Ângelo Miguel Henriques Veiga

Licenciatura em Engenharia Eletrotécnica e de Computadores

A Multiagent based Shop Floor Transportation System Simulator

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e de Computadores

Orientadores : José António Barata de Oliveira, Professor Doutor, FCT-UNL
Luís Domingos Ferreira Ribeiro, Doutor, FCT-UNL

Júri:

Presidente:	Doutor Pedro Alexandre da Costa Sousa
Arguente:	Doutor Tiago Oliveira Machado de Figueiredo Cardoso
Vogais:	Doutor José António Barata de Oliveira Doutor Luís Domingos Ferreira Ribeiro

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Março, 2013

A Multiagent based Shop Floor Transportation System Simulator

Para a Mafalda Patriarca e para a minha Família

Agradecimentos

Este trabalho marca o fim de uma jornada que começou em 2007, sendo que até 2013, estudei Engenharia Eletrotécnica e de Computadores e tive muitas pessoas que contribuíram de uma forma ou de outra para eu atingir este ponto da minha vida, e que sempre me vou lembrar.

Em primeiro lugar, gostava de agradecer à Faculdade de Ciências e Tecnologia, em particular, à Universidade Nova de Lisboa, pela qualidade de Ensino e dedicação aos alunos que a frequentam.

Queria expressar um reconhecimento muito especial para o Doutor Luís Ribeiro, pois sempre foi uma pessoa motivadora e disponível para eu atingir os meus objetivos durante o desenvolvimento deste trabalho, e por isso tenho-o em muita estima e agradecimento e não existe nada que eu possa dizer para agradecer a sua amizade.

Também quero agradecer ao Doutor José Barata por ser o meu orientador e pela oportunidade de desenvolver o meu interesse na área de Robótica e Manufatura que me deu.

Também quero agradecer aos meus colegas de laboratório Rogério Rosa, Mauro Dias e André Rocha e a todos aos meus colegas de faculdade que me acompanharam ao longo do meu curso.

Por último, agradeço à minha família e à Mafalda Patriarca por terem sido as pessoas que me tornaram na pessoa que sou hoje. Sempre me amaram incondicionalmente e ensinaram princípios de vida muito importantes que modelaram a minha personalidade.

Resumo

Atualmente, as empresas são obrigadas a adaptarem-se aos novos paradigma de produção. Por exemplo, a criação e desenvolvimento de novos meios para criar produtos personalizados com ciclos de produção curtos e a baixo custo, enquanto se mantêm os mesmos níveis de produtividade e qualidade. Isto gerou a necessidade de criar sistemas de manufatura cada vez mais ágeis e flexíveis, de modo a acomodaram os novos paradigmas de produção.

Os novos paradigmas de produção, potenciados pelos avanços das tecnologias de informação, **TI**, levam à aceitação do conceito de sistemas multiagentes e das tecnologias relacionadas. Os sistemas multiagentes visam alcançar o desenvolvimento de módulos cujas funções individuais e coletivas adaptam-se e evoluem de forma a assegurarem a aptidão dos sistemas de produção no tratamento de oportunidades de negócio voláteis mas rentáveis.

Com o aumento de componentes autónomos e distribuídos que interagem entre si na execução de projetos, as ferramentas de simulação tradicionais tornar-se-ão insuficientes.

O presente trabalho apresenta a implementação de uma arquitetura baseada em agentes e orientada às interações que implementa o conceito Simulação e que serve como suporte para a introdução de uma nova ferramenta de simulação que tem a capacidade de interligar com qualquer sistema de transporte capaz de gerir qualquer linha de manufatura, e fornecer uma análise estatística do sistema.

Palavras-chave: Ferramenta de simulação, Sistema de Manufatura, Sistema Multiagente, Simulação.

Abstract

Currently, companies are forced to adapt to new production policies. For example, the creation and development of new means to create customized products with short production cycles and low cost, while maintaining the same levels of productivity and quality. This generated the need for manufacturing systems agile and flexible in order to accommodate the new production policies.

The new paradigms of production, boosted by advances in information technology, **IT**, accept the concept of multi-agent systems and related technologies. The multi-agent systems seek to achieve the development of modules whose individual and collective functions adapt and evolve in order to ensure the suitability of production systems in the treatment of business opportunities volatile but profitable.

With the rise of distributed and autonomous components interacting in project execution, traditional simulation tools will become insufficient.

This paper presents the implementation of an architecture based in agents and oriented to interactions that implements the concept of simulation and serves as a support for introducing a new simulation tool that has the capability to interconnect with any transportation system able to manage all manufacturing line, and provide a statistical analysis of the system.

Keywords: Simulation Tool, Manufacturing System, Multi-Agent System, Simulation.

Conteúdo

Lista de Figuras

Lista de Tabelas

Listagens

1

Introdução

Face ao recente conceito de produção personalizável - um paradigma de excelência na indústria e serviços - novos desafios socioeconómicos surgiram e, com eles, novas fronteiras na concorrência empresarial e consequentemente novos objetivos ao nível da política de produção; traduzidos de forma sucinta num enorme aumento da quantidade, variedade e personalização dos bem produzidos sem um correspondente aumento dos custos despendidos.

Assim, de forma a dar resposta às novas exigências do mercado e tornar os sistemas de produção mais flexíveis, ágeis e sustentáveis, verifica-se atualmente uma tendência para o distanciamento de paradigmas de automação centralizada - mais propensos a falhas e de limitada readaptação face às mesmas - e uma aproximação de paradigma de automação distribuída.

É nesta nova linha de pensamento que surge o desenvolvimento de Sistemas de Multiagente num contexto Mecatrónico.

Segundo o **IRDAC**, *Comité Assessor para Pesquisa e Desenvolvimento Industrial da Comunidade Europeia*, a **Mecatrónica é a integração sinergética da engenharia mecânica com a eletrónica e com o controlo inteligente por computador no projeto de processos e de manufatura de produtos** [2] [3].

Este trabalho tem como objetivo o desenvolvimento de uma ferramenta de simulação, utilizando o paradigma Multiagente, capaz de interligar qualquer sistema de controlo ou transporte de um sistema de manufatura. Para além da apresentação gráfica do sistema, este trabalho propõe-se apresentar graficamente os vários resultados numéricos relevantes para a gestão eficiente de qualquer sistema a ser simulado.

1.1 Integração com Projetos de Desenvolvimento

O trabalho proposto foi desenvolvido no âmbito de um projeto europeu denominado *Instantly Deployable Evolvable Assembly Systems*, **FP7 NMP IDEAS**. Como esta ferramenta de simulação permite a interligação de qualquer sistema de controlo, a gestão e controlo da linha de manufatura a simular será realizado através do sistema de transporte deste projeto.

1.2 Principais Contribuições

Esta ferramenta de trabalho contribui para a área de simulação e de manufatura através dos seguintes pontos:

1. Consiste numa ferramenta de simulação baseada em agentes, o que permite efetuar uma distribuição uniforme dos agentes em diversas plataformas.
2. Permite interligar outros sistemas de controlo externos baseados em agentes.
3. Por ser uma ferramenta de simulação baseada em agentes e por ser possível a interligação de um sistema de transporte externo, permite validar e depurar a arquitetura do **IDEAS** e seu sistema de transporte.

1.3 Descrição do Documento

Este documento encontra-se organizado em cinco capítulