.* 2016).

Estima-se que a escolha do tipo de abordagem a ser utilizada na simulação possa impactar diretamente na complexidade do modelo final. Entre as técnicas de Simulação, uma das mais utilizadas é a Simulação a Eventos Discretos (SED), amplamente utilizada para representar o fluxo de materiais em sistemas de manufatura (JAHANGIRIAN *et al.*, 2010), caracterizada por ter ênfase nos processos que compõem o sistema, considerando os elementos necessários para as atividades (pessoal, máquinas etc.) como recursos. Por outro lado, existe a Simulação Baseada em Agentes (ou SBA), que diferentemente de outras abordagens constrói os modelos

focando no comportamento de diversos “agentes”, nome dado aos entes que constituem o sistema.

Porém ainda existe a possibilidade de se combinar mais de uma abordagem de simulação na construção do modelo computacional, o que pode ser chamado de Simulação Híbrida (SH). Uma das formas de se trabalhar com a SH é através da combinação da SED com a SBA, inserindo agentes para representar alguns elementos dentro de modelos SED. Contudo, de acordo Green *et al.* (2017), a SBA e a SED vêm sido usadas na maior parte das vezes de forma isolada.

Eldabi *et al.* (2018) apontam que o interesse na modelagem híbrida tem aumentado nos últimos 10 anos. Para Powell e Mustafee (2014), diversos autores têm optado pelo uso da Simulação Híbrida combinando os pontos fortes de ambas as técnicas (SED e SBA), o que possibilita a análise de determinados sistemas que não eram abordados de forma completa com o uso de apenas uma delas. Entre esses autores pode-se citar: Maione e Naso (2004), Dubiel e Tsimhoni (2005), Hao e Shen (2008), Fakhimi *et al.* (2014), Green *et al.* (2017) e Viana *et al.* (2018).

1.1. Objetivos

A SBA passou a ganhar evidência a partir dos anos 2000 (SIEBERS *et al.*, 2010), com a promessa de representar de forma mais natural e fidedigna os sistemas estudados. Através da SBA é possível representar os ditos fenômenos *emergentes* (que são imprevisibilidades criadas através da interação dos agentes) (BONABEAU, 2002). No entanto, a SED possui uma história mais longa de aplicações na Pesquisa Operacional, que segundo Siebers (2010) tem sido seu principal pilar a mais de 40 anos. Muito ainda se discute a respeito das vantagens e desvantagens de se utilizar cada uma das abordagens ou uma combinação das duas, sendo que essa escolha deve priorizar a eficiência, rapidez e acessibilidade do modelo, segundo Mustafee *et al.* (2017).

Dentro deste contexto, o objetivo deste trabalho é avaliar se a inserção de agentes em modelos SED pode contribuir para a diminuição da complexidade destes. Foram definidas condições de contorno para direcionar a análise, sendo assim, todos os modelos de simulação usados nessa pesquisa representam problemas de manufatura do tipo *flow shop*, e todos foram construídos através do software Anylogic®.

O trabalho consiste em um estudo de caso, e tem por objetivo específico desenvolver um *framework* para analisar quatro **pares** de modelos que consistem, cada par, em dois modelos

que representam o mesmo sistema, porém cada um construído através de uma abordagem diferente (SED e SH). A cada modelo foram atribuídas notas que refletem a sua complexidade, medida em termos de recursos utilizados em cada um, divididos em: recursos do software (blocos e funções disponíveis no Anylogic®, excluindo agentes), agentes (considerando-se cada elemento utilizado para criar o comportamento daquele agente) e linhas de código (que são digitadas diretamente e são necessárias para realizar algumas funções no modelo).

É preciso destacar que os resultados desta pesquisa poderiam ser diferentes se a medição desenvolvida neste fosse aplicada em um outro tipo de processo produtivo diferente do *flow shop*. Sendo assim, não se pode afirmar que o uso de agentes tornaria mais ou menos complexos modelos que representassem processos do tipo *job shop,* por exemplo.

Estima-se que seja possível, através da análise desenvolvida nesta pesquisa, aprofundar as discussões em torno dos benefícios da Simulação Baseada em Agentes no que diz respeito à redução da complexidade de determinados modelos de simulação, reforçando a hipótese de Macal (2016), que afirma que a Simulação baseada em Agentes pode complementar outros tipos de simulação.

1.2. Estrutura do texto

Esta dissertação aborda a comparação de duas abordagens de simulação diferentes: a Simulação a Eventos Discretos e a Simulação Híbrida (composta por elementos de agentes e de eventos discretos) com o objetivo de avaliar a contribuição de cada abordagem na complexidade final dos modelos. Para dissertar sobre este assunto, ela conta com 5 capítulos.

O ***Capítulo 1*** consiste em uma introdução ao tema, com a contextualização, justificativa e objetivos da dissertação. O ***Capítulo 2*** traz um embasamento teórico sobre o tema, e é subdividido em 5 partes (algumas dessas partes ainda possuem subdivisões), abordando o conceito e a história da simulação, definições das diferentes abordagens de simulação estudadas neste trabalho, conceitos sobre complexidade (em geral e relativa a modelos de simulação) e sobre *flow shop*.

O ***Capítulo 3*** apresenta o método utilizado nesta pesquisa, detalhando-o e explicando o passo a passo do *framework* desenvolvido nesta como objetivo específico. No ***Capítulo 4*** se encontra o estudo de caso com a aplicação do *framework* para comparar a complexidade dos diferentes pares de modelos selecionados como casos para o estudo. Finalmente, no ***Capítulo 5*** se

encontram as conclusões e sugestões para trabalhos futuros. Em seguida seguem os apêndices e as referências bibliográficas utilizadas.

2. REVISÃO BIBLIOGRÁFICA

2.1. Simulação computacional

A simulação computacional consiste na prática de se representar a realidade através de modelos computacionais, os quais podem ser utilizados para prever o comportamento de determinados sistemas e testar hipóteses, sem a necessidade de se atuar nos sistemas reais (LAW & KELTON, 1991). Para Shannon (1998), a simulação é ao mesmo tempo uma ciência e uma arte. A parte científica consiste na análise estatística e programação do modelo computacional. Já a parte artística diz respeito à análise e modelagem do sistema, que pode ser realizada de diversas formas (quais elementos incluir, como representar certo fenômeno etc.).

Segundo Law e Kelton (1991), a simulação permite poupar recursos econômicos, otimizar o uso do tempo, prever possíveis acidentes, projetar plantas, identificar gargalos, entre outras vantagens, sendo útil em qualquer uma das fases do ciclo de vida de um sistema de manufatura. Ingalls (2008) define o processo da simulação como o desenvolvimento de modelos dinâmicos que servem para representar sistemas reais dinâmicos, com o propósito de compreender seu comportamento ou desenvolver estratégias de atuação em cima dele.

Sharma (2015) aponta a importância da simulação, que pode prevenir falhas catastróficas causadas por alguma mudança no sistema. Forrester (1968) e Shannon (1998) definem "sistema" como sendo um agrupamento de partes inter-relacionadas que operam em conjunto, visando um objetivo em comum. Shannon (1998) aponta que os termos modelo e sistema são componentes chave para a definição de simulação. Para White Jr. e Ingalls (2018), um modelo é uma entidade que é usada para representar outra entidade por algum propósito. Modelos são usados quando a investigação do sistema real for impraticável ou proibitiva.

Na prática, um modelo de simulação é composto apenas pelas partes que interessam ao modelador, não precisando necessariamente contemplar todos os elementos do sistema real se estes não afetarem os resultados buscados pela análise. Por exemplo, em uma simulação para se estimar o número ideal de funcionários para atender os clientes que precisam sacar dinheiro ou realizar um depósito em um banco não é necessário representar as outras seções deste banco, a menos que a atividade delas interfira diretamente na atividade sendo analisada (LAW & KELTON, 1991). White Jr. e Ingalls (2018) apontam que o modelo deve conter apenas o escopo e nível de detalhes necessários para satisfazer os objetivos do estudo.

A simulação surgiu da necessidade de se analisar sistemas caracterizados pela aleatoriedade e onde se faz necessário a representação do tempo influenciando o sistema. Para Chwif e Medina (2010), um modelo de simulação consegue capturar com mais fidedignidade tais características aleatórias dos sistemas reais, o que não era possível utilizando apenas modelos matemáticos ou outros métodos como a simulação de Monte Carlo. Segundo Van Mierlo e Vanheluwe (2018), na prática os sistemas complexos exibem um comportamento de processamento de eventos, onde o sistema reage a estímulos externos providos do ambiente em que este se situa. Este comportamento pode ser transportado para os modelos computacionais, permitindo a análise de tais sistemas complexos.

Para North e Macal (2007) existem dois tipos fundamentais de modelos: estocásticos e determinísticos. Os modelos determinísticos tendem a ser os mais simples de se usar e mais baratos. Ao rodar um modelo determinístico mais de uma vez com os mesmos valores de entrada, este gerará sempre os mesmos resultados. Porém este tipo de modelo é muito limitado com relação ao número de situações que pode representar, ao contrário dos modelos estocásticos, que aceitam valores aleatórios. Para estes últimos, cada rodada representa uma possível sequência de eventos, e estes modelos normalmente são analisados várias vezes para se chegar a resultados válidos.

Para Chwif e Medina (2010, p.22) “quanto mais complexo, dinâmico e aleatório for um problema, maior será a aplicabilidade das ferramentas de simulação. Para problemas estáticos (onde os estados do sistema não se alteram com o tempo) a simulação não apresenta utilidade prática, e existem diversos modelos matemáticos que podem ser utilizados. No caso de problemas determinísticos (que não apresentam nenhum componente aleatório) a simulação seria subutilizada, e poderia ser substituída por outra técnica. White Jr. e Ingalls (2018) apontam que situações como essa, onde qualquer estado do modelo possa ser encontrado através de soluções analíticas, são raramente o caso em problemas complexos de manufatura.

De acordo com White Jr. e Ingalls (2018), a simulação é uma abordagem particular de se estudar modelos, e é fundamentalmente experimental. Se assemelha a testes de campo, mas neste caso o sistema de interesse é substituído por um modelo computacional. Simular envolve criar um modelo que imite de forma satisfatória o comportamento de um sistema real para assim observar este comportamento, o que permite testar e comparar diferentes *designs*, validar, experimentar e melhor compreendê-lo.

Para White Jr. e Ingalls (2018) as simulações podem categorizadas, segundo sua abordagem, em Simulação de Monte Carlo, SED, SBA e Sistemas Dinâmicos. Além dessas, ainda existe a possibilidade de se realizar a Simulação Híbrida. As diferentes abordagens consistem em:

- Simulação de Monte Carlo: também denominada simulação estática, é particularmente útil para resolver problemas onde as variáveis possuem comportamento aleatório e não são afetadas pelo tempo. Seu nome deriva de uma cidade conhecida pelos seus cassinos e jogos de azar, onde este tipo de simulação pode prever as jogadas de menor risco. SHANNON 1975; CHWIF & MEDINA, 2010).
- Simulação a Eventos Discretos (SED): nestes modelos de simulação as mudanças só podem se dar em momentos específicos do tempo denominados *eventos.* Exemplos de eventos seriam chegadas de clientes para aguardar em uma fila ou o término de uma atividade. Quando os eventos são ativados, algumas ou todas as variáveis podem sofrer mudanças em seus atributos de uma só vez, não existindo estados intermediários para elas ao decorrer do tempo (SAKURADA & MIYAKE, 2009; SCHRIBBER *et al.*, 2014).
- Simulação Contínua ou Sistemas Dinâmicos: o princípio básico da Dinâmica de Sistemas (do inglês *System Dynamics*, ou SD) é que o comportamento emergente do sistema é definido pela forma com a qual seus elementos internos se relacionam entre si. As relações entre estes elementos são comumente representadas através de grafos onde se busca encontrar relações causais e ciclos de *feedback* (VIANA *et al.*, 2014).
- Simulação Baseada em Agentes (SBA): modelos de simulação que possuem entidades denominadas *agentes*. Esses agentes são elementos autônomos dentro do modelo que seguem uma lógica própria e cada um deles pode mudar de estados de forma diferente durante a simulação. Através deste tipo de abordagem é possível representar de forma mais precisa a complexidade advinda do fator humano, por exemplo (MACAL & NORTH, 2009; SIEBERS *et al.*, 2010).
- Simulação Híbrida: é a simulação que combina mais de um tipo de abordagem no mesmo modelo. A hibridização de modelos pode ser feita transferindo dados de um modelo para o outro (modelados em diferentes plataformas) ou se utilizando de algum programa que possibilite mais de um tipo de abordagem de simulação (VIANA *et al.*, 2014).

2.1.1. História da simulação

Para Nance e Sargent (2002), a simulação começou a ser desenvolvida no contexto histórico da Segunda Guerra Mundial, onde surgiram o modelo de Monte Carlo, a simulação contínua e a simulação a eventos discretos. Já a abordagem de simulação baseada em agentes foi consolidada nos anos 90, apesar de seus conceitos teóricos já terem sido definidos no fim dos anos 40 (CHWIF & MEDINA 2010).

Segundo Pidd (2004), inicialmente a simulação era desenvolvida a partir de linguagens computacionais de propostas gerais, como a FORTRAN. A simulação em seu início era muitas vezes considerada uma ferramenta muito cara e limitada, especialmente para sistemas que apresentavam grande complexidade, o que muitas vezes inviabilizava seu uso como ferramenta (GAVIRA, 2003). De acordo com Lobão & Porto (1999), os *softwares* evoluíram desde os modelos físicos em escala, passando pelos modelos matemáticos até se chegar nos simuladores inteligentes e interativos com interface gráfica.

Nance (1993) dividiu a história da simulação em 5 períodos. O primeiro deles, chamado *período de pesquisa*, durou de 1955 a 1960, e foi caracterizado pelo desenvolvimento dos modelos de representação e as técnicas que permitiam resolver problemas antes impossíveis de serem solucionados pelos modelos matemáticos vigentes. Segundo Lobão e Porto (1999), inicialmente os modelos de simulação eram desenvolvidos a partir de linguagens de programação de propósito geral, o que demandava muito esforço e conhecimento técnico por parte dos profissionais responsáveis. Neste período (em 1958) surgiu o primeiro pacote de simulação, o chamado *General Simulation Program* (GSP) (NANCE & OVERSTREET, 2017).

O segundo período, denominado *advento*, durou de 1961 a 1965, e nele surgiram as principais linguagens de programação destinadas à simulação, como o GPSS e o SIMULA. Segundo Nance & Overstreet (2017), os primeiros programas usados para a simulação eram denominados SPL (*Simulation Programming Language*). Já o terceiro período, segundo Nance (1993), foi denominado *período formativo* e durou de 1966 a 1970. Este período marcou a consolidação das definições conceituais, através de um processo de revisão e refinamento dos conceitos, o que obrigou diversos *softwares* de simulação da época a passarem por intensas transformações.

O quarto período foi denominado *período de expansão* por Nance (1993), durando de 1971 a 1978, e foi quando diversas expansões para os programas de simulação (como o GPSS, SIMSCRIPT e GASP) foram lançadas. O principal fator que contribuiu para a evolução dos

programas de simulação foi a evolução dos processadores computacionais (acentuada no período seguinte, no final dos anos 70), que possibilitou o uso da simulação em computadores domésticos, com interfaces gráficas e animações (NANCE & OVERSTREET, 2017).

O quinto período avaliado pelo autor, o qual dura até os dias de hoje, foi denominado *consolidação*, e neste período foi possível expandir a simulação para vários computadores e microprocessadores. Segundo Kelton *et al.* (2007) foi a partir do início desse período, com a crescente acessibilidade aos computadores, que a simulação foi introduzida como disciplina em cursos de Pesquisa Operacional e Engenharia de Produção. O desenvolvimento da eletrônica e da informática possibilitou incorporar aos modelos físicos em escala recursos tais como microprocessadores e micro-sensores eletrônicos que aumentaram muito a capacidade de obtenção de dados dos mesmos (LOBÃO & PORTO, 1999).

2.2. Simulação a eventos discretos

A Simulação a Eventos Discretos (SED), como apontam Siebers *et al.* (2010), vem sendo há mais de 40 anos o principal pilar da comunidade científica em trabalhos que envolvem simulação na Pesquisa Operacional. Um dos fatores responsáveis por esse sucesso foi o avanço do processamento computacional e a evolução das interfaces de usuário permitiram aos programas de SED se tornarem "*drag and drop*". Para Jacob (2013), a SED é amplamente utilizada para se prever comportamentos futuros, e nesse contexto engenheiros e cientistas se utilizam de enorme poder computacional no *design* e avaliação de novos produtos.

Os principais elementos a serem representados na SED de acordo com Schribber *et al.* (2014) são:

-Entidades: são os entes processados pelo sistema (ou as unidades de tráfico representadas na analogia "transação-fluxo"), que respondem aos eventos. Entidades podem ser **externas** (suas chegadas e movimentos são definidos pelo modelador) ou **internas** (criadas e manipuladas pelo *software* de simulação). Como exemplos de entidades pode-se citar matéria-prima, informação, clientes, etc.;

-Recursos: quaisquer elementos do sistema que realizem algum serviço (como as máquinas em uma linha de produção). As entidades tipicamente necessitam dos recursos para prosseguirem no fluxo, sendo que estes quase sempre têm capacidade limitada. Isso faz com que as entidades em fluxo tenham que competir pelo seu uso, gerando filas;

-Controles: são elementos criados para permitir atrasos e alternativas lógicas no fluxo da simulação, como contadores, variáveis, desvios e outros. Como exemplo pode-se citar determinadas regras de ordenação de filas, baseadas em algum atributo das entidades (como a urgência de tratamento dos pacientes em uma fila de hospital);

-Operações: cada um dos passos dados por ou sobre as entidades enquanto elas atravessam o sistema. Em uma manufatura por exemplo, podemos ter diversas operações pelas quais a matéria-prima tem de passar, como torneamento, fresamento e outras.

Além desses elementos, Ingalls (2008) aponta outros componentes em um programa de Simulação a Eventos Discretos, que são os eventos (que são criados pela interação das entidades com os processos - aqui denominados atividades pelo autor), um gerador de números aleatórios, calendário, variáveis globais e de estado e armazenadores de dados estatísticos. Os modelos SED precisam de geradores de números aleatórios e de um calendário por se tratarem de representações de sistemas estocásticos (onde os valores das variáveis são aleatórios e a dinâmica é afetada pelo tempo).

As variáveis globais são aquelas que afetam o modelo todo o tempo, como por exemplo valores que estabelecem limitações no modelo. Já as variáveis de estado estão associadas às entidades, armazenando informações sobre cada uma para controlar o fluxo. Elas podem, por exemplo, separar as entidades por caminhos diferentes, de acordo com os atributos de cada. Os armazenadores de dados estatísticos permitem coletar informações sobre sistema durante a simulação, possibilitando inferências por parte do modelador (INGALLS, 2008).

Viana *et al.* (2014) argumentam que a SED é a abordagem ideal para representar sistemas que possuem redes de filas. Trata-se de uma abordagem altamente flexível onde quase tudo pode ser codificado e os modelos podem ser altamente detalhados, sendo que a maioria dos *softwares* utilizados possuem interfaces gráficas para facilitar a visualização dos processos em tempo real (VIANA *et al.*, 2014).

Schribber *et al.* (2014) definem como modelos de Simulação a Eventos Discretos aqueles onde as mudanças só podem ocorrer em determinados pontos no tempo (aleatórios ou não), chamados **eventos**. Uma definição proposta pelos autores para se compreender a SED é a visão de mundo transação-fluxo (*transaction-flow world view*), que consiste em um conjunto de unidades discretas que se locomovem ponto a ponto e disputam umas com as outras por recursos limitados. Vários sistemas se encaixam nessa analogia, como manufatura, serviços e saúde.

Segundo Shannon (1998), a SED consiste em modelar um sistema em particular através do fluxo de entidades que se movem através dele.

Jacob (2013) aponta um dos fatores-chave que diferenciam a SED de outras abordagens de simulação, que são as restrições que limitam em como o sistema pode sofrer mudanças ao longo do tempo. O programa de computador registra o estado do sistema ao longo do tempo. Por estado do sistema entende-se o estado de cada uma das variáveis que influenciam este sistema. A mudança de estado ocorre em intervalos de tempo denominados eventos, onde pelo menos uma das variáveis sofre uma mudança instantaneamente. Segundo Sharma (2015), na SED cada evento marca uma mudança no sistema, e nenhuma mudança pode ocorrer entre os eventos.

De acordo com Sharma (2015), um dos exemplos clássicos de Simulação a Eventos Discretos é a fila com um atendente. Neste exemplo, a população de clientes é infinita e a fila possui capacidade infinita. A frequência de chegada de clientes e o tempo de atendimento são determinados por distribuições de probabilidade fixas, e os clientes que chegam são atendidos na ordem de chegada, saindo do sistema após serem atendidos.

A qualquer momento durante a simulação deste modelo, o atendimento estará ocupado ou livre, e enquanto estiver ocupado a fila terá um número discreto de entidades aguardando e uma sendo atendida. Através deste modelo simples é possível se extrair diversas informações, como o tempo médio de espera na fila, o tempo médio de atendimento, o número médio de pessoas na fila e inclusive testar hipóteses sobre como a performance do sistema mudaria se fosse possível acelerar o atendimento ou aumentar o número de atendentes (SHARMA, 2015).

Ainda nesse mesmo exemplo de fila com um atendente, Ross (2006) aponta os momentos onde ocorreriam os eventos, que equivalem aos instantes onde ocorrem a chegada de mais um cliente ou quando um deles acaba de ser atendido. Apenas nestes momentos os valores das variáveis são alterados (número de clientes presentes no sistema, total de chegadas e total de atendimentos).

Uma das formas de se trabalhar com a Simulação a Eventos compreende três etapas: concepção ou formulação do modelo, implementação do modelo e análise dos resultados do modelo (CHWIF & MEDINA, 2010; MONTEVECHI *et al.* 2007). A primeira etapa consiste em compreender o sistema a ser simulado e seus objetivos, para se decidir seu escopo, hipóteses e nível de detalhamento, e é aí onde se define o modelo conceitual. Na segunda etapa o modelo conceitual é convertido em modelo computacional, através da utilização de alguma linguagem de simulação ou simulador comercial. Na última etapa, com o modelo computacional já

definido, ele é rodado para se analisar os resultados da simulação (CHWIF & MEDINA 2010). O passo a passo deste processo está representado no *framework* da Figura 1.

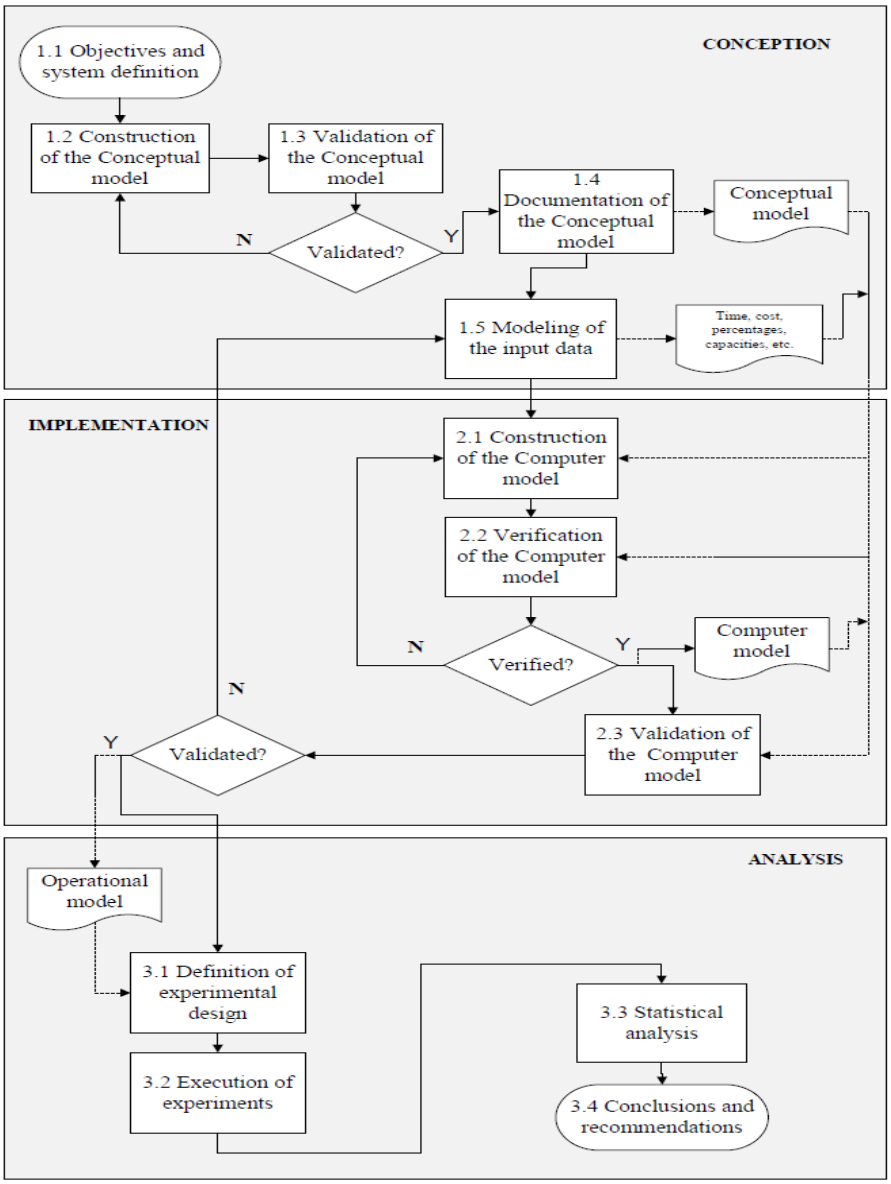


Figura 1 - Metodologia de Simulação
Fonte: Montevechi *et al.* (2007)

Na fase inicial do processo, o sistema a ser simulado deve ser analisado pelo modelador. As ideias a respeito do sistema estão inicialmente armazenadas de forma abstrata na mente deste, e devem ser registradas na forma de modelo, chamado modelo conceitual (LEAL *et al.* 2008). Este modelo serve de base para as etapas seguintes, e auxilia o modelador a definir quais elementos devem ser incluídos no modelo e consequentemente quais dados deverão ser coletados.

Para se registrar o modelo conceitual é necessário se utilizar alguma técnica de modelagem, e segundo Hernandez-Matias *et al.* (2008), não há somente um método de modelagem conceitual que possa modelar por completo um sistema complexo de manufatura. Entre as técnicas de modelagem conceitual podemos citar a ACD (*Activity Cicle Diagram*), que de acordo com Chwif & Medina (2010) consiste em uma forma de modelar as interações entre os elementos de um sistema, e o IDEF-SIM, proposto por Leal *et al.* (2008) que utiliza técnicas de modelagem já consagradas no contexto do BPM (*Business Process Modelling*), como o IDEF0 e IDEF3.

O IDEF-SIM (*Integrated Definition Methods – Simulation*) é técnica de modelagem conceitual proposta por Leal (2008). De acordo com o autor, ele foi criado a partir de três técnicas de modelagem: IDEF-0, IDEF-3 e Fluxograma. Os símbolos e lógicas foram adaptados e direcionados para as finalidades da simulação, como mostra o Quadro 1.

Quadro 1– Simbologia utilizada na técnica proposta IDEF-SIM

Elementos	Simbologia	Técnica de origem
Entidade		IDEF3 (modo descrição das transições)
Funções		IDEF0
Fluxo da entidade		IDEF0 e IDEF3
Recursos		IDEF0
Controles		IDEF0
Regras para fluxos paralelos e/ou alternativos	& Regra E X Regra OU O Regra E/OU	IDEF3
Movimentação		Fluxograma
Informação explicativa		IDEF0 e IDEF3
Fluxo de entrada no sistema modelado		
Ponto final do sistema		
Conexão com outra figura		

Fonte: Leal (2008)

Os processos de verificação e validação dos dados normalmente são executados antes de partir para a próxima etapa da simulação. De acordo com Sargent (2004), a verificação dos dados serve para conferir que o modelo computacional e sua implementação estejam corretos (em outras palavras aumentar a credibilidade do modelo). De acordo com Shannon (1998), o fato do modelo computacional compilar, executar e produzir números não significa necessariamente que que esses valores são representativos do sistema sendo modelado. Após o desenvolvimento do modelo, é preciso garantir que ele esteja operante do modo como o analista pretendia, e para isso é preciso verificá-lo e validá-lo. Um modelo é considerado válido se os valores de seus

outputs estão entre os limites aceitáveis (que são determinados antes da simulação) (SARGENT, 2004).

No que diz respeito ao modelo conceitual, este deve ser validado de forma subjetiva para ser implementado. Esta validação procura garantir que os elementos do sistema e suas relações estão representadas de forma precisa e normalmente é feita por pessoas que tenham conhecimento do sistema real junto aos modeladores. Sargent (2004) apresenta algumas metodologias para a validação subjetiva: se o time responsável pelo modelo for pequeno, ele pode ser validado primeiramente pelos modeladores e posteriormente pelos usuários da simulação. Do contrário pode-se adotar a abordagem de verificação e validação independente, que envia o modelo para ser validado junto a outra equipe de modelagem, não envolvida no projeto.

Com o modelo conceitual bem definido, ele servirá de base para a construção do modelo computacional, na fase de implementação. Através dos softwares de simulação o modelador pode representar as diversas lógicas e interações entre os elementos do sistema no modelo computacional. De acordo com Chwif e Medina (2010), a confecção dos modelos, que antes era feita através de programação em alguma linguagem de computador, como a Fortran e C, se torna extremamente simples com a utilização de um simulador, pois este possui uma interface gráfica que permite ao modelador construir seu modelo através da interligação dos blocos de construção presentes no programa (que podem representar diversos elementos comuns em modelos como filas, processos, chegada e saída de entidades, etc). Para Yucesan e Schruben (1998), a implementação do modelo consiste em traduzir as especificações do sistema para um formato executável em computador.

A última etapa consiste em rodar o modelo para se coletar os dados gerados com a simulação. Para Chwif e Medina (2010), a análise pode começar a partir do momento em que o modelo computacional se torna operacional (verificado e validado). Além dos valores numéricos gerados nas rodadas, a interface gráfica apresenta os processos ocorrendo simultaneamente com o tempo ajustável, auxiliando a realização de análises e servindo inclusive para apresentar o processo para quem estiver interessado em conhecê-lo de forma virtual. Essas etapas, contudo, não devem ser interpretadas como uma sequência linear, já que na prática podem ocorrer diversas iterações e realimentações no processo.

Alguns autores apontaram diversos pontos fracos do uso da Simulação a Eventos Discretos. Siebers *et al.* (2010) apontam a dificuldade de se representar de forma mais precisa o comportamento humano em um sistema. Elkosantini (2015) pontua que uma das causas para essa imprecisão na simulação do fator humano vem do fato de que a SED se utiliza frequentemente de dados determinísticos para representar o comportamento humano nos modelos computacionais.

Dubiel e Tsimhoni (2005) apontam algumas limitações da Simulação a Eventos Discretos na representação do movimento de pessoas: a primeira diz respeito ao movimento das entidades dentro do modelo, que é generalizado e simplificado, seguindo padrões pré-estabelecidos. Essa característica torna a modelagem do movimento humano não realista ao representar situações onde pessoas percorrem trajetos não previamente definidos. A segunda limitação envolve entidades que precisam tomar decisões em curtos períodos de tempo. Para fazer isso em um modelo SED seria necessário acrescentar diversos pontos de decisão durante o trajeto desta entidade.

Uma última limitação apontada por Dubiel e Tsimhoni (2005) se refere à impossibilidade de se programar o movimento das entidades individualmente. Este movimento só pode ser representado em pontos de decisão presentes no fluxo de todas as entidades. Os autores pontuam que essas limitações podem ser contornadas, até certo ponto, pelo uso de funções e por programação, mas isso não muda o fato da SED não ser adequada para modelar o movimento humano.

Mesmo com essas dificuldades em se representar o fator humano, a Simulação a Eventos Discretos continua sendo a abordagem mais utilizada para resolver a maioria dos problemas de simulação em Pesquisa Operacional, por ser o tipo mais difundido e conhecido de simulação (SIEBERS *et al.*, 2010). Uma forma de contornar suas limitações é através da Simulação Híbrida. A abordagem baseada em agentes permite a integração com a Simulação a Eventos Discretos, possibilitando a inserção de agentes em um modelo SED para representar o fator humano.

2.3. Simulação baseada em agentes

Segundo Macal e North (2013), a Simulação Baseada em Agentes (também conhecida pela sigla SBA, do original ABS - *agent-based simulation* ou ABMS – *agent-based modeling and simulation*) é uma abordagem de modelagem e simulação de sistemas caracterizada por possuir

“agentes” que são autônomos, individuais e que interagem entre si. Bonabeau (2002) pontua que ela se trata mais de uma forma de abordagem do que de uma tecnologia em si. Através desta abordagem é possível simular de outras formas os comportamentos individuais e como estes interagem com o meio (MACAL & NORTH, 2013).

Na definição de Shannon (1975), a Simulação Baseada em Agentes é o processo de desenvolver um modelo baseado em agentes e conduzir experimentos neste com o propósito de melhor compreender o sistema real e/ou desenvolver estratégias para operá-lo. Para Sanchez e Lucas (2002), modelos baseados em agentes são aqueles onde diversas entidades individuais interagem e respondem estocasticamente às condições do ambiente onde se encontram, imitando sistemas complexos de larga escala.

De acordo com Macal (2016), a SBA é caracterizada por possuir uma ampla gama de utilizações, e possui características que complementam outros tipos de simulação. Seu foco principal são os agentes, seus comportamentos e os fatores que os influenciam. A SBA busca a forma de melhor descrever o modo com o qual os agentes se comportam e interagem, para entender como funcionam as organizações e estruturas sociais.

A SBA pode ser considerada uma linha de estudo interdisciplinar, seu uso surge em outras disciplinas como ciências sociais, geografia e economia. Tais modelos já foram utilizados para identificar padrões sociais, e podem inclusive ser utilizados para auxiliar previsões econômicas (MACAL, 2016). Segundo North e Macal (2007), a Simulação Baseada em Agentes tem conexões com diversas outras áreas, como sistemas adaptativos complexos, ciência da complexidade e ciência de sistemas. Historicamente os sistemas adaptativos complexos surgiram de estudos sobre adaptação e emergência em sistemas biológicos, que possuíam elementos capazes de adaptar a mudanças no ambiente e se organizar para melhorar as chances de sobrevivência.

Segundo apontam Siebers *et al.* (2010), no ano 2000 a SBA foi proposta por John Sterman como uma ferramenta que ofereceria a oportunidade de desenvolver os modelos e aplicações vigentes da simulação. Durante os 10 anos seguintes foi possível notar um grande avanço no número de aplicações da SBA, como aplicações em tráfico e transportes, cadeias de suprimentos, mercados financeiros, políticas sociais e uso da energia (NORTH & MACAL 2007; SIEBERS *et al.*, 2010). De acordo com Siebers *et al.* (2010), através da SBA é possível incluir modelos descritivos de como as pessoas tomam decisões em uma cadeia de suprimentos, e quais os efeitos dessas decisões na cadeia. Chan *et al.* (2010) afirmam que a popularidade

crescente da SBA se deve em grande parte à sua capacidade de representar a grande complexidade de sistemas reais, muitas vezes compostos por um grande número de agentes que são autônomos, direcionados por objetivos e adaptativos.

Para Siebers *et al.* (2010) até o ano de 2010 existiam poucas evidências de uso da SBA na Pesquisa Operacional, e poucas publicações em *journals* relacionados à mesma. Em contrapartida, existe um grande número de trabalho baseados em SBA em publicações de outras disciplinas, como Ciências Sociais, Ciências da Computação e Economia. Os autores pontuaram que a SED ainda era abordagem mais utilizada na simulação, contudo esta seria substituída um dia, já que o número de pessoas utilizando a SBA estava aumentando, buscando a resolução de problemas de Pesquisa Operacional que não haviam sido corretamente solucionados com outras abordagens (SIEBERS *et al.*, 2010).

Segundo North e Macal (2007), os sistemas que apresentam diversos componentes individuais que contribuem para o todo do sistema podem ser denominados sistemas adaptativos complexos (no original CAS – *Complex Adaptative Systems*). Os "agentes" são os componentes tomadores de decisão em um sistema adaptativo complexo. Agentes possuem regras e padrões de comportamento que os permitem receber e processar informações para alterar o ambiente externo (através de adaptação ou aprendizagem). Nesse sentido, qualquer coisa que faça escolhas em um negócio pode ser visto como um agente: executivos, gerentes, organizações, sistemas computacionais, etc. Diferentemente de objetos, agentes podem tomar decisões autônomas (por exemplo, responder a seu ambiente) e demonstrar comportamento proativo (suas ações podem depender de suas regras internas, as quais podem se modificar durante a simulação) (SIEBERS *et al.*, 2010).

A Simulação Baseada em Agentes, em comparação com a abordagem de eventos discretos possui uma forma diferente de se representar os sistemas. Na Simulação a Eventos Discretos, os processos são representados de forma top-down, ou "de cima pra" baixo, que significa que foco está em modelar o todo do sistema em si e não as suas entidades. Desse modo, o comportamento macro do sistema é modelado e suas entidades possuem características passivas, respondendo às regras estabelecidas (SIEBERS *et al.*, 2010).

Já na Simulação Baseada em Agentes o foco está na modelagem de cada um dos agentes que compõem o sistema, onde eles possuem suas próprias regras e podem tomar diferentes caminhos durante a simulação. Esse tipo de abordagem é chamado bottom-up, ou "de baixo pra

cima", onde o comportamento macro do sistema não é modelado, e emerge das ações e interações dos agentes que o compõem (SIEBERS *et al.*, 2010).

O conceito de agentes dentro de um modelo SBA não é simples de ser definido. Para Macal e North (2013), não existe um consenso universal sobre a definição do termo "agente" nem para o que seria uma modelagem baseada em agentes. Os agentes podem, porém, ser caracterizados como entidades individuais e autônomas capazes de tomar decisões, onde cada uma delas segue suas próprias regras, podendo reagir aos eventos ao redor de si, inclusive interagindo com outros agentes no mesmo ambiente de simulação (BONABEAU, 2002; MACAL & NORTH, 2009; MACAL & NORTH, 2013).

Entre as definições de diversos autores para "agente", pode-se destacar: entidades inteligentes, proativas e autônomas (CHAN *et al.*, 2010); Elementos discretos com aspectos e regras que governam sua tomada de decisões, portanto são capazes de tomar decisões de forma individual (SAMUELSON & MACAL, 2006); Elementos caracterizados por cooperação, inteligência, autonomia e adaptação (LEITÃO, 2009); Entidades discretas de um modelo de simulação capazes de imitar o comportamento das entidades do sistema real (SIEBERS *et al.* 2010).

Os agentes interagem entre si em um ambiente, que segundo Chan *et al.* (2010) pode possuir diversas formas, ser um ambiente 2D ou 3D, ou mesmo baseado em um mapa real gerado por GIS (*Geographic Information System*). Chan *et al.* (2010) apontam que nas áreas de inteligência artificial e ciências sociais é utilizada uma noção mais humanizada de agentes, onde estes podem perceber, raciocinar, reagir, memorizar e tomar iniciativas baseado em seu conhecimento, experiências passadas e regras pré-definidas. Nesse sentido um agente pode ser pensado como um avatar utilizado para representar um ser humano, animal ou objeto inteligente.

Para North e Macal (2007), na SBA usa-se atributos para representar dados de cada agente, como idade, sexo, salário e histórico (para o caso de agentes humanos) ou recursos, tamanho e estratégias para o caso de empresas, por exemplo. Atributos associados a cada agente são dinâmicos e podem variar durante a simulação, em razão das experiências vividas por cada um. Já os comportamentos dos agentes dizem respeito às lógicas usadas por cada agente na tomada de decisões (como cada agente responde a eventos que é submetido) (NORTH & MACAL, 2007). Os comportamentos são descritos por algoritmos de variada complexidade, que podem variar de simples algoritmos de *if-then-else* até representações estocásticas abstratas altamente complexas de mapeamentos de estímulo-resposta (BRAILSFORD, 2014).

Dentro dos comportamentos podem também existir “regras para mudar as regras” que são lógicas usadas para adaptar as regras de tomada de decisões do agente durante a simulação (MACAL & NORTH, 2007; CASTI, 1998 apud MACAL & NORTH, 2007). Com o passar do tempo os agentes podem avaliar quão bem suas regras se saíram, e a partir disso ele pode por exemplo alterar a probabilidade de tomar cada atitude. Tomando como exemplo agentes que representam motoristas, estes podem passar a evitar estacionar seus carros em determinados horários se não encontraram vagas nesses horários em dias anteriores (SANCHEZ & LUCAS, 2002).

Dadas as diversas atribuições para os agentes, que podem variar de acordo com cada modelo de simulação, Macal (2016) apresenta quatro definições para SBA. O autor salienta que não há precisão quanto às definições apresentadas por ele, contudo elas auxiliam a distinguir a Simulação Baseada em Agentes de outras abordagens de simulação, além de distinguir o nível de autonomia e aprendizagem dos agentes em um modelo SBA. Os quatro tipos propostos por Macal (2016) possuem uma ordem crescente de autonomia e complexidade do agente, como se segue:

- SBA individual: um modelo que possui entidades individuais com comportamento autônomo e que se diferem umas das outras. Nessa definição o agente não pode agir de forma reativa a eventos externos, apenas segundo a sua lógica interna.
- SBA autônomo: os agentes possuem um comportamento interno que os permitem ter autonomia, e também podem reagir a eventos no ambiente. Eles possuem um estado interno de onde se baseiam seus comportamentos e esse estado muda durante a simulação, porém aí ainda não se incluem as interações entre os agentes.
- SBA interativo: nessa definição os agentes podem interagir entre si e com o ambiente, podendo alterar seus comportamentos em resposta a mudanças aleatórias no sistema ou ao comportamento de outros agentes, por exemplo.
- SBA adaptativo: nesta definição os agentes podem aprender, ajustando seus comportamentos baseando-se nas próprias experiências (ocorridas durante a rodada da simulação). Cada aprendizado faz com que o agente acrescente informações ao seu estado, alterando-o com o tempo, podendo assim reagir aos mesmos estímulos em uma mesma simulação de forma diferentes.

A SBA possui alguns elementos principais, segundo Sakurada e Miyake (2009), que são:

-Estados: cada estado representa um conjunto particular de comportamentos de um agente que responde a eventos externos. Estados podem ser simples ou compostos (estado que contém outros estados). Como exemplos de estados pode-se citar, para um agente que representa um operador, estados como "trabalhando", "descanso", "aguardando", entre outros;

-**Transição:** determina as mudanças de um estado para outro, e é ativada quando determinadas condições são satisfeitas. As condições que governam a mudança de estados são pré-definidas no modelo e possuem diversas formas como taxas, limites de tempo, troca de mensagens entre agentes, ou mesmo condições *booleanas* ("sim" ou "não");

-Diagrama de Estados: composto pelos diversos estados e transições entre eles, modelando a dinâmica dos agentes. Permite definir comportamentos mais sofisticados que não são possíveis de ser representados pela SED ou Dinâmica de Sistemas. O diagrama define como os diferentes estados podem ser ativados através dos eventos que ocorrem no sistema;

-**Comunicação/ interação entre agentes:** as mensagens são pequenos conjuntos de informação que são passados pelos agentes, podendo representar informações, comandos, sinais, etc. Através delas se dão as interações dos agentes, sendo que muitas vezes o recebimento das mensagens é utilizado como condição para a transição de estados do agente.

Para North e Macal (2007), a SBA é fundada sobre a noção de que o todo de vários sistemas ou organizações é mais do que a soma das partes que os constituem. Para representar estes sistemas, eles devem ser compreendidos como uma coleção de componentes que interagem entre si, cada um possuindo determinadas regras e responsabilidades. Alguns desses componentes podem ser mais influentes que os outros, mas nenhum deles determina o funcionamento do sistema sozinho. As reações onde os resultados finais são mais do que a soma dos resultados individuais de cada elemento são denominadas comportamentos emergentes. Segundo Bonabeau (2002), na SBA as propriedades do sistema emergem a partir das interações dos agentes. Brailsford (2014) aponta que a emergência é uma propriedade do sistema e não dos agentes em si, já que só surge perante a interação destes.

O uso da Simulação Baseada em Agentes, apesar de não muito difundido na Pesquisa Operacional, apresenta diversas vantagens. Macal e North (2009) afirmam que a SBA é o novo paradigma de modelagem para futuros projetos de sistema e manufatura, pois permite a inserção de alto nível de individualização, autonomia e interatividade dos agentes. Chan *et al.* (2010) apontam que a SBA possui a capacidade de representar a grande complexidade dos sistemas reais, muitas vezes compostos por um grande número de agentes que são autônomos,

direcionados por objetivos e adaptativos. Para Siebers *et al.* (2010), através da SBA é possível incluir modelos descritivos de como as pessoas tomam decisões em uma cadeia de suprimentos, e quais os efeitos dessas decisões na cadeia. Fakhimi *et al.* (2014) apontam que comparada a outras técnicas de modelagem e simulação, a SBA pode prover uma visão mais realista de sistema complexos com muitas interdependências.

Segundo Chan *et al.* (2010), o que torna a SBA poderosa não é a inteligência dos agentes, mas sim a simulação das interações destes que cria oportunidades para o melhor entendimento de sua natureza. Ela permite prever efeitos emergentes de pequenas interações locais, identificar e explicar comportamentos benéficos e prejudiciais e, principalmente, testar mecanismos que possibilitem encorajar tais comportamentos benéficos e desencorajar os prejudiciais. Macal e North (2009) afirmam que a SBA é o novo paradigma de modelagem para futuros projetos de sistema e manufatura, pois permite a inserção de alto nível de individualização, autonomia e interatividade dos agentes.

2.3.1. Comparações entre a SED e a SBA

No meio acadêmico, dentro da área de Pesquisa Operacional, para muitos pesquisadores a palavra "simulação" é tratada como Simulação a Eventos Discretos (BRAILSFORD, 2014). É visível na literatura uma falta de concordância de onde deve se usar a Simulação a Eventos Discretos e onde deve se usar a Simulação Baseada em Agentes. Siebers *et al.* (2010) apontam que segundo a prática da boa modelagem é recomendado que se responda as perguntas que a pesquisa pretende responder primeiro para identificar quais métodos são os mais aplicáveis depois.

Em 2010 Siebers *et al.* Publicaram um artigo intitulado "*Discrete-event simulation is dead, long live agent-based simulation!"* onde se propunha substituir aos poucos o uso da SED pela SBA na Pesquisa Operacional. Na opinião dos autores o pouco uso da SBA na comunidade de Pesquisa Operacional se deve à grande disponibilidade de *softwares* de simulação a eventos discretos e a experiência dos pesquisadores com eles. Nesse sentido as pessoas tendem a utilizar ferramentas que dominam até onde isso seja possível. Mas os autores acreditam que no futuro haja uma tendência ao uso de ferramentas baseadas em agentes devido à crescente necessidade de utilizá-las na resolução de problemas que não podem ser abordados pela SED ou pela Dinâmica de Sistemas, como por exemplo modelos de infecção ou pandemias (SIEBERS *et al.* 2010).

Já em 2014 foi publicado por Brailsford um artigo intitulado "*Discrete-event simulation is still alive and kicking!*", onde a autora propôs que a SBA não será amplamente aceita na comunidade de Pesquisa Operacional até que ela tenha sido demonstrada útil para a solução de problemas práticos, com uma exceção à regra no que diz respeito aos modelos de movimentação de pessoas, onde a SBA se demonstrou mais eficaz. Brailsford (2014) demonstrou a possibilidade de se usar a SED para solucionar diversos problemas propostos em Siebers *et al.* (2010).

Alguns autores apontam vantagens de se substituir a SED pela SBA. Siebers *et al.* (2010, p.204) afirmam que a "SBA vai nos ajudar a compreender melhor sistemas reais onde a representação ou modelagem de muitos indivíduos é importante e onde esses indivíduos tenham comportamento autônomo". Segundo Siebers *et al.* (2010), as pessoas tendem a utilizar as ferramentas com as quais estejam mais familiarizadas, o que justifica o pouco número de pesquisadores que substituíram a SED pela SBA.

Baines *et al.* (2005) afirmam que é importante incluir particularidades do fator humano nos modelos de simulação de forma a melhorar sua precisão, como comportamento e desempenho. Portanto existem determinadas brechas na SED que devem ser preenchidas para que o os aspectos humanos sejam representados de forma mais próxima da realidade. Segundo Brailsford *et al.* (2012), a diferença entre os resultados dos sofisticados e complexos modelos de simulação e os resultados dos sistemas reais é um problema comum, e é causada pelo fato de os modelos de simulação não consideram o comportamento humano.

Swain (2007) aponta que operações industriais e militares raramente operam de uma forma previsível, sendo comum o surgimento de situações emergentes geradas pela interação de seus agentes. Num sistema real, trabalhadores e equipamentos operam de formas bastante diferentes uns dos outros, e poderiam ser melhor representados em um modelo de simulação se fossem programados de forma individual e autônoma. Contudo, poucos modelos da SED permitem considerar que na ocorrência de um problema as entidades podem se comportar de forma individual, interagindo com as outras e passando por um processo de aprendizagem (mudança de estado).

Sanchez e Lucas (2002) apontam um exemplo de aplicação da SBA na representação do movimento de evacuação de pessoas em um estádio. Tentar criar um script que represente o movimento de forma global pode ser muito trabalhoso e difícil de modificar. O uso de agentes permite representar todo o fluxo de pessoas associando a cada uma comportamentos simples como: buscar a saída, fazer curvas para contornar obstáculos, ficar próximo de família e amigos,

entre outros. A partir do momento que as regras foram codificadas, é fácil reproduzir tais regras para um número qualquer de agentes.

Para Brailsford (2014), a Simulação a Eventos Discretos também possui suas entidades, que de forma similar aos agentes possuem atributos que determinam seu comportamento e suas interações dentro do sistema. Contudo, uma diferença crucial entre as duas abordagens foi apontada em Siebers *et al.* (2010), e diz respeito ao objetivo final destas abordagens, já que algumas aplicações clássicas de SBA tinham como objetivo descobrir certos fenômenos na natureza, em oposição à SED que tem por objetivo encontrar soluções para problemas práticos. Para Chan *et al.* (2010), o que diferencia SED da SBA é a flexibilidade de cada uma e a eficiência em modelar diferentes tipos de sistema. Enquanto a SBA é útil para modelar sistemas com entidades que interagem constantemente umas com as outras, a SED possui como foco simular os eventos e seus relacionamentos em uma dinâmica de eventos discretos.

Para Mittal (2016), ações de agentes podem ser modeladas como eventos em um modelo SED, porém isso pode aumentar exponencialmente o tamanho do modelo, tornando-o ineficiente e difícil de se analisar. Já sistemas de formação de filas, de qualquer complexidade, são difíceis de se modelar em um ambiente baseado em agentes pois requerem muitos recursos computacionais. A literatura aponta que, nos últimos anos, diversos autores têm voltado sua atenção para a simulação híbrida, o que pode indicar um aumento do uso de modelos SH nos próximos anos (ELDABI *et al.*, 2018).

O Quadro 2 apresenta algumas diferenças entre as duas abordagens. Pelas definições encontradas nela, Siebers *et al.* (2010) concluem que não existem modelos puramente baseados em agentes na Pesquisa Operacional. O que se usa são modelos que possuem na sua estrutura elementos de SED para representar o fluxo de processos e a inserção de agentes (substituindo os “recursos” da SED) para representar as entidades capazes de tomar decisões.

Quadro 2- Diferenças entre modelos SED e SBA

Modelos SED	Modelos SBA
Orientado por processos (*top-down*), foco no sistema em detalhes e não nas entidades	Foco nos indivíduos (*bottom-up*), na modelagem das entidades e interações entre elas
Controle centralizado	Controle descentralizado
Entidades passivas que sofrem ação dos processos do sistema; Sua tomada de decisões é modelada como parte do sistema	Entidades ativas que podem tomar iniciativa das suas ações; A inteligência é representada dentro de cada indivíduo
Filas são um elemento essencial	Não há conceito de filas
Fluxo de entidades pelo sistema, modela-se o comportamento macro do mesmo	O comportamento macro não é modelado, este emerge das micro-decisões de cada agente e das interações entre eles
Dados de entrada são normalmente baseados em valores coletados (objetivos)	Dados de entrada são normalmente subjetivos ou baseados em teorias

Fonte: adaptado de Siebers *et al.* 2005.

2.4. Simulação híbrida

Segundo Eldabi *et al.* (2016), a Simulação Híbrida (SH) pode ter vários tipos. Existem modelos híbridos compostos por um ou mais modelos de simulação, como combinando simulação a eventos discretos com agentes ou dinâmica de sistemas e mesmo combinando simulação com técnicas analíticas de outras áreas da Pesquisa Operacional. Dada a crescente complexidade do mundo moderno, a simulação híbrida tem se tornado um importante campo de estudo dentro da área de Modelagem e Simulação (ELDABI *et al.,* 2016). Green *et al.* (2017) apontam que, até recentemente, a simulação a eventos discretos e a simulação baseada em agentes eram usadas geralmente em separado, sendo a SED mais usada para avaliar a performance de sistemas de manufatura e a SBA para modelar o comportamento organizacional.

Para Mustafee *et al*. (2017) os benefícios de se utilizar qualquer método, ou combinação de métodos, na modelagem é permitir que o modelo atinja seu objetivo mais eficientemente, rapidamente, mais facilmente e com menor custo. Segundo Mittal (2016), a simulação híbrida integra os benefícios e capacidades de ambos os tipos de simulação, possibilitando ao modelador capturar os elementos fundamentais e o comportamento do sistema.

Mustafee *et al.* (2017) apontam que o significado e uso do termo "híbrido" variam consideravelmente. Para modelagem e simulação este termo é comumente usado ao se referir a modelos que combinam mais de uma técnica (como SED, SBA e Dinâmica de Sistemas). Segundo os autores, a crescente disponibilidade de softwares que possibilitam o uso de mais de uma técnica no mesmo modelo foi uma das responsáveis pelo aumento do interesse na SH. Brailsford define, em Mustafee *et al.* (2017) um modelo híbrido como sendo aquele que, quando implementado em computador utiliza mais de um paradigma de simulação.

De acordo com Viana *et al.* (2014), combinar diferentes modelos em um framework híbrido representando partes de um sistema maior não é algo novo. Kittipittayakorn e Ying (2016) usaram um modelo híbrido combinando SED e SBA para simular o processo de formação de filas no setor ortopédico de um hospital. A Simulação a Eventos Discretos foi utilizada para representar os processos e as filas e os agentes foram inseridos para representar o fator humano, que foram necessários para melhor representar o comportamento dos pacientes e trabalhadores do hospital. Porém, como afirma Mittal (2016), a literatura sugere que a SH é uma tendência crescente, porém ainda se encontra nos estágios iniciais de seu desenvolvimento.

Dubiel e Tsimhoni (2005) ressaltam a importância de se utilizar a Simulação Híbrida, combinando a SED com a SBA, para possibilitar a modelagem de determinadas particularidades de um sistema, como padrões de decisão e movimentos de indivíduos. Os autores usaram agentes para melhor representar o fluxo de pessoas em um parque temático, tarefa que, segundo os eles, seria inviável de se cumprir apenas utilizando-se da SED. Viana *et al*. (2018) desenvolveram um modelo híbrido que teve como objetivo otimizar a alocação de pessoal em um hospital para diminuir as filas do setor de maternidade. Neste modelo, os processos principais da clínica são representados pela SED, enquanto as pacientes e alguns elementos mais complexos dentro do sistema são construídos como agentes.

Green *et al.* (2017) desenvolveram um modelo híbrido (SED e SBA) para investigar como a predisposição e as tendências altruísticas dos trabalhadores poderiam influenciar na sua tendência a ajudar os colegas de trabalho. Os autores se basearam em dados de um experimento com pessoas reais, e transformaram este experimento em um modelo de simulação. Neste experimento os operadores atuam em três postos de trabalho processando caixas, e ao terminar eles podem sair ou ficar para ajudar seus colegas. No modelo de simulação o fluxo de processos e o trabalho dos operadores foram representados pela SED, e os dados eram então enviados a cada agente, que segundo suas características decidia ou não se iria ajudar os colegas.

Mustafee *et al.* (2017) apontam que o significado do termo "simulação híbrida" pode variar, e outros termos já foram utilizados para representar a combinação de diferentes técnicas de simulação num mesmo modelo, como simulação multi-método, multi-paradigma, misturada e combinada. Os autores se baseiam nas definições de Mingers e Blocklesby (1997) para definir quatro formas de hibridização: multi-paradigma, multi-métodos, multi-técnica e multi-ferramentas. "Paradigmas" podem ser divididos em qualitativos ou quantitativos; "Metodologia" está associada às suposições usadas para modelar, distinguindo modelos discretos de contínuos; O termo "técnica" está associado à diferentes abordagens de simulação,

como SED, SBA ou Dinâmica de Sistemas; Finalmente, "ferramentas" são os *softwares* ou pacotes de simulação utilizados, podendo um modelo híbrido utilizar mais de um.

2.5. Complexidade

Segundo ElMaraghy *et al.* (2012), o significado do termo "complexidade" é ambíguo, não existindo uma definição precisa para ele. Para os autores, seguindo uma definição mais próxima do dicionário Oxford, um sistema pode ser considerado mais complexo na medida em que mais elementos e relações entre esses elementos existirem. Além disso, na falta de um conceito geral cada área das ciências e engenharia define e enxerga a complexidade de formas diferentes.

Entre as diversas definições e conceitos relacionados à complexidade, podem ser citados: aspecto cognitivo, número de elementos e as relações entre eles, incertezas, tempo de execução de um modelo, dificuldade de programar, entre outras (BROOKS & TOBIAS, 1996, CHWIF *et al.*, 2000, ELMARAGHY *et al.*, 2012; POPOVICS & MONOSTORI, 2016; SARJOUGHIAN, 2017).

Simon (1962) apresenta a definição de sistemas complexos como sendo aqueles formados a partir de um grande número de partes que interagem de uma forma não simples, onde o todo é mais que a soma das partes, e não é trivial compreender o comportamento do todo a partir do comportamento de cada parte. Para o autor, a complexidade frequentemente assume a forma de hierarquia, que ocorre quando um sistema é composto por subsistemas inter-relacionados, que por sua vez são compostos de outros subsistemas, até se chegar nos subsistemas elementares.

Na Física costuma-se usar o conceito de "partícula elementar", apesar do fato de que cada nova descoberta aponta maiores subdivisões dessas partículas antes consideradas indivisíveis. Outras áreas do conhecimento adotam como subsistemas elementares objetos de estudo que compõem sistemas maiores, como células, proteínas ou aminoácidos, e até mesmo, para a astronomia, estrelas e galáxias inteiras (SIMON, 1962). Morr (2014) aponta que a emergência da complexidade está geralmente relacionada a formas de comportamento coletivo guiadas por fortes interações, tais como as que ocorrem entre as partículas em movimento de condutores.

Edmonds (1998) propõe uma definição para complexidade, onde esta seria uma propriedade da expressão linguística que dificulta a formulação do comportamento de um todo mesmo quando se tem informação completa a respeito dos menores componentes e suas inter-relações. O autor divide a complexidade entre aquela que seria a complexidade "do sistema" contra aquela que seria a complexidade "do observador", relacionada ao fato de que nós humanos, em nossas

interações, não percebemos a complexidade real de um sistema, mas sim uma redução da mesma, já que nós apenas conseguimos lidar com uma complexidade reduzida (EDMONDS, 1998).

O conceito de complexidade pode ser associado, segundo Edmonds (1998), a 5 idéias. A primeira seria o tamanho, normalmente medido de forma casual, mas associado ao número de partes e inter-relações entre elas. A segunda seria a ignorância, ou a falta de entendimento do ponto de vista do observador ao se deparar com um sistema. Um terceiro conceito seria o da descrição mínima, também chamada complexidade de Kolmogorov (CORNACCHIO, 1977, *apud* EDMONDS 1998). Este conceito mede a complexidade de uma linguagem pelo tamanho mínimo possível que uma linguagem pode ser descrita. Os dois últimos conceitos que segundo Edmonds (1998) podem ser associados à complexidade são a variedade e a ordem (e desordem) aparente dos elementos compondo o todo.

2.5.1. Complexidade em simulação e modelagem de sistemas

Segundo Ahmed *et al.* (2016), a simulação é o negócio de reduzir a complexidade do mundo real. Na literatura de modelagem e simulação, recomenda-se que o modelador mantenha seus modelos pequenos e simples, porém isso não vem ocorrendo na prática, onde o que se observa é que o tamanho e a complexidade dos modelos de simulação vem aumentado com o tempo (CHWIF *et al.* 2000; AHMED *et al.*, 2016;). Sarjoughian (2017) defende que o objetivo básico da modelagem de sistemas é reduzir sua escala e complexidade ao mínimo possível, e a modelagem é a principal barreira que afeta a simulação, a verificação e a validação.

Ahmed *et al.* (2016) apontam que tamanho e complexidade são dois dos mais importantes fatores que afetam a prática da modelagem e simulação. Chwif *et al*. (2000) apontam que a complexidade dos modelos de simulação tem impacto não só no processamento computacional, mas em outros aspectos como a gestão de projetos de simulação, comunicação, identificação das restrições, entre outros. Os autores também propõem que se busque formas de reduzir a complexidade dos modelos ao invés de se tentar lidar com modelos muito complexos.

Sarjoughian (2017) diferencia os conceitos de *escala* e *complexidade*. Para o autor, escala se refere ao tamanho de um modelo, e está associada ao número de partes que o mesmo contém. Normalmente é associada a valores quantitativos, mas em certos casos (onde for muito trabalhoso ou impossível contar o número de elementos) ela pode ser medida de forma qualitativa. Já a complexidade se refere a uma ordem que regula o sistema, com contornos bem definidos, que se reduz à medida em que o sistema se torna menos complicado. Esta pode ser

alterada quando se modifica os tipos de componentes e suas relações, além da diminuição da escala.

Yucesan e Schruben (1998) definiram complexidade de modelos de simulação como a medida que reflete os requerimentos impostos pelo modelo aos recursos computacionais. Os autores apontam três desses tipos de requerimentos: espaço que o modelo exige no computador, o tempo necessário no desenvolvimento do modelo e o tempo e esforço envolvido em construir, testar, implementar, modificar, manter e comunicar o modelo. Yucesan e Schruben (1998) afirmam que a complexidade está normalmente associada às características estruturais do modelo. Para uma melhor definição do termo *modelo* dentro do contexto de modelagem e simulação, os autores definem modelo como um sistema usado como substituto de outro sistema.

ElMaraghy *et al.* (2012) diferem sistemas complicados de sistemas complexos. Para os autores, um sistema complicado seria aquele que possui muitas partes, o que o tornaria mais difícil de se entender, enquanto sistema complexo seria aquele que pode gerar incertezas em seu desenvolvimento ou intrinsecamente em seu *design*. Popovics e Monostori (2016) afirmam que, geralmente, a complexidade está vinculada ao conceito de sistema, composto por componentes interdependentes que interagem entre si. Assim, aumenta-se a complexidade de um dado sistema à medida em que há um crescimento no número de componentes e na quantidade de conexões entre eles.

ElMaraghy *et al.* (2012) aponta que a complexidade computacional pode ser medida pela quantidade de recursos computacionais necessários para se atingir determinado objetivo em um modelo. Para os autores, a forma mais eficaz de se diminuir a complexidade de um modelo é reduzindo o número de elementos. Com relação à mensuração da complexidade de um modelo de simulação, Chwif *et al.* (2000) apontam que não é possível se obter uma única e absoluta medida de complexidade de um sistema, porém uma comparação de modelos alternativos em diferentes circunstâncias é possível.

Segundo Yucesan e Schruben (1998), a medição da complexidade pode ser útil para classificar diferentes modelos e se obter dados para serem usados em pesquisas de metodologias de simulação. Os autores também afirmam que uma importante aplicação da medição da complexidade de modelos seria ao comparar e avaliar diferentes técnicas de modelagem e análise (YUCESAN & SCHRUBEN, 1998).

2.6. Flow shop

Também chamado de *layout* linear, o termo "*flow shop"* se refere a uma linha de produção onde todos os processos seguem uma estrutura linear. Para Hordones *et al.* (2016), os problemas de programação de tarefas em sistemas de produção são tipicamente classificados em função do fluxo das operações nas máquinas. Neste sentido, Fuchigami e Rangel (2015) definem *flow shop* como um sistema produtivo onde as tarefas seguem o mesmo fluxo pelas máquinas na mesma ordem, sendo que essas tarefas podem ou não sofrer algum tipo de permutação.

Segundo Bultmann *et al.* (2018), em um modelo *flow shop* clássico a transferência de tarefas é assíncrona, assim uma tarefa é transferida para a próxima máquina assim que seu processamento terminar na máquina atual e se inicia logo que a máquina seguinte esteja desocupada. Slack *et al.* (2002) apresentam alguns tipos de arranjos físicos: posicional, por processo, celular e por produto. Dentro dessas definições, o *flow shop* pode ser considerado um arranjo físico por produto, onde os agentes transformados seguem um fluxo regular, de fácil interpretação e controle.

Neste tipo de arranjo a sequência de máquinas e operadores (ou demais recursos transformadores) deve ser adaptada da forma mais eficiente possível (dando origem a diversos problemas de otimização de *scheduling*). Para Moura (2008), este tipo de *layout* é mais indicado para instalações que produzem um número reduzido de itens em grande quantidade. No arranjo físico por produto, segundo Corrêa e Corrêa (2005), a lógica usada para organizar a posição relativa dos recursos corresponde à sequência de etapas que agregam valor ao produto, sendo conveniente quando a produção é caracterizada por um pequeno número de produtos diferentes e um alto volume de produção.

Segundo Krajewski *et al.* (2009), um arranjo físico por produto levanta determinadas questões de gerenciamento como o balanceamento de linha. Nesse arranjo as diferentes estações de trabalho não podem operar independentemente umas das outras, e a linha será tão rápida quanto a sua estação de trabalho mais lenta (KRAJEWSKI *et al.*, 2009). A Figura 2, baseada em Slack *et al.* (2002), mostra os tipos principais de arranjo físico e quanto estes devem ser aplicados, com relação à variedade e ao fluxo de produção. Percebe-se que o arranjo por produto, ou *flow shop* é melhor utilizado para linhas com baixa variedade e alto volume de produção.

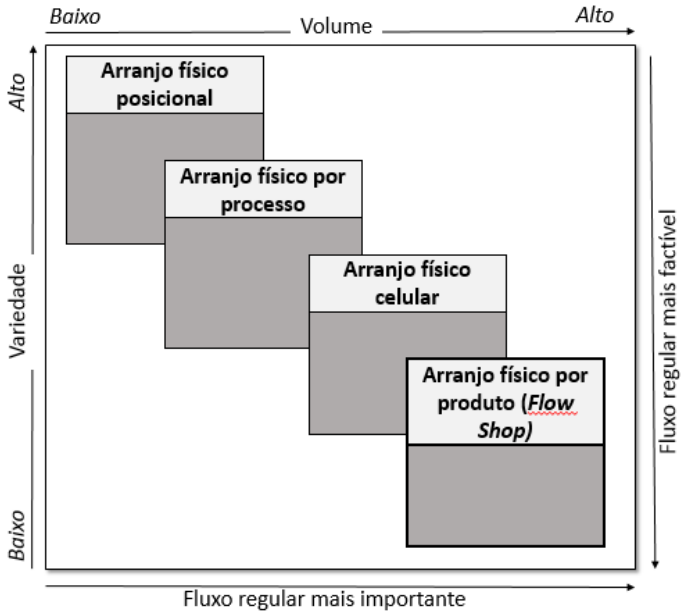


Figura 2 - Posição do processo no contínuo volume-variedade
Fonte: adaptado de Slack *et al.* (2002).

3. METODOLOGIA

3.1. Objetivos e métodos

Este estudo busca aprofundar a discussão sobre a utilização da simulação baseada em agentes no contexto da pesquisa operacional, onde ainda existe uma discussão sobre quando ela deve ou não ser usada, evidenciada em alguns artigos, em especial em Siebers *et al.* (2010) e Brailsford *et al.* (2014). Para isto, foi adotado o método do estudo de caso, aplicado na avaliação da complexidade de alguns pares de modelos de simulação, sendo cada par composto por um modelo construído a partir da SED e outro a partir da SH usando elementos de eventos discretos com a inserção de agentes.

O objetivo geral deste estudo é avaliar se a introdução de agentes na Simulação a Eventos Discretos pode contribuir na redução da complexidade do modelo computacional. Mas para se medir a complexidade dos modelos foi necessário desenvolver um método próprio, já que a literatura até então não fornecia nenhum modelo, passo-a-passo ou *framework* voltado na medição da complexidade de modelos de simulação. Como objetivo específico foi proposto um modelo de avaliação (*framework*) para medir a complexidade em modelos de simulação, explicado em detalhes na seção 3.2 deste trabalho. Popovics e Monostori (2016) apontam para a necessidade de se ter uma ferramenta de avaliação para sistemas complexos, já que nas duas últimas décadas os sistemas de manufatura vêm se tornando cada vez mais complexos para atender à crescente demanda por rapidez e flexibilidade.

De acordo com Cauchick Miguel *et al.* (2010), a partir da pesquisa operacional, que tinha em sua origem uma característica pragmática, voltada para a resolução de problemas práticos, surgiu uma linha de pesquisa, divulgada amplamente nos Estados Unidos, focada em problemas mais idealizados e em construir conhecimento científico. Nesse contexto, a modelagem e simulação pode ser considerada uma metodologia de pesquisa quantitativa. Contudo, o trabalho em questão não tem como foco usar a simulação para explicar o funcionamento de um fenômeno real, e sim analisar o próprio fenômeno da construção de modelos de simulação.

O método de pesquisa denominado “estudo de caso” consiste em investigar um determinado fenômeno dentro de um contexto real contemporâneo através da análise de um ou mais objetos de estudo denominados “casos”. Dentre os benefícios principais da condução de um estudo de caso estão o maior entendimento de um dado fenômeno ou a possibilidade de se desenvolver novas teorias a respeito do mesmo (CAUCHICK MIGUEL *et al.*, 2010).

Segundo Voss *et al.* (2002), o Estudo de Caso em Gestão de Operações tem um papel importante para desenvolver um entendimento qualitativo sobre algum fenômeno. Este pode responder a questões como “o que”, “por que” e “como” tendo como base o conhecimento da natureza de um fenômeno, bem como testar, validar e refinar teorias sobre o mesmo. Pesquisas de caso normalmente usam um *framework* que precisa explicar graficamente os elementos a serem estudados e definir as perguntas a serem respondidas pela pesquisa. Para Yin (2001) o estudo de caso está longe de ser apenas uma estratégia exploratória, podendo haver estudos de caso de natureza exploratória, descritiva ou explanatória. Os estudos de caso também podem ser classificados segundo à quantidade de casos (único ou múltiplos) e quanto ao número de unidades de análise (holísticos - unidades únicas de análise e incorporados - unidades múltiplas).

Segundo Cauchick Miguel (2007), a condução adequada de um estudo de caso não é algo trivial, e os trabalhos desta natureza são frequentemente criticados por limitações metodológicas na escolha dos casos, análise dos dados e geração das conclusões. O autor sugere que a pesquisa seja conduzida com o rigor metodológico necessário, e para isso pode-se adotar uma sequência de etapas como ilustrado na Figura 3. O estudo de caso deste trabalho foi organizado tendo esse procedimento como guia, e o Capítulo 4 desta dissertação foi construído de acordo com as etapas seguidas na condução desta pesquisa. Este estudo em questão pode ser classificado, segundo as definições de Yin (2001) em um estudo de caso exploratório, de casos múltiplos e holísticos (4 pares de modelos submetidos ao mesmo *framework* de avaliação de complexidade).

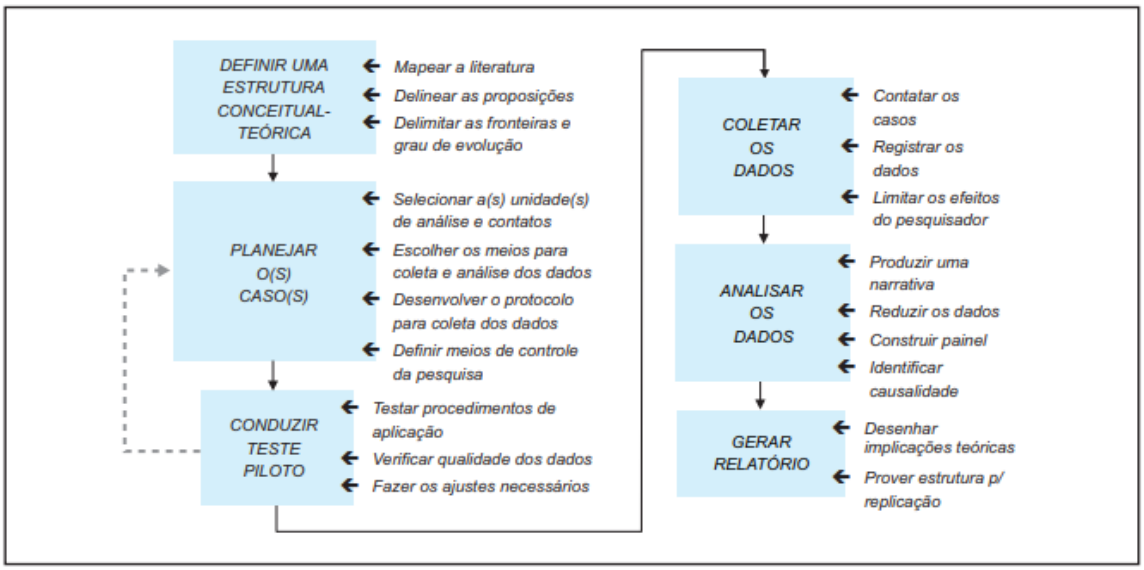


Figura 3 - Etapas de um estudo de caso.
Fonte: Cauchick Miguel (2007)

O método do estudo de caso foi adotado nesta pesquisa para buscar responder determinadas questões sobre o fenômeno em estudo. Tal fenômeno consiste na influência do tipo de abordagem de simulação utilizada na complexidade do modelo computacional, no caso a Simulação a Eventos Discretos e a Híbrida, combinando eventos discretos com agentes. O trabalho não tem por objetivo avaliar a complexidade relacionada às outras etapas da simulação, como a fase de coleta de dados, a fase da modelagem conceitual e a análise dos resultados, focando apenas na etapa da implementação do modelo computacional.

O campo da modelagem e simulação é vasto, permitindo representar sistemas de manufatura, serviços e fenômenos naturais e sociais diversos. Sendo assim, optou-se por definir condições de contorno para direcionar o estudo, limitando os tipos de modelos que pudessem ser incluídos neste trabalho. Foi definido que os modelos representariam sistemas de manufatura do tipo *flow shop*, e que cada modelo seria desenvolvido no mesmo *software* (Anylogic®). Portanto a comparação da complexidade teve por objetivo avaliar a influência da SED e da SH na complexidade de modelos computacionais com estas características.

A escolha de se usar o mesmo *software* para todos os modelos visou garantir que as diferenças intrínsecas a cada pacote de simulação não influenciassem na complexidade dos mesmos. O *software* Anylogic® é um dos pacotes comerciais desenvolvidos pela XJ Technologies da Rússia, e possui diversas funcionalidades para se desenvolver um modelo baseado em agentes. Ele funciona com base na linguagem Java e pode ser usado para criar modelos de simulação a eventos discretos, baseados em agentes e de dinâmica de sistemas. Através dele foi possível construir e comparar os modelos seguindo a metodologia definida.

Cada um dos modelos que compõe cada par foi construído pelo mesmo modelador, e dois destes pares já haviam sido publicados em trabalhos acadêmicos: Barbieri (2016) e Bernardo (2017), respectivamente. Os demais modelos foram construídos pelo autor desta dissertação, e todos os oito modelos foram validados por dois especialistas em simulação, para garantir que nenhum destes tenha sido feito de uma forma mais complexa que o necessário. A pressuposição de que essa análise de complexidade pode ser feita se baseia na revisão de literatura sobre complexidade em modelos de simulação, e indica que, apesar de não existir uma definição universal sobre o termo “complexidade”, esta pode estar associada ao número de elementos no modelo e as relações entre eles (ELMARAGHY *et al.*, 2012, POPOVICS & MONOSTORI, 2016, SARJOUGHIAN, 2017) e que é possível se comparar a complexidade de modelos alternativos modelados de formas diferentes (CHWIF *et al.*, 2000).

3.2. *Framework* para a comparação de complexidade

Um modelo de simulação é composto de vários elementos. No caso de modelos baseados na SED, entre esses elementos pode-se citar os processos, entradas e saídas de entidades, pontos de tomada de decisões, conectores entre diferentes blocos, entidades, recursos e variáveis, entre outros. Em modelos híbridos com a SBA acrescenta-se ainda os agentes e seus diagramas de estado, compostos pelos estados, transições entre eles e as ações que cada agente pode executar ao entrar/sair de um estado. Somando a isso tudo ainda há a necessidade de se digitar linhas de código em vários modelos de simulação, se a lógica necessária não estiver inclusa nas funções do *software* sendo utilizado.

Como discutido anteriormente neste trabalho, a complexidade pode ser associada ao número de elementos presente no modelo e das relações entre eles. Ao se tentar desenvolver modelos que representam o mesmo sistema real em diferentes abordagens (SED e SBA, por exemplo), é esperado que cada modelo alternativo apresente um número diferente de elementos, devido às diferenças dessas abordagens. O número de elementos de um modelo pode ser mensurado diretamente, e as relações entre estes elementos podem estar associadas a conectores, funções e variáveis por exemplo, os quais podem ser igualmente mensurados.

Um modelo construído no Anylogic® pode conter blocos que representam funções pré-definidas, representando diversas funções comumente representadas em modelos de simulação (tráfego de veículos, pedestres, sistemas de manufatura, entre outros), mas muitas funções necessitam de códigos computacionais inseridos diretamente no modelo, que precisam ser mensurados igualmente. Para isso, os códigos foram mensurados segundo o número de linhas, separadas por ponto e vírgula na programação. Segundo Borys *et al.* (2010), a métrica das linhas de código (*Lines of Code* - LOC) é a forma mais comum de se medir projetos de *softwares* orientados a objetos (como é o caso da linguagem Java, utilizada no Anylogic®).

Além da complexidade estar associada ao número de elementos presente no modelo, o termo também pode ser associado à dificuldade de se compreender ou desenvolver um determinado modelo (ELMARAGHY *et al.*, 2012; SARJOUGHIAN, 2017). Em muitos casos um modelo pode possuir um número grande de elementos repetidos, como por exemplo em um modelo que possua mais de uma linha de produção do mesmo produto. Sendo assim, um modelo com menor escopo pode exigir um maior conhecimento de simulação e do *software* da parte do modelador enquanto outro, que demanda mais tempo, utiliza poucas funções diferentes repetidas diversas vezes.

Sendo o tempo uma medida subjetiva, este não foi incluído no *framework* em questão, pois entende-se que este esteja diretamente associado à escala do modelo, e varia de acordo com a experiência e habilidade de cada modelador. Contudo, uma medida foi acrescentada para avaliar a dificuldade inerente a cada modelo (SED e SH), e esta está associada ao número de elementos **diferentes** necessários para se desenvolver cada um. Nesta medida, cada tipo diferente de blocos, funções, variáveis, comandos de programação, etc. utilizados foi contado apenas uma vez. As duas medidas (número total de elementos e número de elementos diferentes) são abordadas em separado, e utilizadas em cada par de modelos analisados.

A contagem dos elementos foi feita excluindo-se todos os recursos idênticos entre os modelos de cada par, ou seja, entram na contagem apenas os elementos **exclusivos** de cada um (são excluídos os elementos associados à animação, por exemplo, e demais blocos da SED que são aproveitados na versão SH). Para realizar a contagem de elementos encontrados em cada modelo, estes foram divididos em três tipos:

> **Recursos de software*:*** Aqui se incluem todos os blocos com funções pré-definidas disponíveis no programa, além dos conectores entre os blocos e todas as alterações feitas nas funções relacionadas a cada bloco (como a inserção de funções de probabilidade dentro dos processos, alterações nos valores, inserção de linhas de código, etc, exemplificadas pela Figura 4). Cada termo em **negrito** na Figura 4 representa que uma alteração ali foi feita, e essa marcação ajuda a medir o número de recursos utilizados. Exemplos dos principais blocos do Anylogic® utilizados na construção dos modelos são apontados no Quadro 3. Importante ressaltar que, com o objetivo de simplificar um pouco a contagem, os valores numéricos encontrados nos cronogramas ("*Schedules*") e tabelas função ("*Table Functions*") (explicados com mais detalhes no Quadro 3), não foram mensurados, tampouco alterações no nome de qualquer bloco.

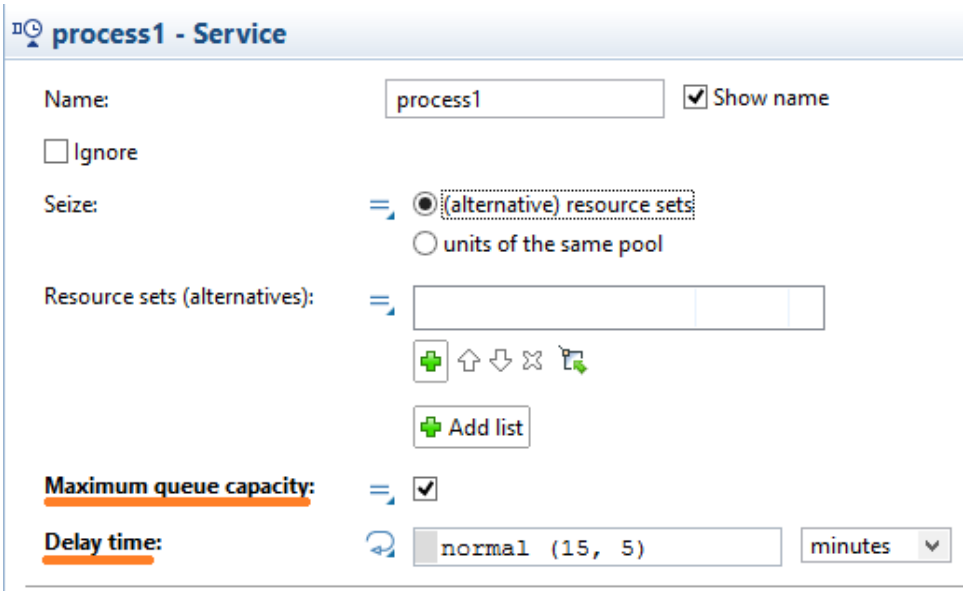


Figura 4- Exemplos de funções dos blocos do Anylogic®
Fonte: Autor

Quadro 3- Principais blocos do programa Anylogic®

Ilustração	Nome	Função
	*Source* (Entrada)	Inicia o modelo e gera entidades
	*Queue* (Fila)	Armazena entidades que não podem ser processadas no processo posterior
	*Service* (Serviço)	Captura, protela e libera entidades
	*Delay* (Protelação)	Protela entidades
	*Sink* (Saída)	Finaliza o modelo e absorve as entidades processadas
	*Resource Pool* (Conjunto de recursos)	Define as unidades de recurso que possuem a capacidade de capturar e de liberar entidades dentro dos elementos *Seize, Release, Assembler e Service*
	*Resource Task* (Tarefas dos recursos)	Permite customizar diversas tarefas a serem executadas pelos recursos (*Resource Pools*)
	*Schedule* (Cronograma)	Define um cronograma a ser seguido dentro dos elementos *Resource pool, Source e PedSource*
	*Table Function* (Tabela Função)	Permite guardar valores numéricos separados em linhas e colunas, e esses valores podem ser alterados e referenciados por outras funções.
	*Variables* (Variáveis)	Armazena resultados do modelo computacional e modela, no decorrer do tempo, unidades de dados ou características do objeto
	*Agent* (Agente)	Tem a capacidade de representar diversos componentes presentes em sistemas reais através de atributos e comportamentos
	*Parameter* (Parâmetro)	Descreve o objeto modelado representando as características do mesmo
	*Event* (Evento)	Usado para alterar algo no modelo em determinado período de tempo ou sob alguma condição

Fonte: Adaptado de Barbieri (2016).

Linhas de código: Nesse tipo incluem todas as linhas de código, mensuradas como foi explicitado na seção 3.2 deste trabalho. O Anylogic® permite a inserção de funções em diversos momentos durante a simulação, como na entrada e saída das entidades pelos processos, ou dentro do diagrama de estados dos agentes. Normalmente são digitadas dentro das funções dos blocos, dos estados e transições de estados dos agentes. A Figura 5 apresenta alguns exemplos que foram utilizados em um dos casos avaliados neste trabalho.

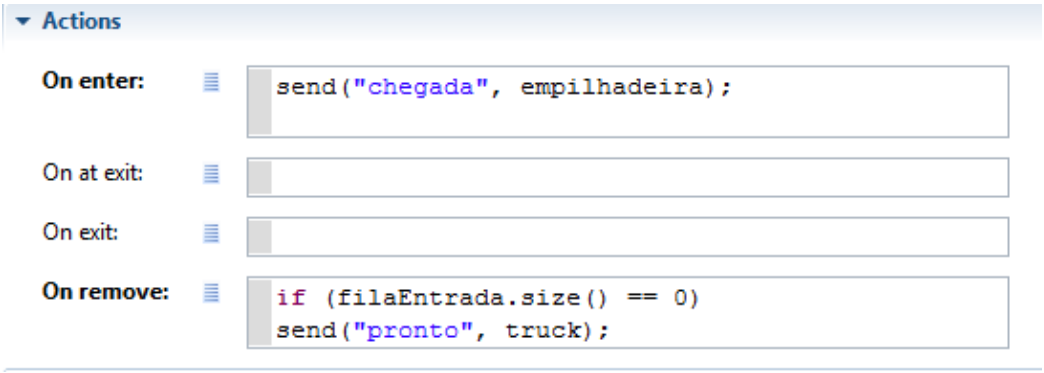
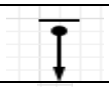
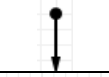
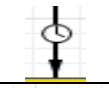
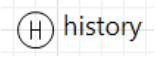
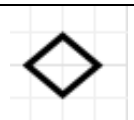


Figura 5 - Exemplos de linhas de código inseridas nas funções dos blocos do Anylogic®
Fonte: Autor

Elementos de agentes: Aqui entram apenas os elementos utilizados dentro de cada aba de agente, *sem contar as linhas de código* (que são mensuradas separadamente, seja dentro ou fora da aba dos agentes). No Anylogic® esses elementos não são exibidos na aba principal do modelo (denominada "*main*"), e sim em abas paralelas, uma para cada agente presente naquele modelo. Essas abas paralelas dos agentes também podem conter alguns elementos utilizados na aba principal, como os "*parameters*", "*events*" e "*variables*". O Quadro 4 exemplifica os principais elementos específicos para agentes.

Quadro 4- Principais elementos exclusivos de agentes

	*Entry Point* (Ponto de Entrada)	Indica o estado inicial de um diagrama de estado
	*Initial State Pointer* (Indicador do estado inicial)	Inicia o estado que se encontra dentro de outro estado
	*State* (Estado)	Representa o estado atual do agente. O elemento estado é acionado por meio de transições
	*Transition* (Transição)	Realiza a transição de um estado para outro conforme lógica estabelecida. Os tipos de evento que acionam a transição são: *Timeout Rate*, *Condition*, *Message* e *Agent Arrival*.
	*History* (Histórico)	Direciona o agente para algum estado armazenado na memória
	*Branch* (Bifurcação)	Permite ao agente escolher entre dois ou mais estados diferentes de acordo com condições pré-estabelecidas

Fonte: Autor.

Ao final da análise, cada **caso** gera quatro tabelas: duas contabilizando o uso total de recursos para ambas versões SED e SH e duas contabilizando o uso de recursos diferentes para ambas. As duas comparações são feitas em separado, sendo possível que em um caso a versão SED possua mais elementos no total mas a versão SH necessite de um maior número de recursos diferentes, por exemplo. Os recursos, dentro de cada tabela, são também divididos segundo a natureza de cada modelo. Suponha-se um modelo com dois operadores, uma empilhadeira mais

o fluxo de processos. Para este exemplo, os três tipos de recursos (recursos de agentes, recursos de software e linhas de código) seriam distribuídos segundo esses elementos (para facilitar visualmente a compreensão de cada modelo, indicando onde tais recursos foram usados) como ilustrado nas Tabelas 1 e 2 (os valores encontrados nas Tabelas 1 e 2 são meramente ilustrativos). A lógica apresentada nas tabelas abaixo serve de base para todas as tabelas geradas pelo *framework* utilizado.

Tabela 1- Exemplo de tabela.

Tabela de Exemplo – Recursos Totais SED					
	Operador01	Operador02	Empilhadeira	Processos	Total
Recursos de Agentes	0	0	0	0	**0**
Recursos de *Software*	4	3	9	1	**17**
Linhas de Código	3	3	3	0	**9**
Total	**26**				

Fonte: Autor

Tabela 2 – Segundo exemplo de tabela.

Tabela de Exemplo – Recursos Totais SH					
	Operador01	Operador02	Empilhadeira	Processos	Total
Recursos de Agentes	4	4	6	0	**14**
Recursos de *Software*	1	2	2	1	**6**
Linhas de Código	2	2	5	4	**13**
Total	**33**				

Fonte: Autor

Sendo assim, segundo a tabela 1 a implementação do elemento “operador 1” utilizou 4 recursos de software, 3 linhas de código e nenhum recurso de agentes. O valor em negrito, na linha denominada “Total” representa a soma de todos os elementos da tabela, e este valor é que determina qual dos dois modelos é o mais complexo. Neste caso hipotético, a versão mais complexa é a SH, representada na Tabela 2 (33 recursos usados contra 26 da versão SED). A mesma lógica é usada para a medida de recursos diferentes, que para cada caso é apresentada posteriormente à comparação de recursos totais.

Uma coisa importante a ser frisada nesta etapa, é que não foram usados pesos diferentes para discernir os três tipos de recursos. Cada bloco usado em um *software* de simulação (como o bloco de “processos”, por exemplo) representa um conjunto de códigos computacionais que são simplificados através da funcionalidade *drag-and-drop* para facilitar seu uso e entendimento. Portanto, em termos de recursos computacionais, um bloco normalmente vale mais que uma simples linha de código. Neste trabalho porém, o que está sendo avaliado é a complexidade do ponto de vista do modelador, segundo o número de etapas que este precisa para implementar cada lógica no seu modelo de simulação, e portanto, recursos de *software*, linhas de código e

recursos de agentes possuem o mesmo peso na contagem final da complexidade de cada modelo.

Cada par de modelos foi assim analisado e comparado, a fim de se verificar qual das versões (SED ou SH) utilizou mais recursos de forma geral e em seguida qual delas utilizou um maior número de recursos diferentes (obviamente, os recursos do tipo "agentes" só estão presentes nos modelos SH). Este *framework* pode ser replicado para qualquer modelo construído a partir do Anylogic®, mas não se pode afirmar que ele funcionaria para modelos construídos em qualquer *software* ou pacote de simulação.

Uma representação gráfica deste *framework* é apresentada na Figura 6, detalhando cada etapa do processo de medição da complexidade desenvolvido neste trabalho.

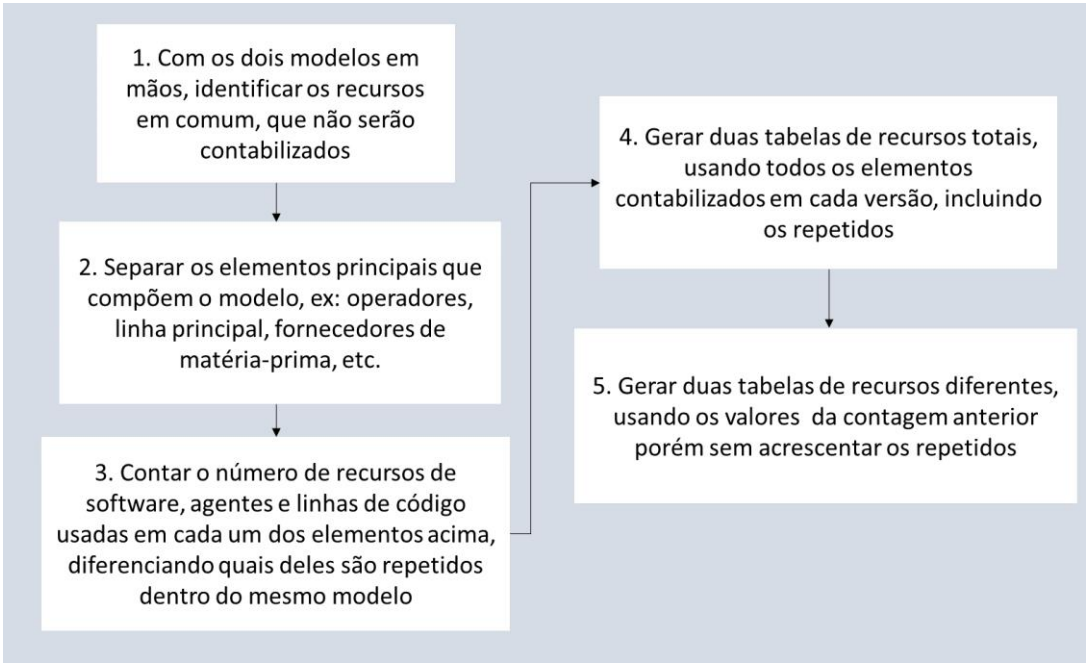


Figura 6 – Framework desenvolvido e utilizado na pesquisa.
Fonte: autor.

4. ESTUDO DE CASO

Este capítulo apresenta as etapas do estudo de caso e foi estruturado com base no método proposto por Cauchick Miguel (2007), apresentado na Figura 3. Sendo assim, cada seção do capítulo apresenta um dos passos apontados no fluxograma da metodologia usada.

4.1. Definição de uma estrutura conceitual-teórica

Para Cauchick Miguel (2007), a primeira etapa na condução de um estudo de caso é definir um referencial conceitual-teórico para o trabalho, onde é possível identificar lacunas que justifiquem a relevância da pesquisa. É nesta etapa onde são extraídos os construtos, que são elementos da literatura que representam um conceito a ser verificado, neste caso empiricamente. De acordo com a Figura 3, a etapa de definir a estrutura conceitual compreende mapear a literatura, delinear as proposições e delimitar as fronteiras e graus de evolução.

No que diz respeito ao mapeamento da literatura, este estudo de caso surgiu devido a determinadas lacunas apontadas na mesma, o que motivou a se buscar uma forma de mensurar modelos de simulação que utilizassem elementos da SBA em contrapartida a modelos construídos puramente pela SED, de forma empírica (apesar de os modelos utilizados não se basearem em problemas reais, eles representam situações tipicamente encontradas na literatura da simulação e modelagem). O estudo bibliográfico foi imprescindível para a elaboração do *framework* de medição de complexidade, já que permitiu uma melhor reflexão do significado do termo “complexidade” e o que ele representa para os modelos de simulação.

As proposições foram delineadas em seguida: como explicado anteriormente, foi necessário desenvolver um *framework* para medir a complexidade de cada par de modelos, já que não existia tal ferramenta na literatura até então. Foi então definido que a complexidade seria medida através do número de recursos exclusivos (com relação ao par de cada um) utilizados em cada modelo, poupando o trabalho de se medir elementos idênticos dos dois modelos e permitindo uma análise quantitativa de cada um. Em seguida, para direcionar a pesquisa e definir os tipos de modelos que seriam analisados, determinadas delimitações, ou condições de contorno foram definidas: ficou definido que todos os modelos deveriam representar sistemas de manufatura *flow shop*, e que todos seriam desenvolvidos utilizando o mesmo *software*, e validados por dois especialistas.

4.2. Planejamento dos casos

Anteriormente a este trabalho, dois pares de modelos já haviam sido desenvolvidos em outros trabalhos (BARBIERI, 2016; BERNARDO, 2017), contemplando, cada par, um modelo desenvolvido em SED e outro na SH. Esses quatro modelos se encaixavam nas condições de contorno estipuladas aqui, sendo que um trabalho teve por objetivo avaliar a contribuição da SH na representação do fator humano em sistemas de manufatura (BARBIERI, 2016), e o outro teve como objetivo a medição da complexidade dos dois modelos (BERNARDO, 2017). Este último serviu de inspiração para o desenvolvimento do *framework* utilizado aqui, que teve por objetivo aprofundar o estudo desenvolvido nele.

Sendo assim, estes modelos foram aproveitados para este estudo, e foram submetidos ao método de medição de complexidade desenvolvido nesta pesquisa. Os últimos dois pares de modelos foram desenvolvido exclusivamente para este estudo, onde o primeiro consiste em uma adaptação de um dos modelos presentes nos tutoriais do *software* Anylogic®, adaptado para se encaixar nas condições de contorno estabelecidas neste trabalho. O segundo foi inspirado por um problema encontrado em Hao e Shen (2008), sobre produção puxada. Estima-se que através da análise destes quatro pares de modelos, cada um representando determinadas situações encontradas na manufatura, a contribuição da SBA para este tipo de modelo possa ser melhor avaliada.

Pelo fato de todos os casos serem modelos computacionais não houve a necessidade de se estipular meios e protocolos para a coleta dos dados, que foi facilmente extraída de cada um dos modelos abrindo-os através do programa onde eles foram desenvolvidos. Segundo Eisenhardt (1989), uma quantidade de 4 a 10 casos normalmente é o suficiente para um estudo com casos múltiplos, e para este trabalho foi determinado que quatro casos seriam o suficiente, devido a restrições de tempo e a dificuldade de se encontrar pares de modelos com essas características na literatura. No entanto, o *framework* desenvolvido aqui pode ser facilmente aplicado em futuros estudos de caso para comparar outros tipos de modelos.

4.3. Condução de um teste piloto

Segundo Cauchick Miguel (2007), a condução de testes piloto em estudos de caso não é uma prática muito comum, porém é importante para auxiliar na coleta de dados. Neste estudo em particular, os dados de interesse se encontravam nos próprios modelos computacionais, que podem ser revistos a qualquer momento, e o teste piloto não foi usado nessa questão. O

framework, no entanto, evoluiu com o decorrer deste estudo, sendo que um artigo aprovado no 50º Simpósio Brasileiro de Pesquisa Operacional apresentou uma versão inicial deste (DELA SAVIA *et al.*, 2018), mais próxima do trabalho de Bernardo (2017). O artigo permitiu identificar a qualidade dos dados obtidos pelo *framework*, que foi revisto e aprimorado.

4.4. Coleta de dados

Dos quatro casos analisados, dois já estavam prontos e em posse do pesquisador. Os dois últimos casos foram desenvolvidos nesta etapa. Foram gerados assim mais dois pares de modelos, obedecendo às condições de contorno estabelecidas. Segundo Cauchick Miguel (2007), nesta etapa é importante tentar limitar os efeitos do pesquisador sobre o objeto analisado. A complexidade de um modelo pode variar segundo o modelador e o programa utilizado, e para reduzir ao máximo os ruídos, cada um dos modelos que compõem cada par foram elaborados pelo mesmo modelador, e todos eles foram validados com a ajuda de dois especialistas em Simulação Baseada em Agentes.

4.5. Análise dos dados

4.5.1. Considerações iniciais

Como explicado anteriormente a análise dos dados consistiu em analisar cada modelo e contar o número de recursos utilizados, separando-os por tipos, como detalhado na seção 3.2. Antes de se iniciar a análise, o *framework* precisou estar bem definido para que se soubesse com exatidão quais dados seriam considerados em cada modelo. Nesta sessão serão citados diversos tipos de blocos do Anylogic®, que são explicados em mais detalhes nos Quadros 3 e 4. O resultado final da análise de complexidade será apresentado na seção seguinte.

Seguindo o *framework* estipulado, a coleta de dados para cada par de modelos seguiu três etapas: primeiramente registrou-se os elementos em comum nos dois modelos, excluindo-os da análise, e a partir daí os recursos exclusivos utilizados em cada modelo foram somados e comparados (a contagem inicial foi feita manualmente). Durante essa contagem os elementos do mesmo tipo foram marcados, para possibilitar em seguida a contagem do número de recursos **diferentes** utilizados em cada modelo (lembrando que, segundo a seção 3.2, cada par de modelos foi submetido a dois tipos de análise de complexidade). Os casos estão listados segundo a ordem em que foram **avaliados** nesta pesquisa, e não a que foram desenvolvidos.

Em seguida à apresentação detalhada de cada par de modelos é apresentada a mensuração dos mesmos em quatro tabelas: duas delas medindo o número total de recursos utilizados (versão SED e SH respectivamente) e mais duas medindo o uso de recursos diferentes (versão SED e SH respectivamente). Os números em negrito no texto fazem referência aos valores apresentados na respectiva tabela.

4.5.2. Primeiro caso

O primeiro par de modelos, baseado no trabalho de Bernardo (2017) e também presente em Dela Savia *et al.* (2018), representa uma linha de produção que atua sobre uma peça através de três estações de trabalho, na qual há uma fila com capacidade 1 entre cada uma delas. A peça deve ser inspecionada no primeiro posto, embalada no segundo e expedida no terceiro. Cada posto de trabalho possui um operador associado e estes trabalham em turnos das 8 às 17h, com pausas para o almoço entre 12 e 13h. A função de cada operador é buscar a peça na fila anterior e realizar a tarefa no seu respectivo posto de trabalho. Os tempos definidos para cada etapa são distribuições normais e apresentados em segundos: o tempo de inspeção possui $\mu = 70{,}14$ e $\sigma = 11{,}18$, para a embalagem $\mu = 61{,}33$ e $\sigma = 8{,}29$, e para a expedição $\mu = 50{,}97$ e $\sigma = 5{,}18$. Novas peças entram no sistema assim que o operador 1 termina seu trabalho. A Figura 7 apresenta a estrutura conceitual do sistema representado nestes modelos, desenvolvida pela técnica IDEF-SIM. O caso 2 apresenta uma estrutura idêntica à deste caso, tendo como única diferença o comportamento dos operadores, que não é representado em um modelo IDEF-SIM por se tratar de uma lógica de simulação baseada em agentes.

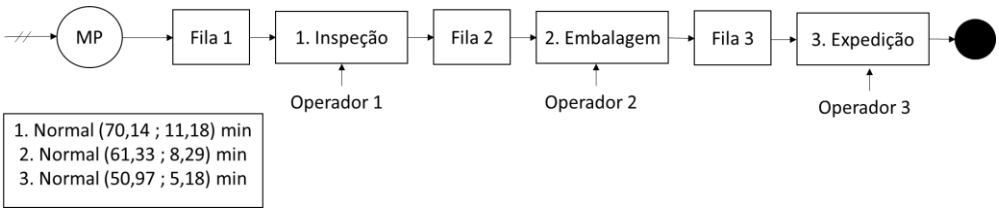


Figura 7 - Modelo conceitual em IDEF-SIM para os casos 1 e 2
Fonte: Autor

O primeiro passo da análise é identificar os elementos em comum nos dois modelos. Ambos apresentam o processo principal, composto por um bloco tipo "*source*" para gerar as peças, três operações representadas por blocos do tipo "*delay*" e um "*sink*" para remover as peças processadas do sistema, além de uma variável usada para armazenar a produção total ("TotalProduzido") e algumas linhas de código que interagem com esta variável. São usadas 3

"*resource pools*" para representar os operadores. A Figura 8 apresenta esses elementos em comum.

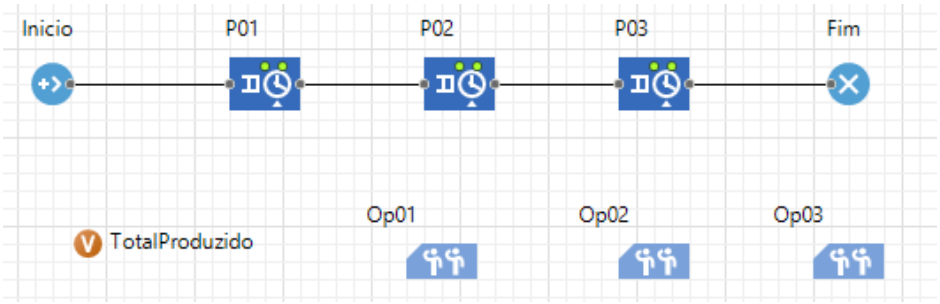


Figura 8 - Primeiro caso elementos em comum
Fonte: Autor.

A versão SED deste modelo utiliza como recursos extras uma "*schedule*" (cronograma) geral, e outros três "*schedules*", um para cada operador. Cada operador possui três variáveis, que recebem os valores da produção de cada um, seu estado atual (descansando ou trabalhando) e sua presença ou não no trabalho. Esses recursos estão demonstrados na Figura 9. As "*resource pools*" de cada um ainda recebem duas alterações para associar cada recurso aos cronogramas e uma linha de código para igualar a animação com a versão SH (indicando os estados de cada trabalhador durante a simulação).

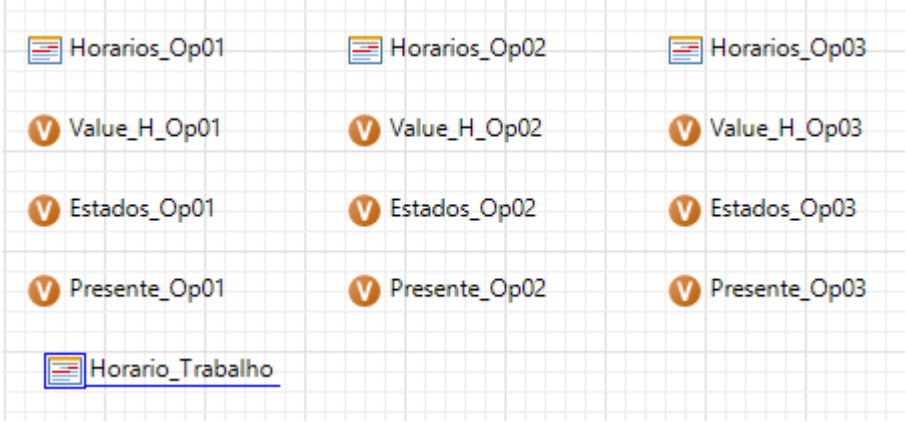


Figura 9 - Primeiro caso: elementos exclusivos da versão SED.
Fonte: Autor.

Já a versão SH utiliza outros recursos, com a adição de três agentes na aba "*main*" e uma alteração em cada "*resource pool*" dos operadores, associando a capacidade de cada um à lógica de agentes (o recurso decide se está no período de trabalho ou não pelo diagrama de estados do agente, através de mensagens). A aba de agentes deste modelo, com seu diagrama de estados se encontra na Figura 10. Essa aba possui um cronograma idêntico aos da versão SED e as três variáveis foram substituídas pelo diagrama de estados em conjunto com uma única variável.

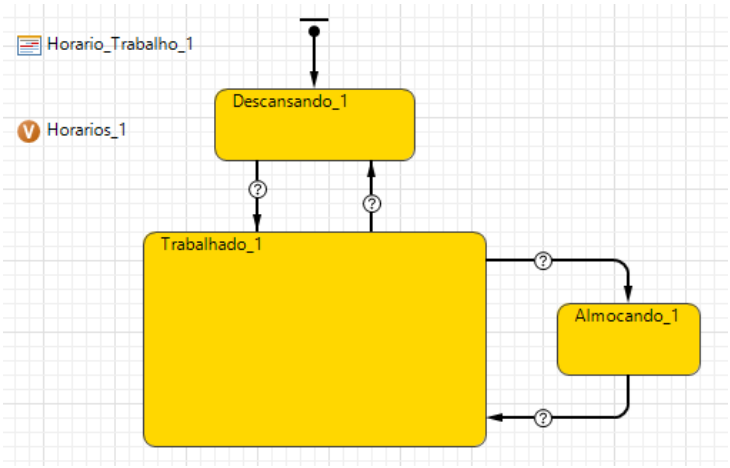


Figura 10 - Elementos de agentes do primeiro modelo
Fonte: Autor.

Com relação à medição da complexidade, as Tabelas 3 e 4 apresentam os dados relativos ao número total de recursos usados em ambos os modelos deste par. Já as Tabelas 5 e 6 apresentam o número de elementos diferentes usados em cada um.

Pela Tabela 3 percebe-se que, para a versão SED, foram necessários 18 recursos (variáveis e cronogramas e alterações nos valores) para regular o trabalho dos operadores, definindo o ponto de partida de cada um e associando-os a um recurso "*Schedule*" (1 recurso no sistema) que define a rotina dos mesmos (totalizando **19** recursos de *software* para operadores). Cada operador usou uma linha de código para se conectar ao "schedule" mencionado anteriormente, totalizando mais **3** linhas de código. Esta versão utilizou um total de **22** recursos.

Tabela 3- Recursos utilizados no primeiro caso SED

Recursos usados - SED				
	Processos	Operadores	Sistema	Total
Recursos de Agentes	0	0	0	**0**
Recursos do *Software*	0	18	1	**19**
Linhas de Código	0	3	0	**3**
Total	**22**			

Fonte: Autor

A versão SH substituiu a maioria dos recursos de *software* por agentes, modificando a forma com a qual o sistema foi representado. Neste caso, foram inseridos 3 agentes no sistema, cada um possuindo um diagrama de estados composto de 10 elementos (entre blocos de estado, transições, uma variável e um cronograma) além de 11 linhas de código regulando a animação e lógica de trabalho dos agentes (totalizando **33** linhas de código e **30** recursos de agentes). Cada operador também sofreu uma alteração no seu "*resource pool*" (totalizando 3) para

receber os valores gerados no diagrama de estados, e como o bloco agente é inserido na aba principal do modelo, os agentes somaram mais 3 recursos de *software* no sistema, totalizando **6**. Essa versão utilizou um total de **69** recursos, como indicado na Tabela 4.

Tabela 4- Recursos utilizados no primeiro caso SH

Recursos usados - SH				
	Processos	Operadores	Sistema	Total
Recursos de Agentes	0	30	0	**30**
Recursos do *Software*	0	3	3	**6**
Linhas de Código	0	33	0	**33**
Total	**69**			

Fonte: Autor

Ao se avaliar os recursos diferentes utilizados, no caso da versão SED tem-se o "*schedule*", as "*variables*", duas alterações no "*ResourcePool*" (totalizando **4** recursos de *software*), referentes ao controle da capacidade e **1** linha de código que permite mudar a visibilidade do operador durante a simulação, totalizando assim **5** recursos diferentes usados, como mostra a Tabela 5.

Tabela 5- Recursos diferentes do primeiro caso SED

Recursos diferentes - SED				
	Processos	Operadores	Sistema	Total
Recursos de Agentes	0	0	0	**0**
Recursos do *Software*	0	4	0	**4**
Linhas de Código	0	1	0	**1**
Total	**5**			

Fonte: Autor

A Tabela 6 apresenta a versão SH, que possui outro "*Schedule*" e "*Variable*" (**2** recursos diferentes de *software*), além dos blocos de estado dos agentes e conectores entre estados (**4** tipos diferentes de recursos). Além destes tem-se **3** linhas de código: para alterar a visibilidade, mudar os valores das variáveis e mudar a capacidade dos trabalhadores durante a simulação. Esta versão utiliza **9** tipos diferentes de recursos no total.

Tabela 6- Recursos diferentes do primeiro caso SH

Recursos diferentes - SH				
	Processos	Operadores	Sistema	Total
Recursos de Agentes	0	4	0	**4**
Recursos do *Software*	0	1	1	**2**
Linhas de Código	0	3	0	**3**
Total	**9**			

Fonte: Autor

Pode-se perceber que o modelo SH foi considerado mais complexo em ambos os casos, sendo aproximadamente 3,14 vezes mais complexo com relação a recursos totais e 1,8 vezes mais complexo com relação ao uso de recursos diferentes. Uma análise comparativa entre os quatro casos será apresentada no item 4.6 deste capítulo.

4.5.3. Segundo caso

O segundo caso, baseado em Barbieri (2016), apresenta a mesma linha de produção do caso anterior, com os mesmos tempos e processos. Aqui, a diferença é que foi utilizado um sistema que permitiu que o rendimento do operador pudesse variar durante a simulação. Esse sistema consiste na variação do ritmo de trabalho de cada operador, e foi utilizada a tabela do método Westinghouse, proposto por Barnes (1977). Esse método avalia o rendimento dos trabalhadores a partir de quatro fatores: Habilidade, Esforço, Condições e Consistência. Seus valores influenciam positiva e negativamente o ritmo de trabalho dos operadores, sendo determinados de maneira estocástica (BARBIERI, 2016).

A variação do ritmo de cada operador é ditada pelo objetivo diário de produção, definido inicialmente como 322 peças. A produção de cada operador é verificada várias vezes e se os valores estiverem mais baixos ou mais altos que o esperado, o ritmo deste operador se modifica, segundo a soma dos números da tabela Westinghouse que correspondem aos seus valores atuais de Habilidade, Condições, Esforço e Consistência. O Quadro 5 apresenta os valores matemáticos do sistema Westinghouse.

Quadro 5- Fatores da Tabela Westinghouse

Habilidade			Esforço		
+0,015	A1	Muito Habilidoso	+0,013	A1	Excessivo
+0,13	A2		+0,12	A2	
+0,11	B1	Excelente	+0,10	B1	Excelente
+0,08	B2		+0,08	B2	
+0,06	C1	Bom	+0,05	C1	Bom
+0,03	C2		+0,02	C2	
0,00	D	Médio	0,00	D	Médio
-0,05	E1	Regular	-0,04	E1	Regular
-0,10	E2		-0,08	E2	
-0,16	F1	Fraco	-0,12	F1	Fraco
-0,22	F2		-0,17	F2	
Condições			Consistência		
+0,06	A	Perfeita	+0,04	A	Perfeita
+0,04	B	Excelente	+0,03	B	Excelente
+0,02	C	Boa	+0,01	C	Boa
0,00	D	Média	0,00	D	Média
-0,03	E	Regular	-0,02	E	Regular
-0,07	F	Fraca	-0,04	F	Fraca

Fonte: Adaptado de Barnes (1977).

Para este caso, o modelo conceitual é idêntico ao apresentado na Figura 7. Ambos modelos deste caso apresentam, em comum, os três processos e a variável “total produzido”, como no caso anterior (idêntico à Figura 9). O que se acrescenta neste caso em ambos os modelos são códigos que regulam o quanto foi produzido em cada processo, permitindo criar a lógica para a variação de rendimento de cada operador durante a simulação, que são diferentes em cada versão. Analisando-se o modelo SED, ele possui 2 alterações nas funções de cada “resource pool” (para alterar sua capacidade durante a simulação), um cronograma “schedule” (do sistema, afeta todos os operadores, apresentado na Figura 11) e os diversos atributos de cada operador, também demonstrados na Figura 11. Com exceção do cronograma geral, os atributos dentro desta *box* da Figura 11 são replicados 3 vezes, uma para cada operador.

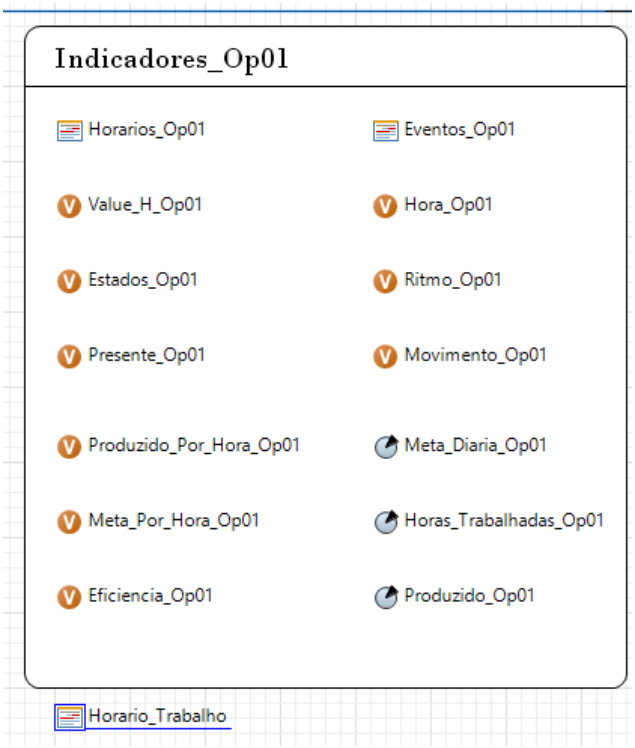


Figura 11 - Segundo caso SED
Fonte: Autor.

Esses elementos dentro da *box* da Figura 11 são usados para representar a variação da produção de cada operador segundo as variáveis da Tabela Westinghouse. Consistem em 9 variáveis, 3 parâmetros e dois cronogramas. Esses elementos ainda recebem 6 alterações em seus valores, para funcionarem devidamente (alguns parâmetros e variáveis iniciam com um valor inicial fixo, ou proporcional a alguma outra variável ou parâmetro). As variáveis recebem os valores associados à produção, ritmo, estado, etc. de cada operador durante a simulação, já os parâmetros são sempre comparados com valores padrão para regular a alteração do ritmo do

operador. Os dois cronogramas indicam os momentos onde ocorrem mudanças no estado do operador.

Baseado na metodologia Westinghouse, cada operador ainda possui mais 12 parâmetros associados aos seus valores de Esforço, Habilidade, Condições e Consistência, sendo três parâmetros para cada valor (negativo, nulo e zero). Cada valor está associado a uma função tabela com os dados da tabela Westinghouse, como na Figura 12.

Figura 12- Parâmetros Westinghouse no modelo SED
Fonte: Autor.

Já para o modelo SH desse caso, foram inseridos três agentes, e cada “resource pool” recebeu uma alteração na sua capacidade (associada ao agente, e não mais ao cronograma geral, como no modelo anterior). Aqui, toda a complexidade associada às variações de ritmo dos trabalhadores se encontra dentro de cada agente. O diagrama de estados para os agentes possui 7 estados e 12 transições, além de uma “schedule”, cinco variáveis e três parâmetros, que exercem função semelhante ao caso anterior (mas neste caso são necessários um menor número de elementos devido ao diagrama de estados do agente já controlar a mudança de estado destes). Já a lógica de variação de ritmo foi construída de forma similar ao modelo SED, possuindo os parâmetros e funções apresentados na Figura 12. A diferença aqui é que a lógica que conecta

cada operador aos valores destas tabelas e variáveis se encontra dentro do diagrama de estados, e não nos processos da aba principal, que recebe os valores gerados pelo agente por mensagens. A Figura 14 apresenta os recursos dentro de cada agente (variáveis, parâmetros, etc.). O diagrama de estados dos agentes necessitou de um grande número de linhas de código para representar a lógica necessária (detalhados em seguida).

A aba principal do modelo SH e o diagrama de estados com os recursos inseridos nos agentes se encontram, respectivamente, nas Figuras 13 e 14.

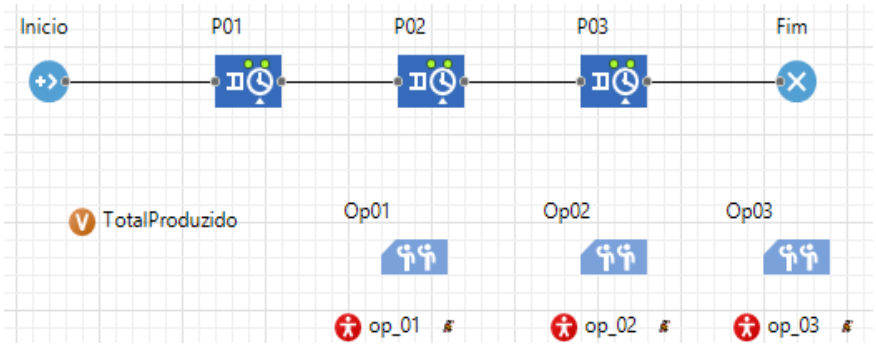


Figura 13- Segundo modelo SH
Fonte: Autor.

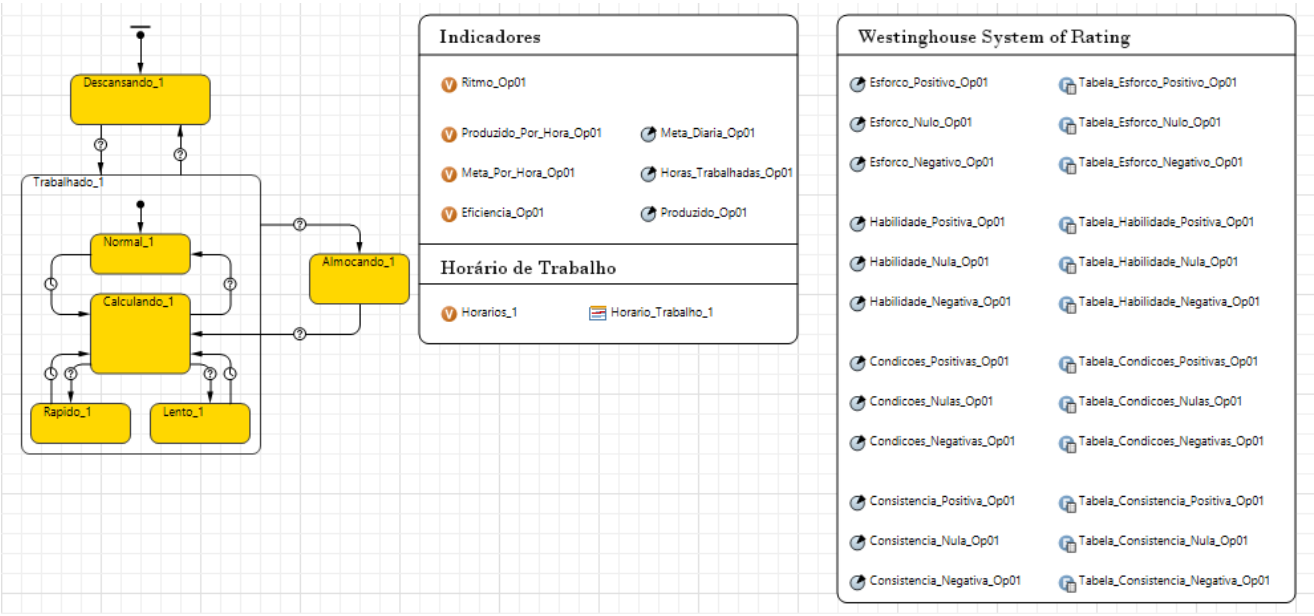


Figura 14- Elementos de agentes do segundo modelo SH
Fonte: Autor.

As Tabelas 7 e 8 apresentam a mensuração dos dados relativos ao número total de recursos usados, e as Tabelas 9 e 10 apresentam a mensuração do número de recursos diferentes utilizados.

Segundo a Tabela 7, o modelo SED possui **1** cronograma geral (contabilizado no sistema) e uma linha de código em cada operador, que afeta a animação destes durante a simulação (**3** linhas de código no total). Cada operador possui 2 alterações em seus "*resourcePools*" como mencionado anteriormente (totalizando 6), e a soma de todos os elementos da Figura 11 totaliza mais 57 recursos de *software*. Finalizando, tem-se os 12 parâmetros e 12 funções apontados na Figura 12, que multiplicados por 3 operadores somam mais 72 recursos (**135** recursos de *software*, sendo 6 + 57 + 72). O total de recursos usados no modelo é **139.**

Tabela 7- Recursos utilizados no segundo caso SED

Recursos usados - SED				
	Processos	Operadores	Sistema	Total
Recursos de Agentes	0	0	0	**0**
Recursos do *Software*	0	135	1	**136**
Linhas de Código	0	3	0	**3**
Total	**139**			

Fonte: Autor

O modelo SH, como mostra a Tabela 8, insere **3** blocos de agente no sistema e uma alteração na capacidade de cada "*resourcePool*" (diferente das duas alterações do modelo anterior, resultando em **3** recursos de *software*). O restante dos recursos se encontra no diagrama de estados dos agentes, apresentado na Figura 14, com 11 elementos na *box* à esquerda (5 variáveis, 3 parâmetros e um cronograma, além de 2 alterações nos valores iniciais das variáveis), 24 elementos na *box* à direita e mais 19 elementos no diagrama de estados propriamente dito (7 blocos e 12 conectores). Isso totaliza 54 recursos, que multiplicados pelo número de agentes (3) resulta em **162** recursos na aba de agentes para os operadores no total.

Como mencionado anteriormente, muitas linhas de código foram usadas para criar a lógica da variação de ritmo dentro do diagrama de estados segundo a metodologia Westinghouse. Um total de 69 linhas de código foram usadas por agente, totalizando **207** linhas para todos os três. Um total de **375** recursos foi usado nessa versão.

Tabela 8- Recursos utilizados no segundo caso SH

Recursos usados - SH				
	Processos	Operadores	Sistema	Total
Recursos de Agentes	0	162	3	**165**
Recursos do *Software*	0	3	0	**3**
Linhas de Código	0	207	0	**207**
Total	**375**			

Fonte: Autor

Com relação aos recursos diferentes, o modelo SED, representado pela Tabela 9, apresenta **1** recurso do sistema (cronograma), **1** linha de código (visibilidade dos operadores durante a simulação), e **7** tipos de elementos associados aos operadores (variáveis, funções, parâmetros, 2 alterações na capacidade, e 2 tipos de alterações nas variáveis e parâmetros). Totaliza-se **9** recursos diferentes utilizados.

Tabela 9- Recursos diferentes do segundo caso SED

Recursos diferentes - SED				
	Processos	Operadores	Sistema	Total
Recursos de Agentes	0	0	0	**0**
Recursos do *Software*	0	7	1	**8**
Linhas de Código	0	1	0	**1**
Total	**9**			

Fonte: Autor

A versão SH, vide Tabela 10, utilizou **9** tipos diferentes de recursos (variáveis, parâmetros, cronogramas, etc., além dos blocos e conectores específicos de agentes) no diagrama de estados dos agentes. **1** tipo de bloco "agente" é introduzido no sistema, além da alteração na capacidade das "*resourcePools*" para os operadores (**1** recurso de *software*). Entre as 69 linhas de código usadas em cada agente, foram usadas **6** funções diferentes, que afetam a animação do modelo, alteram a capacidade e os valores dos parâmetros de cada agente durante a simulação e reiniciam os valores em cada ciclo de trabalho. A versão SH utilizou-se de **17** tipos diferentes de recursos.

Tabela 10- Recursos diferentes do segundo caso SH

Recursos diferentes - SH				
	Processos	Operadores	Sistema	Total
Recursos de Agentes	0	9	1	**10**
Recursos do *Software*	0	1	0	**1**
Linhas de Código	0	6	0	**6**
Total	**17**			

Fonte: Autor

Pode-se perceber que o modelo SH foi considerado mais complexo em ambos os casos, sendo aproximadamente 1,49 vezes mais complexo com relação a recursos totais e aproximadamente 1,89 vezes mais complexo com relação ao uso de recursos diferentes. Uma análise comparativa entre os quatro casos será apresentada no item 4.6 deste capítulo.

4.5.4. Terceiro caso

O terceiro caso foi desenvolvido especialmente para este trabalho, e é baseado em um exemplo disponível no Anylogic®. Mudanças foram feitas no intuito de adequá-lo às condições de contorno definidas nesta pesquisa. O modelo consiste em uma fábrica com um torno CNC, um depósito e uma empilhadeira. De hora em hora um caminhão chega, depositando em média 4 peças (1 peça mais o resultado de uma distribuição binomial com p = 0,5 e n = 6), que devem ser levadas pela empilhadeira até o depósito, liberando o caminhão para retornar. Em seguida a empilhadeira leva as peças para serem processadas pelo CNC, que leva um tempo para processá-las determinado por uma distribuição triangular com mínimo 2, média 3 e máximo de 5 minutos. Foi definido que a empilhadeira só carrega uma peça por vez e o torno tem a capacidade de processar também uma peça por vez. A Figura 15 apresenta a estrutura conceitual do modelo em IDEF-SIM.

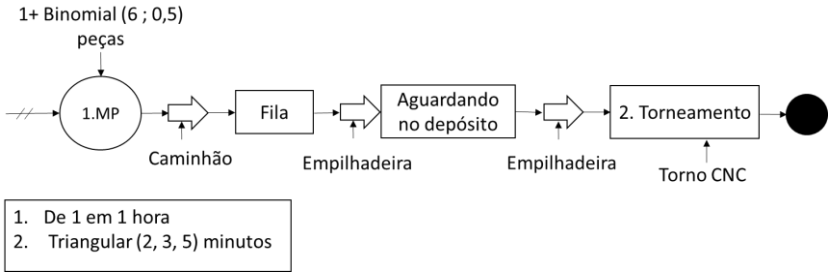


Figura 15 - Modelo conceitual em IDEF-SIM para o caso 3.
Fonte: Autor.

Diferentemente dos outros casos, onde os modelos já haviam sido validados, este par de modelos ainda precisou ser validado para poder ser usado como um dos casos. Por se tratar de um modelo hipotético, este não foi validado com dados reais, mas sim com relação às duas versões do modelo, que precisaram provar estatisticamente representar o mesmo sistema. A validação se deu através de um teste do tipo *two-sample t*, de média e variância populacional desconhecidas, segundo Montgomery e Runger (2003).

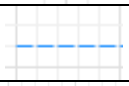
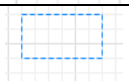
O teste serviu para verificar se seria possível aceitar a hipótese que as médias populacionais relativas ao número total de peças processadas pelo CNC após 2 dias seriam iguais para os dois modelos, com um grau de significância de 5%. Para se calcular o número de amostras necessário foi utilizada a Equação 1, extraída de Montgomery e Runger (2003), onde $z_{\alpha/2}$ equivale a 1,675 (pela tabela da distribuição normal), σ equivale ao desvio padrão da amostra piloto (neste caso, 30 valores de cada modelo foram coletados) e E equivale ao erro máximo tolerado, aqui estipulado como 5 unidades.

$$n = \left(\frac{z_{\alpha/2}\,\sigma}{E}\right)^2 \quad (1)$$

A Equação 1 retornou um valor mínimo de amostras igual a 7 para a versão SED e 14 para a versão SH, o que está bem abaixo da amostra piloto, significando que um número maior de dados não precisou ser coletado. Os 30 valores gerados pelo modelo SED e pelo modelo SBA se encontram nos Apêndices A e B, respectivamente. Assim, o teste *two sample t* foi realizado no programa Minitab® versão 17, gerando um intervalo de confiança para a diferença das médias equivalente a (-3,91 ; 6,18), e como este intervalo contém o valor zero não se pode rejeitar a hipótese de que as médias dos dois modelos são iguais.

Este modelo utiliza outros elementos do Anylogic®, que serão explicados em seguida, no Quadro 6, e não foram utilizados nos dois casos anteriores.

Quadro 6- Recursos diferentes utilizados no terceiro modelo

	*Point Node* (Nó Pontual)	Define um ponto no espaço como referência
	*Path* (Trajetória)	Define uma trajetória a ser percorrida pelo espaço
	*Rectangular Node* (Nó Retangular)	Semelhante ao *Point Node* porém com uma área retangular
	*Seize* (Capturar)	"Captura" um recurso, para que ele não possa ser usado por outras entidades ou agentes
	*Release* (Liberar)	Libera um recurso capturado pelo "*seize*"
	*Enter* (Entrada)	Funciona em conjunto com o comando (em código) "*remove*", e permite que uma entidade ou agente seja transportado de um elemento a outro
	*MoveTo* (Mover até)	Move uma entidade que percorre o fluxo de um ponto a outro
	*Hold* (Impedir)	Impede o fluxo de entidades até que alguma condição seja satisfeita

Fonte: Autor.

Os dois modelos apresentam, como elementos em comum, uma imagem de fundo que serve de mapa para definir as distâncias percorridas pela empilhadeira e pelo caminhão durante a simulação, com alguns pontos ("*nodes*") e caminhos ("*paths*") previamente definidos. Um "*source*" que serve de entrada para as peças (trazidas pelo caminhão) e algumas filas de espera tipo "*queues*" (aguardando empilhadeira para ir ao depósito e para ir ao CNC), além dos blocos que utilizam o recursos CNC ("*seize*" e "*release*").

O modelo SED é apresentado na Figura 16 e cria toda a lógica de transporte através de processos. O caminhão é considerado um recurso, criado por um "*source*" e faz sua trajetória com blocos do tipo "*moveTo*", descarrega através de um código para criar as entidades e é retirado da simulação ao sair, aparecendo outro em seu lugar uma hora depois. A empilhadeira é tratada como um recurso e seus movimentos são controlados pelos blocos "*moveTo*", e as peças usam "*seize*" e "*release*" para utilizá-la para serem levadas até o depósito e posteriormente até o torno. Um bloco "*hold*" controla o fluxo de entidades, impedindo que elas usem a empilhadeira até que o torno se desocupe.

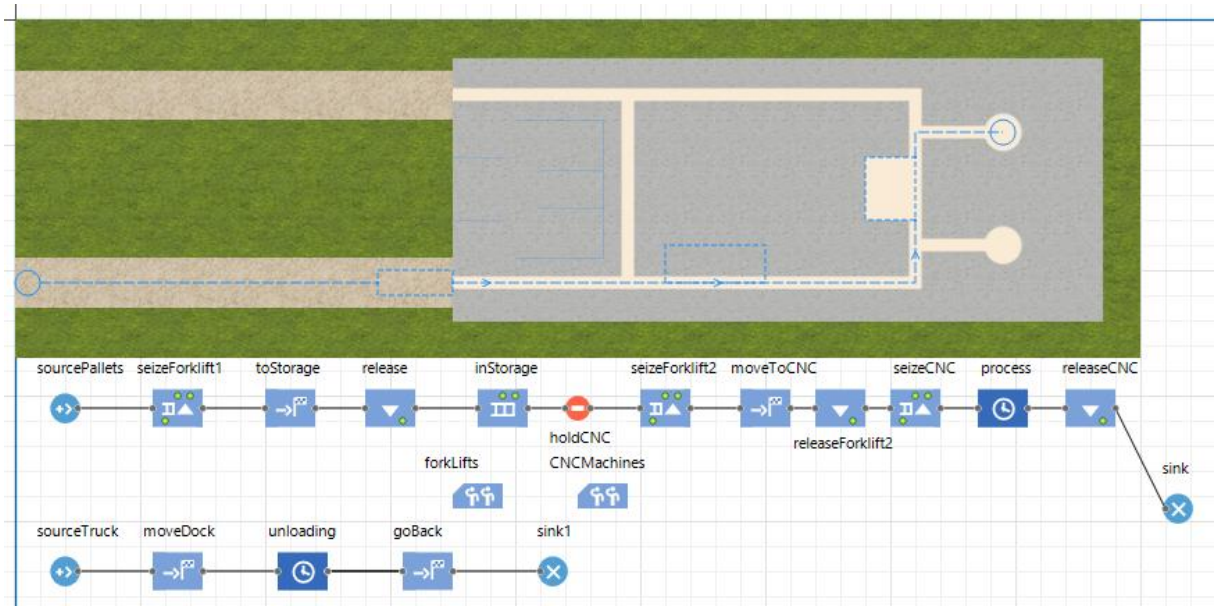


Figura 16- Terceiro modelo SED
Fonte: Autor.

A versão SED utiliza alguns códigos para desbloquear e bloquear o fluxo e para liberar o caminhão para retorno. Já a versão SH substitui muitos dos blocos encontrados na Figura 16 por ações dos agentes, considerando a empilhadeira e o caminhão como tais. Os processos neste modelo se encontram na Figura 17, onde observa-se algumas "brechas" na sequência de processos. Isso ocorre quando a peça é pega pela empilhadeira, e a partir daí seu movimento é controlado através do diagrama de estados do agente "*forklift*". Para isso, diversos comandos foram utilizados no diagrama de estados de agentes, e as peças retornam ao fluxo através dos blocos "*enter*".

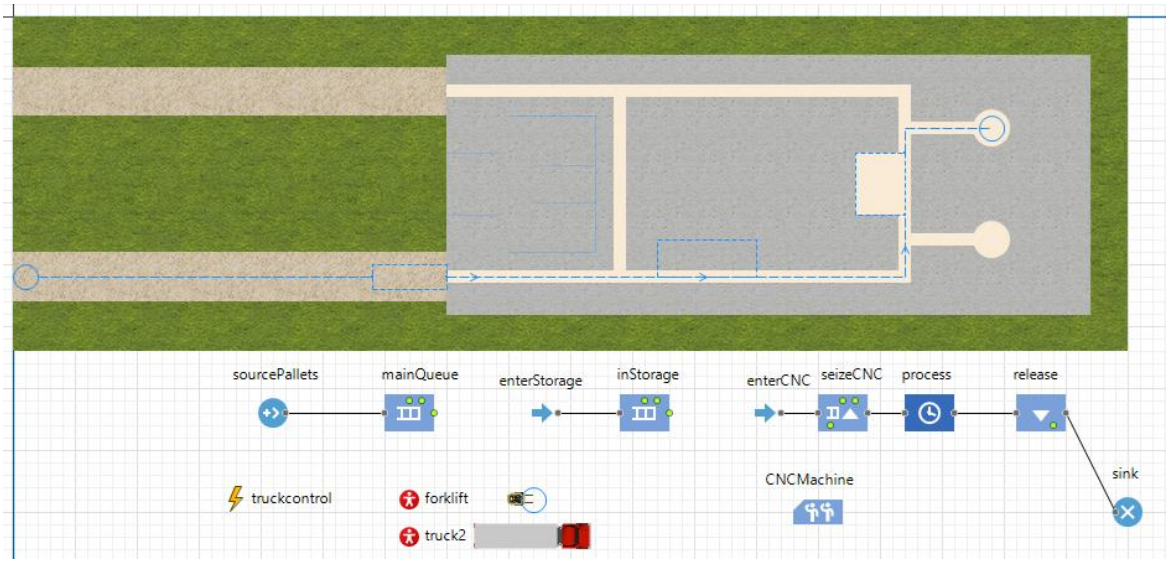


Figura 17- Terceiro modelo SH
Fonte: Autor.

Neste caso, tratando-se de uma simulação híbrida, apenas o torno CNC foi tratado como um recurso, estando presente uma "*resource pool*" associada ao mesmo. Um evento regula as chegadas do agente caminhão (chamado "*truck2*"). A maioria de blocos e linhas de código se encontram nos diagramas de estado dos dois agentes, apresentados nas Figuras 18 e 19, respectivamente. O diagrama do caminhão é bem simples e depende de duas mensagens para agir: uma para entrar no sistema (regulada pelo evento na aba principal) e uma gerada quando a empilhadeira coleta a última peça de dentro dele, para fazê-lo sair do sistema. Já a empilhadeira apresenta um diagrama bem mais complexo, com diversas decisões a tomar, e usa duas variáveis, uma para interagir com as peças e outra para se dirigir ao torno CNC no momento exato.

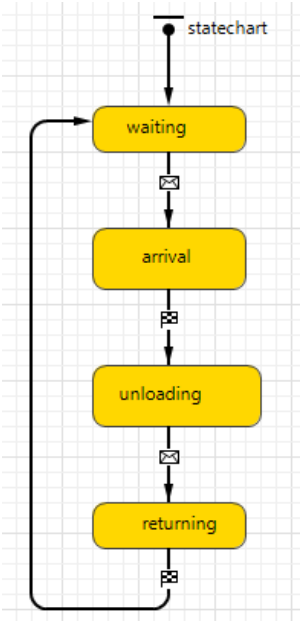


Figura 18- Diagrama de estados do caminhão
Fonte: Autor

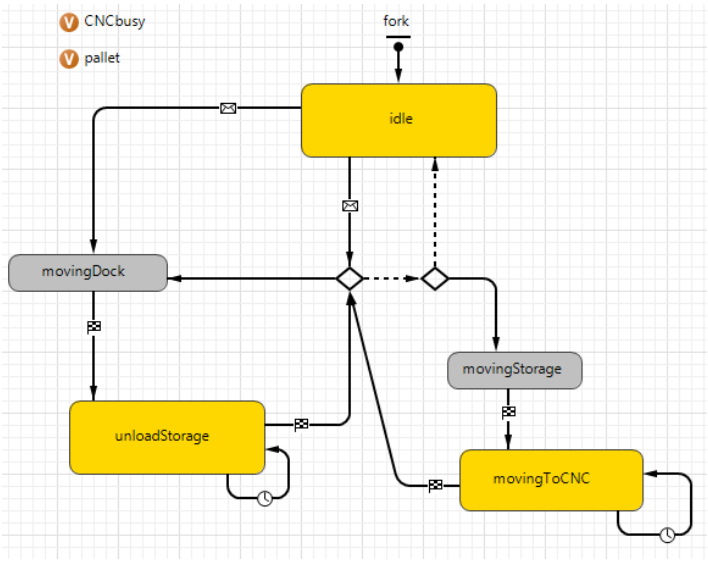


Figura 19- Diagrama de estados da empilhadeira
Fonte: Autor

A prioridade da empilhadeira é sempre pegar as peças no caminhão para liberá-lo, antes de leva-las para o torno. Ela toma decisões de como agir nos blocos "*branch*". Ao todo, esse diagrama utiliza 16 linhas de código para interagir com as peças, o torno e tomar decisões. A maior diferença entre as duas versões (SED e SH) é onde se encontra a complexidade: na SED utiliza-se diversos blocos em sequência enquanto na SH a maior parte dos comandos e decisões são realizados pelos dois agentes.

As Tabelas 11 e 12 apresentam o número total de recursos utilizados em cada modelo. Já as Tabelas 13 e 14 apresentam o número de elementos diferentes usados em cada um. Para o modelo SED, foi inserido um "*resource pool*" para a empilhadeira, com 3 alterações para definir posição inicial, velocidade e animação (total de **4** elementos). O fluxo do caminhão foi todo construído na aba principal (vide Figura 16) através de 5 blocos (com 7 alterações no "*source*" para criar o comportamento do caminhão, uma no bloco "*Move to*" e outra no "*delay*") e 4 conectores, totalizando assim **19** recursos de *software* para ele. **3** linhas de código foram também usadas (uma no próprio fluxo do caminhão e outra no fluxo principal) para criar a interação deste com os processos principais.

O fluxo principal de processos possui **25** recursos a mais que a versão SED (7 blocos e 6 conectores). Aqui o os blocos "*seize*" e "*release*" são usados para chamar a empilhadeira pra buscar as peças, inclusive um "*block*" foi usado para impedir que a empilhadeira carregue as peças antes do torno CNC se desocupar. Além desses 13 recursos, os blocos "*seize*" possuem algumas alterações para especificar o recurso utilizado e sua prioridade, os blocos "*move to*" possuem especificações de destino e o "*block*" possui uma alteração na lógica, totalizando mais 12 recursos, somando 25 recursos totais de *software* para o fluxo principal. Por último, o fluxo principal ainda possui **1** linha de código para desbloquear o caminho para o CNC, que na versão SH é substituída pelo agente. Como pode ser visto na Tabela 11, este modelo utilizou **52** recursos.

Tabela 11- Recursos utilizados no terceiro caso SED

Recursos usados - SED				
	Caminhão	Empilhadeira	Processos	Total
Recursos de Agentes	0	0	0	**0**
Recursos do *Software*	19	4	25	**48**
Linhas de Código	3	0	1	**4**
Total	**52**			

Fonte: Autor

Já na versão SH foram inseridos 2 blocos diferentes (do tipo "*enter*") nos processos principais, e um evento ("*truckcontrol*") responsável por iniciar as atividades do caminhão, que passa a ser um agente. Esse evento possui 2 alterações para definir os horários das atividades do caminhão. Ao "*source*" do fluxo principal foi adicionada uma alteração para definir um tipo de agente e possibilitar sua interação com a empilhadeira, totalizando **7** recursos de *software* no fluxo principal.

Como o caminhão e a empilhadeira se tornaram agentes, um bloco "*agent*" foi criado para cada um com duas alterações para definir a posição inicial e a velocidade (assim, **3** recursos de *software* para cada). Foi necessário programar para criar a interação dos processos com os 2 agentes, sendo que a primeira "*queue*" usa 3 linhas de código (uma para interagir com a empilhadeira e duas para interagir com o caminhão), o "*releaseCNC*" interage com a empilhadeira (liberando o acesso ao torno CNC) através de mais 2 linhas e o evento ("*truckcontrol*") interage com o caminhão através de uma linha, totalizando **6**.

Como pode ser visto na Tabela 12, o diagrama de estados do caminhão possui 4 blocos e 6 transições entre eles (totalizando **10** recursos de agentes) e foram necessárias **3** linhas de código para seu movimento e ações (segundo Figura 18). A empilhadeira foi o agente que mais consumiu recursos, possuindo um complexo esquema de transição entre estados e interações com o fluxo principal. Possui 2 variáveis (uma interage com os *pallets*, que são o produto transportado até o CNC, e outra com o CNC em si). Ao todo, são 5 estados diferentes e 13 transições entre estados, além de duas bifurcações, totalizando **22** recursos. A lógica usada para ligar o agente ao fluxo e interagir com os *pallets* requeriu **16** linhas de código, como condições para o agente tomar certos caminhos, remoção dos *pallets* de um ponto a outro, movimentação, interação com o torno CNC, etc. Este modelo usou um total de **71** recursos.

Tabela 12- Recursos utilizados no terceiro caso SH

Recursos usados - SH				
	Caminhão	Empilhadeira	Processos	Total
Recursos de Agentes	10	22	0	**33**
Recursos do *Software*	3	3	7	**13**
Linhas de Código	3	16	6	**25**
Total	**71**			

Fonte: Autor

Na Tabela 13, em termos de recursos diferentes, a versão SED usou os blocos que interagem com recurso ("*seize*" e "*release*"), "*move to*", "*block*" e algumas alterações nos blocos dos processos principais que não foram necessárias na outra versão, totalizando **8**, mais **1** código para interagir com o caminhão. O fluxo do caminhão usa 4 alterações no "*sourceTruck*" e 2 no processo "*delay*" (totalizando **6** recursos), além de **3** linhas de código. A empilhadeira usa, de diferente, apenas duas alterações no seu "*resource pool*" (**2** recursos de *software*). No total, este modelo utiliza **20** tipos diferentes de recursos.

Tabela 13- Recursos diferentes do terceiro caso SED

Recursos diferentes - SED				
	Caminhão	Empilhadeira	Processos	Total
Recursos de Agentes	0	0	0	**0**
Recursos do *Software*	6	2	8	**16**
Linhas de Código	3	0	1	**4**
Total	**20**			

Fonte: Autor

Conforme mostra a Tabela 14, pode-se observar que a empilhadeira necessitou de **8** tipos de códigos diferentes para suas funções mencionadas anteriormente, em contraste com os **2** códigos diferentes do caminhão e **5** do sistema (dentro do fluxo principal, interagindo com os agentes). O fluxo principal usou **10** recursos diferentes, incluindo os blocos do tipo "*enter*" e a inserção de agentes. Os elementos típicos de agentes foram contabilizados primeiro no caminhão (**4** deles na figura 18, além do evento) e aqueles que se repetiam não foram somados na coluna da empilhadeira, restando **2** elementos que são as variáveis e as bifurcações. Totaliza-se **32** recursos diferentes usados nesse modelo.

Tabela 14- Recursos diferentes do terceiro caso SH

Recursos diferentes - SH				
	Caminhão	Empilhadeira	Processos	Total
Recursos de Agentes	4	2	0	**6**
Recursos do *Software*	1	0	10	**11**
Linhas de Código	2	8	5	**15**
Total	**32**			

Fonte: Autor

Pode-se perceber que o modelo SH foi considerado mais complexo em ambos os casos, sendo aproximadamente 1,4 vezes mais complexo com relação a recursos totais e aproximadamente 1,6 vezes mais complexo com relação ao uso de recursos diferentes. Uma análise comparativa entre os quatro casos será apresentada no item 4.6 deste capítulo.

4.5.5. Quarto caso

O quarto caso também foi desenvolvido para esta pesquisa, e consiste em uma linha de produção com três processos em sequência (similar ao primeiro caso), sendo que neste exemplo a produção é puxada. Os pedidos chegam (a uma taxa de 0,7 pedido/ hora) e aguardam até que os três operadores estejam disponíveis para processar o produto, e isso foi implementado no modelo através do recurso "*block*", que bloqueia a linha de produção até que determinada condição seja cumprida. Os três processos têm sua duração segundo a distribuição normal,

sendo (em minutos) μ = 15 e σ = 5 para os processos 1 e 3 e μ = 20 e σ = 5 para o processo 2. A maior diferença, com relação aos casos anteriores, reside no fato de que cada operador se dispõe de no máximo 4 peças, que são consumidas uma a uma cada vez que este opera o produto. Quando o estoque de peças acaba para algum dos três operadores este deve aguardar até que uma empilhadeira se desloque até seu posto de trabalho e reponha este estoque. A reposição acontece segundo uma ordem de prioridade (processo 1, 2 e 3 nessa ordem) e o operador da empilhadeira checa o processo de 30 em 30 minutos para conferir se alguma reposição é necessária. A Figura 20 apresenta o modelo conceitual em IDEF-SIM para o este caso.

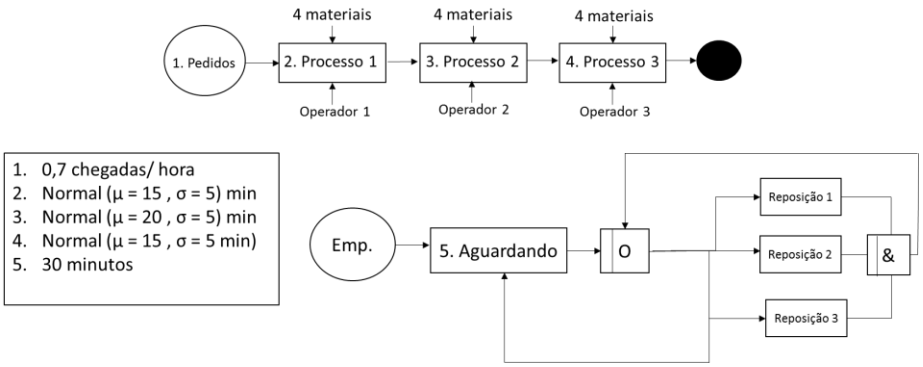


Figura 20 - Modelo conceitual em IDEF-SIM para o caso 4
Fonte: Autor

Ambas as versões SED e SBA apresentam o mesmo fluxo principal (segundo Figura 21) com a entrada de pedidos ("*source*"), fila ("*mainqueue*", onde os pedidos aguardam até que o produto anterior tenha sido finalizado), três processos, saída e bloqueadores de fluxo, além das variáveis associadas ao número de peças em estoque, que são consumidas e repostas (denominadas "*materials"* 1, 2 e 3, recebem valores de 0 a 4). O primeiro bloqueador de fluxo se encontra antes do primeiro posto (denominado "*push*") e é regulado pela lógica da produção puxada, se abrindo quando o terceiro operador termina o seu trabalho, e se fechando novamente assim que um pedido passa por ele. Os outros três bloqueadores de fluxo ("*limit1*", "*limit2*" e "*limit3*") se fecham quando o número de peças no estoque do operador seguinte se iguala a zero, abrindo-se novamente à medida em que essas peças vão sendo repostas pela empilhadeira.

A empilhadeira se situa a uma distância aproximada de 300 metros do processo 2, como indicado na Figura 21 e se locomove a uma velocidade de 2 metros por segundo. Quando solicitada, viaja ao posto de trabalho que requer a reposição (segundo a prioridade especificada anteriormente) repondo os materiais, então confere novamente se mais algum posto precisa de

peças. Quando mais nenhum deles possuir um valor “0” associado a seu estoque, a empilhadeira aguarda 30 minutos até a próxima verificação.

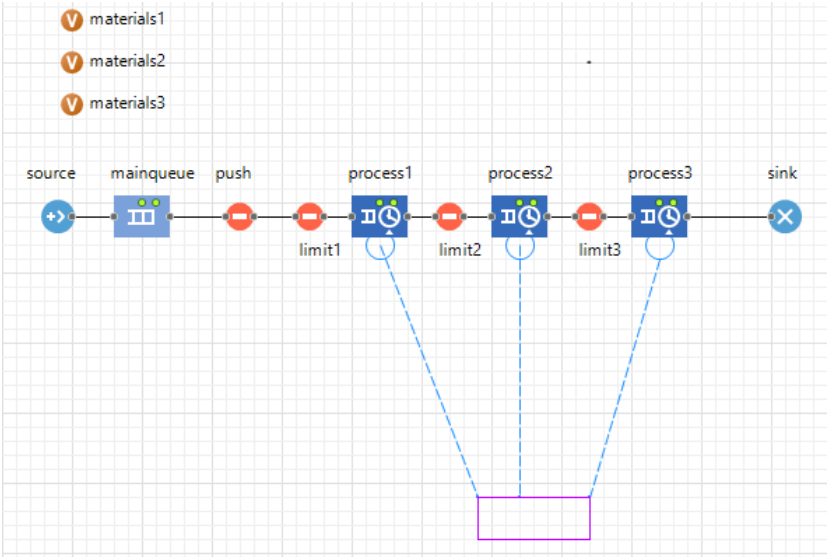


Figura 21– Fluxo principal do quarto caso
Fonte: Autor

De forma similar ao caso anterior, este par de modelos também precisou ser validado para poder ser usado. Por também se tratar de um modelo hipotético, ele foi validado em relação às duas versões do modelo, que precisaram provar estatisticamente representar o mesmo sistema. A validação se deu através do mesmo teste *two-sample t* apresentado anteriormente, neste caso buscando verificar se o número total de peças produzidas após 4 dias seriam iguais para os dois modelos, com um grau de significância de 5%. Para se calcular o número de amostras necessário foi utilizada a Equação 1, tal como no caso anterior. Neste caso, o erro utilizado foi de 3 unidades, e a amostra piloto contou com 30 dados para cada modelo, retornando um desvio padrão de aproximadamente 7,5 para o modelo SED e 8,13 para o modelo SH. Assim, a Equação 1 retornou um valor mínimo de amostras igual a 18 para a versão SED e 21 para a versão SH, não sendo necessário coletar mais dados, e estes se encontram nos Apêndices C e D, respectivamente. Assim, o teste *two-sample t* gerou intervalo de confiança para a diferença das médias equivalente a, aproximadamente, (-2,63 ; 5,43) e como este intervalo contém o valor zero, não se pode rejeitar a hipótese de que as médias dos dois modelos são iguais.

A versão SED deste caso precisou de um recurso “*transporter*” (representando a empilhadeira) e um “*resourceTask*” para a mesma, permitindo regular seu comportamento de verificar os pedidos de 30 em 30 minutos. A empilhadeira executa 2 tipos de ação: carregar peças para

reposição e voltar ao descanso, retornando 30 minutos depois. A tarefa de mover as peças recebeu uma prioridade maior, sendo sempre executada se houver algum pedido pendente. Um novo fluxo foi criado, considerando os pedidos como entidades, que usam o recurso empilhadeira para fazê-la mover ao encontro dos processos repondo-os, usando a lógica de "*seize*" e "*release*". Dois blocos de bifurcação foram adicionados para direcionar a empilhadeira ao processo correto. Os elementos exclusivos da versão SED se encontram na Figura 22.

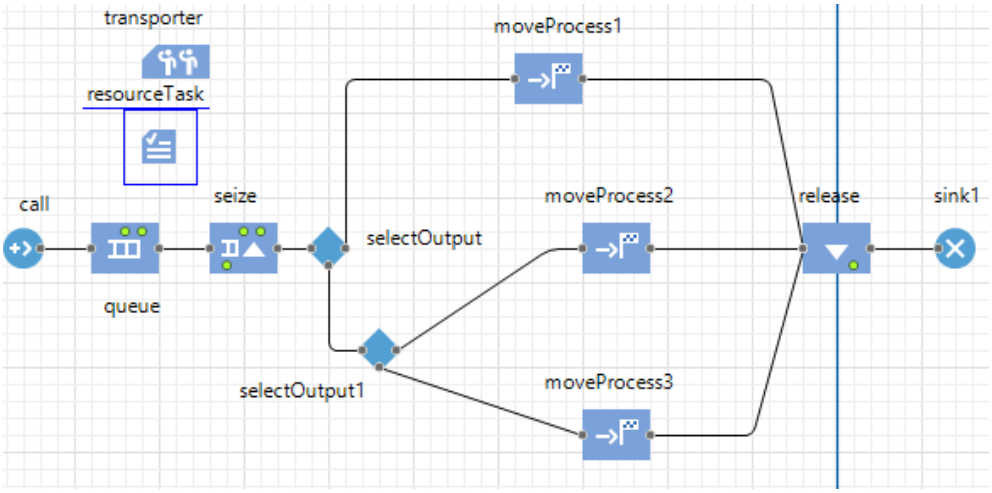


Figura 22– Elementos exclusivos da versão SED – caso 4
Fonte: Autor

Na versão híbrida apenas um elemento foi adicionado à aba principal: o agente "*transporter*", que representa a empilhadeira, e seu diagrama de estados, que regula todo o processo de verificar de 30 em 30 minutos se há algum pedido pendente e se houver, entregá-los dando prioridade aos processos na ordem: 1, 2 e 3. O diagrama de estados do agente "*transporter*" se encontra na Figura 23.

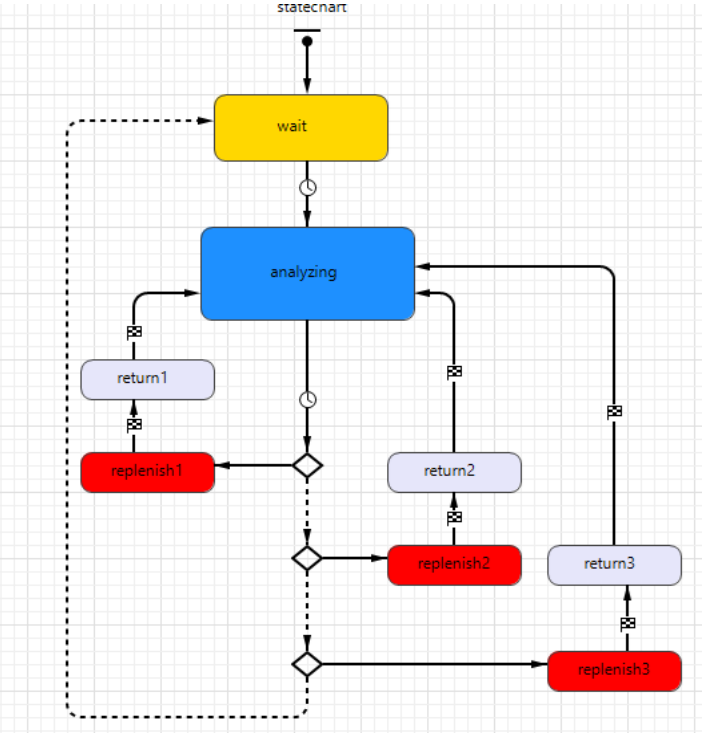


Figura 23- Diagrama de estados da empilhadeira – caso 4
Fonte: Autor

As Tabelas 15 e 16 apresentam o número total de recursos utilizados em cada modelo. Já as Tabelas 17 e 18 apresentam o número de elementos diferentes usados em cada um.

A Tabela 15 indica que **3** linhas de código foram usadas no fluxo principal (servem para injetar pedidos no fluxo secundários da Figura 22 quando necessário), os **42** recursos de *software* usados representam blocos, conectores e mudanças nos blocos usados para criar o fluxo da Figura 22, com o acréscimo de **8** linhas de código. O sistema necessitou de **4** recursos, referentes ao "*resourceTask*" e suas alterações. No total, **57** elementos foram usados neste modelo.

Tabela 15- Recursos usados no quarto caso SED

Recursos usados - SED				
	Processos	Transporte	Sistema	Total
Recursos de Agentes	0	0	0	**0**
Recursos do *Software*	0	42	4	**46**
Linhas de Código	3	8	0	**11**
Total	**57**			

Fonte: Autor

A Tabela 16 mostra que o número de recursos de agentes usados para o modelo SH, equivale a **23**, apresentados na Figura 23. Dentro do mesmo diagrama de estados foram necessárias mais **15** linhas de código, para direcionar o movimento do transportador, repor o estoque, etc. **1** recurso de *software* foi usado para inserir o bloco "*agent*", cujo diagrama de estado foi

mencionado anteriormente (os recursos de agentes calculados são os que se encontram dentro da aba do diagrama de estados, o bloco "*agent*" em si também é considerado um recurso de *software* por se encontrar na aba principal).

Tabela 16- Recursos usados no quarto caso SH

Recursos usados - SH				
	Processos	Transporte	Sistema	Total
Recursos de Agentes	0	23	0	**23**
Recursos do *Software*	0	0	1	**1**
Linhas de Código	0	15	0	**15**
Total	**39**			

Fonte: Autor

As Tabelas 17 e 18 calculam o número de recursos utilizados sem levar em consideração os blocos e comandos repetidos. Mesmo assim, a lógica de transporte no caso da SED se mostrou mais complexa. Na versão SED, como pode ser visto na Tabela 17, o transporte pela empilhadeira utilizou **11** recursos de *software* e **2** linhas de código para ser representado através de um fluxo com bifurcações e comandos de movimentação, que pode ser visto na Figura 22. Para criar a ligação entre os dois fluxos (dos processos e do transporte), **1** tipo de código foi usado e o sistema usou ainda uma "*resource task*" e um "*resource pool*" (**2** recursos de *software* no sistema). No total, este modelo utilizou **16** tipos diferentes de recursos.

Tabela 17- Recursos diferentes do quarto caso SED

Recursos diferentes - SED				
	Processos	Transporte	Sistema	Total
Recursos de Agentes	0	0	0	**0**
Recursos do *Software*	0	11	2	**13**
Linhas de Código	1	2	0	**3**
Total	**16**			

Fonte: Autor

Já o modelo SH representou a empilhadeira introduzindo **1** bloco "*agent*", e seu diagrama de estados contou com **6** tipos de recursos diferentes **4** tipos de códigos (vide Figura 23). A Tabela 18 apresenta os valores desta versão, que totaliza **11** tipos diferentes de recursos usados.

Tabela 18- Recursos diferentes do quarto caso SH

Recursos diferentes - SH				
	Processos	Transporte	Sistema	Total
Recursos de Agentes	0	6	0	**6**
Recursos do *Software*	0	0	1	**1**
Linhas de Código	0	4	0	**4**
Total	**11**			

Fonte: Autor

Pode-se perceber que este foi o único entre os quatro casos onde o modelo SED foi considerado mais complexo em ambas as comparações, sendo aproximadamente 1,46 vezes mais complexo com relação a recursos totais e aproximadamente 1,45 vezes mais complexo com relação ao uso de recursos diferentes. Uma análise comparativa entre os quatro casos será apresentada no item 4.6 deste capítulo.

4.6. Considerações finais

A partir da análise realizada neste estudo de casos múltiplos foi possível perceber que, para os três primeiros casos, a simulação híbrida se demonstrou mais complexa tanto em termos de recursos totais utilizados quanto de recursos diferentes utilizados. O quarto caso foi o único caso em que sua versão SH foi considerada menos complexa em ambas as mensurações.

Como o primeiro caso apresenta um sistema demasiadamente simples, comumente representado em casos de simulação a eventos discretos, seria esperado que a inserção de agentes nesse exemplo só tenha aumentado a complexidade. Já o segundo caso apresenta uma situação mais tipicamente associada a agentes, relativa ao comportamento humano, e neste caso a diferença de complexidade entre as duas versões do modelo foi significativamente menor se comparada ao primeiro caso, apesar que para desenvolver a versão SH deste segundo caso foi necessário o conhecimento de diversas funções dentro do programa.

O terceiro caso apresenta uma situação com uma empilhadeira que realiza dois tipos diferentes de função, o que demandou uma programação complexa ao se transformar esse recurso em um agente. O caminhão, sendo representado como um agente, utilizou menos recursos do que sua contraparte SED, porém o agente “empilhadeira” demandou a maior parte dos recursos do modelo SH, substituindo diversos blocos de processos, porém gerando uma complexidade grande o bastante que não justificou a redução obtida no caminhão e nos blocos de processos.

O quarto caso foi o único onde a inserção de agentes ocasionou redução na complexidade do modelo. É possível perceber que, para este caso, a empilhadeira apresenta um comportamento mais flexível, podendo optar por três caminhos diferentes além de ter seus períodos de inatividade (espera por 30 minutos até conferir os pedidos por materiais). O fato de o uso de agentes reduzir a complexidade para este caso vai ao encontro às afirmações de Dubiel e Tsimhoni (2005) que apontaram a utilidade da SBA para representar quaisquer entidades dentro do modelo que possuam um padrão de movimentos e tomada de decisões mais complexo. Os dois gráficos a seguir, apresentados na Figuras 24 e 25 resumem os resultados desta pesquisa,

ordenando os quatro casos em ordem crescente de complexidade máxima (isto é, de acordo com o modelo mais complexo de cada par). A linha azul representa o valor total de recursos empregados nas versões SED de cada caso enquanto a linha laranja representa o valor total de recursos para as versões SH.

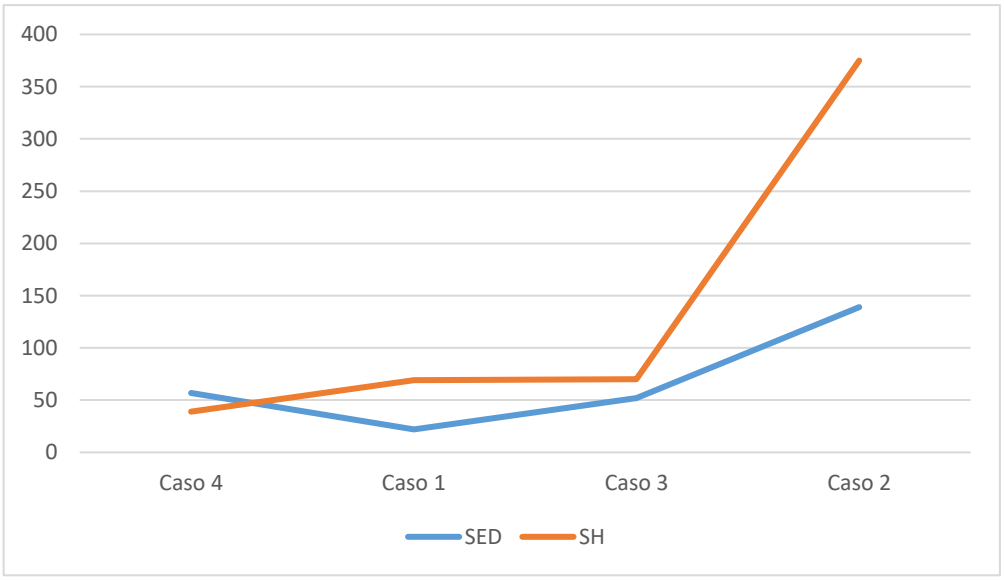


Figura 24– Recursos totais usados em cada caso
Fonte: Autor

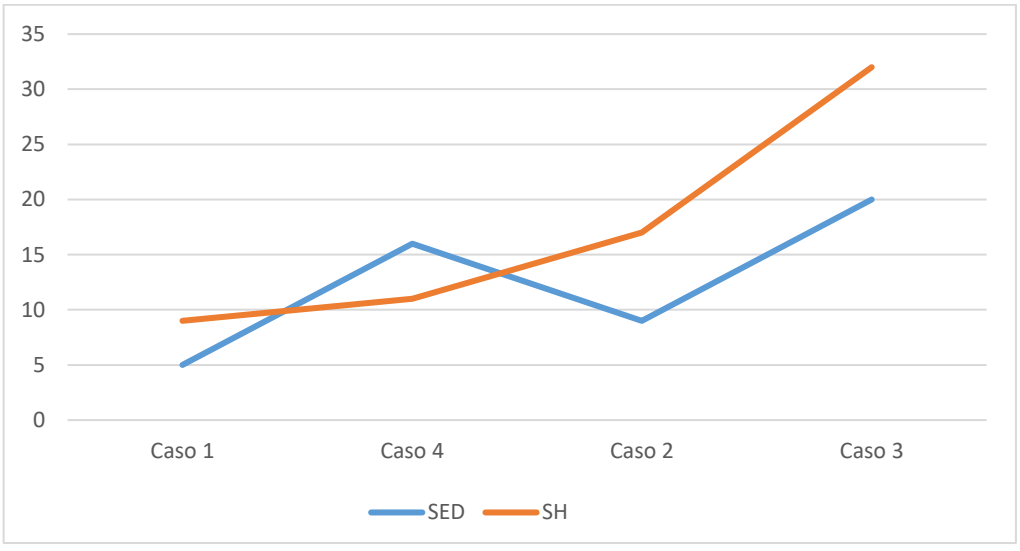


Figura 25– Recursos diferentes usados em cada caso
Fonte: Autor

Como pode ser visto pelo gráfico da figura 24, o caso 4 foi o que demandou um menor número de recursos totais em contrapartida ao caso 2, que na sua versão SH utilizou quase 7 vezes o valor da versão SED do caso 4. Os casos 1 e 3 apresentaram valores intermediários, sendo que

a diferença de complexidade entre versões SED e SH foi mais acentuada nos casos 1 e 2 (especialmente neste último) do que nos casos 3 e 4.

Já a Figura 25 apresenta as medidas relativas ao número diferente de recursos usados em cada modelo. Ao se comparar esse gráfico com o gráfico anterior pode-se observar que nem todos os modelos que apresentaram um grande número total de elementos apresentaram também um grande número de recursos diferentes. Isso se deve ao fato de que em tais modelos muitos blocos e linhas de código foram inseridos repetidas vezes, especialmente no segundo caso. Pela Figura 25 pode-se considerar que, de todos os modelos, a versão SH do caso 3 foi a que mais exigiu conhecimento do *software* e de programação, enquanto a versão SED do caso 1 foi a que menos exigiu conhecimentos específicos.

5. CONCLUSÕES

A presente dissertação tratou do estudo da utilização de dois tipos diferentes de simulação: a simulação a eventos discretos e a simulação híbrida, combinando eventos discretos com agentes. A simulação baseada em agentes obteve uma maior atenção no meio acadêmico a partir dos anos 2000, e é considerada uma abordagem multidisciplinar, sendo aplicada em estudos sociológicos, biológicos, econômicos e sociais, além das engenharias. No ramo da pesquisa operacional, no entanto, a simulação a eventos discretos já estava consagrada como a prática vigente de simulação por mais de 40 anos.

Neste contexto, a simulação baseada em agentes, seja utilizada em conjunto com outras abordagens ou não, apareceu como uma nova proposta dentro da pesquisa operacional, que como dito por Macal (2016) poderia complementar outros tipos de simulação, principalmente na busca de uma representação mais fidedigna do fator humano. Siebers *et al.* (2010) apontaram o crescimento da prática da SBA no meio da gestão de processos, e sugeriram que esta poderia inclusive substituir a simulação a eventos discretos no futuro. Contudo, nem todos os especialistas em simulação concordam com esta afirmação, como reforçado por Brailsford (2014), que afirmou que muitos dos benefícios alegados pelos entusiastas da SBA poderiam ser obtidos através da SED.

Pode-se perceber um *gap* na literatura relacionado à utilização da simulação baseada em agentes, na sua forma pura ou híbrida, a respeito de quando esta deve ou não ser usada no contexto da pesquisa operacional/ gestão de processos. Segundo Mustafee *et al.* (2017), a escolha de se usar uma ou mais técnicas de simulação deve favorecer a eficiência, rapidez e acessibilidade do modelo, o que abre espaço para discussões que tenham como objetivo mensurar as vantagens e desvantagens de cada técnica para cada situação específica.

Entre as características dos modelos de simulação, a complexidade é algo que o modelador deve tentar reduzir ao máximo possível, mantendo seu modelo acessível e eficiente. Chwif *et al.* (2000) apontam que a complexidade dos modelos vem aumentando com o tempo, e que isso traz consequências negativas para a utilização dos mesmos. Neste trabalho a complexidade foi o critério chave para medir a viabilidade de cada técnica de simulação. Como objetivo principal, foi possível comparar o efeito do uso de cada técnica na complexidade do modelo final, através de um estudo de caso aplicado em modelos de manufatura *flow shop*. Como objetivo secundário, foi desenvolvido um *framework* para poder mensurar a complexidade dos modelos, permitindo esta análise.

Como não existe uma definição clara na literatura para o termo “complexidade”, foi necessário definir quais critérios seriam mensurados nos modelos. Segundo a literatura, a complexidade em modelos de simulação está relacionada principalmente ao número de elementos e às relações entre eles, além da dificuldade inerente a desenvolver e interpretar cada modelo. Assim, cada modelo recebeu duas análises: uma associada ao número total de elementos e outra associada à quantidade de recursos diferentes necessários para desenvolvê-lo. Através da análise foi possível perceber que a simulação híbrida, apesar de viabilizar diferentes formas de se abordar tais modelos, apresentou maior complexidade em todos os casos aqui abordados.

Para o caso 1, a versão híbrida apresentou mais do que o triplo da complexidade da versão SED em termos de recursos totais, já que o caso consistia em uma linha de produção tipicamente associada à simulação a eventos discretos e a inserção de agentes apenas aumentou sua complexidade. Em termos de recursos diferentes utilizados, a versão SH apresentou quase o dobro que sua contrapartida SED. O segundo caso apresentou um modelo muito maior em termos de elementos totais utilizados, com relação aos outros 3 casos. Isso se deve à inserção da tabela Westinghouse para regular o ritmo de produção dos operadores, cuja implementação necessitou de um grande número de parâmetros, variáveis, tabelas e linhas de código. Em termos de recursos diferentes utilizados, a versão SH apresentou um pouco menos que o dobro da sua contrapartida. Porém, em termos de recursos totais usados, a versão SH atingiu 375 recursos totais, concentrados no diagrama de estados dos agentes que necessitou da repetição de diversos comandos e códigos para regular a transição do operador entre os diversos estados e alterar seus parâmetros de acordo com a metodologia Westinghouse. O mesmo efeito foi obtido com o uso de 139 recursos totais na versão SED.

Já o caso 3 foi aquele onde a sua versão SH exigiu o maior número de recursos diferentes, entre blocos, funções e códigos dentro do Anylogic®, mas sua versão SED também apresentou uma quantidade considerável de recursos diferentes usados. A diferença de recursos totais utilizados (70 da versão SH contra 57 da versão SED) entre as duas versões foi menos acentuada que nos dois casos anteriores. Por fim, o quarto e único caso entre eles onde a versão SH foi considerada menos complexa em ambas as análises apresentou uma diferença pouco acentuada entre as duas versões, para ambas as medições (39 contra 57 para recursos totais e 11 contra 16 para recursos diferentes). Através da análise deste caso foi possível perceber que a inserção de agentes em modelos SED pode, de fato, contribuir para a redução da complexidade destes, indo ao encontro da afirmação de Macal (2016) que diz que a simulação baseada em agentes pode complementar outros tipos de simulação.

Espera-se que este estudo contribua com a discussão sobre a viabilidade da utilização de agentes em modelos de simulação de manufatura. Apesar da simulação híbrida ter apresentado uma maior complexidade em três dos quatro casos aqui abordados, o quarto caso serviu como um exemplo de que esta pode complementar a simulação a eventos discretos em determinadas situações. Além disso, seu uso pode abrir novas possibilidades de se representar as diversas situações encontradas nos modelos de simulação. É importante lembrar que este estudo foi direcionado a problemas do tipo manufatura *flow shop*, e portanto não se sabe qual seria o impacto da simulação híbrida na complexidade de outros tipos de outros tipos de modelos, o que abre margem para futuros trabalhos. Também como sugestão para futuros trabalhos fica a ideia de se tentar aprimorar e aprofundar o *framework* disponibilizado aqui.

APÊNDICE A - SAÍDAS DO TERCEIRO CASO SED

195	199	186	195	185	189
200	205	188	203	207	193
189	197	189	191	182	205
198	180	191	200	180	191
204	187	183	196	203	188

APÊNDICE B - SAÍDAS DO TERCEIRO CASO SH

200	204	214	194	208	198
183	183	213	200	209	191
181	185	184	177	173	198
204	190	204	203	190	209
181	180	193	197	199	188

APÊNDICE C - SAÍDAS DO QUARTO CASO SED

66	60	74	64	64	69
80	76	72	65	67	64
61	66	78	79	77	67
79	76	71	67	66	58
79	68	93	74	71	70

APÊNDICE D - SAÍDAS DO QUARTO CASO SH

88	61	71	67	56	59
70	71	68	55	75	62
74	81	79	72	67	73
80	78	66	77	67	65
61	74	60	72	73	57

REFERÊNCIAS:

AHMED, R.; SHAH, M.; UMAR, M. Concepts of Simulation Model Size and Complexity. **International Journal of Simulation Modelling,** v. 15, n. 2, p. 213-222, 2016.

AXELROD, R. Advancing the art of simulation in the social sciences. **Complexity**, v.3 n.2 p. 16-22, 1997.

BAINES, T. S.; ASCH, R.; HADFIELD, L.; MASON, J. P.; FLETCHER, S.; KAY, J. M. Towards a theoretical framework for human performance modeling within manufacturing systems design. **Simulation Modelling Practice and Theory**, n.13, p. 486-504, 2005.

BARBIERI, J. P. Análise da Representação do Fator Humano Presente em um Sistema de Manufatura Através da Simulação Híbrida. 102p. Dissertação (Mestrado) – Universidade Federal de Itajubá, Itajubá, 2016.

BARNES, R. M. **Estudos de movimentos e de tempos: projetos e medidas do trabalho.** 6. ed. São Paulo: Edgard Blücher, 1977.

BERNARDO, M. M. Comparação da Complexidade de Modelos Híbridos e Discretos no Ramo da Manufatura. 33 f. Dissertação (Mestrado em Engenharia de Produção) - Programa de Pós-Graduação em Engenharia de Produção, Itajubá, MG, UNIFEI, 2017.

BONABEAU, E. Agent-based modeling: methods and techniques for simulating human systems. **Proceedings of the National Academy of Sciences**, v. 99, n. 3, p. 7280-7287, 2002.

BORYS, M.; JUSZCZYK, M.; MILOSZ, E.; MILOSZ, M.; MURIJAS, P.; PANKZIC, B. **Advanced Object-Oriented Technology**. Polish Information Processing Society, 2010.

BRAILSFORD, S. Discrete-event simulation is still alive and kicking! **Journal of Simulation,** v. 8, n. 1, p. 1-8, 2014.

BROOKS, R. J.; TOBIAS, A. M. Choosing the Best Model: Level of Detail, Complexity and Model Performance. **Mathematical and Computer Modelling,** v. 24, n. 4, p. 1-14, 1996.

BULTMANN, M.; KNUST, S.; WALDHERR, S. Synchronous flow shop scheduling with pliable jobs. **European Journal of Operational Research,** v. 270, p. 943-956, 2018.

CAUCHICK MIGUEL, P. A. Estudo de caso na engenharia de produção: estruturação e recomendações para sua condução. **Produção,** v. 17, n. 1, p. 216-229, 2007.

CAUCHICK MIGUEL, P. A.; FLEURY, A.; MELLO, C. H. P.; NAKANO, D, N.; TURRIONI, J. B.; LEE HO, L.; MORABITO, R.; MARTINS, R. A.; PUREZA, V. **Metodologia de pesquisa em engenharia de produção e gestão de operações.** Rio de Janeiro, Elsevier, 2010.

CHAN, W. K. V.; SON Y. J.; MACAL, C. M. Agent-Based Simulation Tutorial – Simulation of Emergent Behavior and Differences Between Agent-Based Simulation and Discrete-Event Simulation. In: Winter Simulation Conference, **Proceedings...** Baltimore, MD, USA, 2010.

CHWIF, L.; MEDINA, A. C. **Modelagem e Simulação de Eventos Discretos – Teorias e Aplicações.** 3 ed. São Paulo, 2010.

CORRÊA, H. L.; CORRÊA, C. A. **Administração de Produção e de Operações.** São Paulo, Atlas, 2005.

DELA SAVIA, T.; PINHO, A. F.; BARBIERI, J. P.; BERNARDO, M. M. Comparação da complexidade entre modelos de simulação a eventos discretos e híbridos no ramo da manufatura. In: L Simpósio Brasileiro de Pesquisa Operacional, **Anais...** Rio de Janeiro, RJ, 2018.

DUBIEL, B.; TSIMHONI, O. Integrating Agent Based Modeling into a Discrete Event Simulation. In: Winter Simulation Conference, **Proceedings...** Orlando, FL, USA, 2005.

EDMONDS, B. What is complexity? The philosophy of complexity per se with application to some examples in evolution. 1998. Disponível em: https://pdfs.semanticscholar.org/84d1/a3a5b8d3d8865cd2bd2b1d7e387ce68bd708.pdf?_ga=2.207015699.1288186792.1542844305-1162842931.1542844305, acesso 21, nov 2018.

EFTHYMIOU, K.; PAGOROPOULOS, A.; PAPAKOSTAS, N.; MOURTZIS, D.; CHRYSSOLOURIS, G. Manufacturing systems complexity: An assessment of manufacturing performance indicators unpredictability. **CIRP Journal of Manufacturing Science and Technology**, v.7, pp.324-334, 2014.

EISENHARDT, K. M. Building Theories from Case Study Research. **Academy of Management Review**, v. 14, n. 4, p. 532-550, 1989.

ELDABI, T.; BALABAN, M.; BRAILSFORD, S.; MUSTAFEE, N.; NANCE, R. E.; ONGGO, B. S.; SARGENT, R. G. Hybrid Simulation: Historical Lessons, Present Challenges and Futures. In: Winter Simulation Conference, **Proceedings...** Arlington, VA, USA, 2016.

ELDABI, T.; BRAILSFORD, S.; DJANATLIEV, A.; KUNC, N.; MUSTAFEE, N.; OSORIO, A. F. Hybrid simulation challenges and opportunities: a life-cycle approach. In: Winter Simulation Conference, **Proceedings...** Gothenburg, Sweden, 2018.

ELMARAGHY W.; ELMARAGHY, H.; TOMIYAMA, T.; MONOSTORI, L. Complexity in engineering design and manufacturing. In: CIRP Manufacturing Technology, **Proceedings...** 2012.

ELKOSANTINI, S. Toward a new generic behavior model for human centered system simulation. **Simulation Modelling Practice and Theory**, v. 52, p. 108-122, 2015.

FAKHIMI, M.; ANAGNOSTOU, A.; STERGIOULAS, L.; TAYLOR, S. J. E. A Hybrid Agent-based and Discrete Event Simulation Approach For Sustainable Strategic Planning and Simulation Analytics. In: Winter Simulation Conference, Proceedings... Huntington Beach, CA, USA, 2014.

FORRESTER, J. W. Industrial Dynamics-After the First Decade. **Management Science**, v. 14, n. 7, p. 398-415, mar 1968.

FUCHIGAMI, H. Y.; RANGEL, S. Métodos heurísticos para maximização do número de tarefas just-in-time em flow shop permutacional. In: XLVII Simpósio Brasileiro de Pesquisa Operacional, **Anais...** Porto de Galinhas, PE, 2015.

GAVIRA, M. O. Simulação computacional como uma ferramenta de aquisição de conhecimento. 163p. Dissertação (Mestrado) – Universidade de São Paulo, São Carlos, 2003.

GREEN, J. J.; KREJCI, C. C.; CANTOR, D. E. A hybrid simulation model of helping behavior. In: Winter Simulation Conference, **Proceedings...** Las Vegas, NV, USA, 2017.

HAO, Q.; SHEN, W. Implementing a hybrid simulation model for a Kanban-based material handling system. **Robotics and Computer-Integrated Manufacturing,** v. 24, p. 635-646, 2008.

HARRISON, J. R.; LIN, Z.; CARROL, G. R.; CARLEY, K. M. Simulation Modeling in Organizational and Management Research. **Academy of Management Review,** v. 32, n. 4, p. 1229-1245, 2007.

HERNANDEZ-MATIAS, J.C.; VIZAN, A.; PEREZ-GARCIA, J.; RIOS, J. An integrated modeling framework to support manufacturing system diagnosis for continuous improvement. **Robotics and Computer-integrated manufacturing**, v. 24, n. 2, p. 187-199, 2008.

HORDONES, P. A.; CAMARGO, V. H.; FUCHIGAMI, H. Y. Programação da produção em *flow shop* permutacional envolvendo medidas de atraso: uma contribuição bibliométrica. In: XLVIII Simpósio Brasileiro de Pesquisa Operacional, **Anais...** Vitória, ES, 2016.

INGALLS, R. G.; Introduction to Simulation. In: Winter Simulation Conference, **Proceedings...** Miami, FL, USA, 2008.

JACOB, M. Discrete Event Simulation. **Resonance**, p.78-86, 2013.

JAHANGIRIAN, M. T.; ELDABI, A.; NASEER.; STERGIOULAS, L. K.; YOUNG, T. Simulation in Manufacturing and Business: A Review. **European Journal of Operational Research,** v. 203, p. 1-13, 2010.

KELTON, D.W.; SADOWSKI, P.R.; Sturrock, D.T. **Simulation with Arena.** 4. Ed. McGraw-Hill, 2007.

KITTIPITTAYAKORN, C.; YING, K. C. Using the Integration of Discrete Event and Agent-Based Simulation to Enhance Outpatient Service Quality in na Orthopedic Department. **Journal of Healthcare Engineering.** 8 p., 2016.

KRAJEWSKI, L.; RITZMAN, L.; MALHOTRA, M. **Administração da Produção e Operações.** 8 ed. São Paulo, Pearson Prentice Hall, 2009.

LAW, A. M.; KELTON, D. W. **Simulation, modeling & analysis**. 2 ed. New York, McGraw-Hill, 1991.

LEAL, F.; ALMEIDA, D. A.; MONTEVECHI, J. A. B. Uma Proposta de Técnica de Modelagem Conceitual para a Simulação Através de Elementos do IDEF. In: XL Simpósio Brasileiro de Pesquisa Operacional, **Anais...** João Pessoa, PB, BR, 2008.

LEITÃO, P. Agent-based distributed manufacturing control: A state-of-the-art survey. **Engineering Applications of Artificial Intelligence**, v. 22, p. 979–991, 1999.

LOBÃO, E. C.; PORTO, A. J. V.; Evolução das técnicas de simulação. **Produção,** v.9, n.1, p. 13-21, 1999.

MACAL, C. M. Everything you need to know about agent-based modelling and simulation. **Journal of Simulation**, v.10, n.02, pp.144-156, 2016.

MACAL, C. M.; NORTH, M. J. Introductory Tutorial: Agent-Based Modeling and Simulation. In: Winter Simulation Conference, **Proceedings...** Washington, D. C., USA, 2013.

MAIONE, G.; NASO, D. Modelling adaptive multi-agent manufacturing control with discrete event system formalism. International Journal of Systems Science, v. 35, n.10, p. 591-614, 2004.

McLEISH, D. L. **Monte Carlo Simulation & Finance.** Wiley Finance.

MINGERS, J.; BROCKLESBY, J. Multimethodology: Towards a Framework for Mixing Methodologies. **Omega,** v. 25, n. 5, p. 489-509, 1997.

MITTAL, A. "Hybrid Simulation Modeling for Regional Food Systems". M.S. Thesis. Ames, IA: Department of Industrial and Manufacturing Systems Engineering, Iowa State University, 2016.

MONTEVECHI, J. A. B.; LEAL, F.; PINHO, A. F.; COSTA, R. F. S.; OLIVEIRA, M. L. M.; SILVA, A. L. F. Conceptual modeling is simulation projects by mean adapted IDEF: na application in a brazilian tech company. In: Winter Simulation Conference, **Proceedings...** Baltimore, M. D., USA, 2010.

MOURA, Reinaldo A. **Armazenagem: Do Recebimento à Expedição em Almoxarifados ou Centros de Distribuição.** 5. Ed. São Paulo: Instituto IMAM, 2008. v.2

MORR, D. K. Lifting the fog of complexity. **Science,** v. 343 n. 6169 p. 382 - 383, 2014.

MUSTAFEE, N.; BRAILSFORD, S.; DJANATLIEV, A.; ELDABI, T.; KUNC, M.; TOLK, A. Purpose and Benefits of Hybrid Simulation: Contributing to the Convergence of its Definition. In: Winter Simulation Conference, **Proceedings...** Las Vegas, NV, USA, 2017.

NANCE, R. E.; SARGENT, R. G. Perspectives on the evolution of simulation. **Operations Research**, v.50, n.1, p.161-172, 2002.

NORTH, M. J.; MACAL, C. M. **Managing Business Complexity**. Oxford University Press, 2007.

PIDD, M. Simulation worldviews - so what? In: Winter Simulation Conference, **Proceedings...** Washington, DC, USA, 2004.

POPOVICS, G.; MONOSTORI, L. An approach to determine simulation model complexity. **Procedia CIRP,** v. 52, p. 257-261, 2016.

WHITE Jr, K. P.; INGALLS, R. G. The Basics of Simulation. In: Winter Simulation Conference, **Proceedings...** Gothenburg, Sweden, 2018.

ROSS, S. M. **Simulation.** 4. ed. Academic Press, Elsevier, 2006.

SAKURADA, N.; MIYAKE, D. I. Simulação baseada em agentes (SBA) para modelagem de sistemas de operações. In: XII Simpósio de Administração da Produção, Logística e Operações Internacionais, **Anais...** São Paulo, SP, 2009.

SAMUELSON, D. A.; MACAL, C. M. Agent-Based Simulation Comes of Age: *Software* opens up many new areas of application. **OR/MS Today.** v. 33, N. 4, 2006.

SANCHEZ, M. S.; LUCAS, T. W. Exploring the World of Agent-Based Simulations: Simple Models, Complex Analyses. In: Winter Simulation Conference, **Proceedings...** San Diego, CA, USA, 2014.

SARGENT, R. G. Validation and Verification of Simulation Models. In: Winter Simulation Conference, **Proceedings...** Washington, D.C., USA, 2014.

SARJOUGHIAN, H. S. Restraining complexity and scale traits for component-based simulation models. In: Winter Simulation Conference, **Proceedings...** Las Vegas, NV, USA, 2017.

SCHRIBER, T. J.; BRUNNER, D. T.; SMITH, J. F. Inside discrete-event simulation software: how it works and why it matters. In: Winter Simulation Conference, **Proceedings...** Huntington Beach, CA, USA, 2014.

SHANNON, R. E. **Systems Simulation: the art and science**. Englewood Cliffs: Prentice-Hall, 1975.

SHANNON, R. E. Introduction to the Art and Science of Simulation. In: Winter Simulation Conference, **Proceedings...** Washington, D.C., USA, 1998.

SHARMA, P. Discrete-Event Simulation. **International Journal of Scientific and Technology Research**, v. 4, n. 2, p. 136-140, 2015.

SIEBERS, P. O.; MACAL, C. M.; GARNETT, J.; BUXTON, D; PIDD, M. Discrete-event simulation is dead, long live agent-based simulation! **Journal of Simulation**, v. 4, n. 2, p. 204-210, 2010.

SIMON, H. A. The architecture of complexity. **American Philosophical Society,** v. 106, n. 6, p. 467-482, 1962.

SLACK, N.; CHAMBERS, S.; JOHNSTON, R. **Administração da Produção.** 2ª Ed. São Paulo: Atlas, 2002.

SWAIN, J.J. New Frontiers in Simulation: Biennial survey of discrete-event simulation software tools. OR/MS Today, v. 34, 2007.

VAN MIERLO, S.; VANGHELUWE, H. Introduction to statecharts modeling, simulation, testing, and deployment. In: Winter Simulation Conference, **Proceedings...** Gothenburg, Sweden, 2018.

VIANA, J.; BRAILSFORD, S. C.; HARINDRA, V.; HARPER, P. R. Combining discrete-event simulation and system dynamics in a healthcare setting: A composite model for Chlamydia infection. **European Journal of Operational Research,** v. 237, p. 196-206, 2014.

VIANA, J.; SIMONSEN T. B.; DAHL, F. A.; FLO, K. A hybrid discrete event agent based overdue pregnancy outpatient clinic simulation model. In: Winter Simulation Conference, **Proceedings...** Gothenburg, Sweden, 2018.

VOSS, C.; TSIKRIKTSIS, N.; FROHLICH, M. Case research in operations management. **International Journal of Operations & Production Management,** v. 22, n. 2, p. 195-219, 2002.

WALDROP, M. M. **Complexity - The Emerging Science at the Edge of Order and Chaos.** 1992.

YIN, R. K. **Estudo de caso – planejamento e métodos.** 2. Ed. Rio de Janeiro, Bookman, 2001.

YUCESAN, E.; SCHRUBEN, L. Complexity of Simulation Models: A Graph Theoretic Approach. **Journal on Computing,** v.10, n.1, p.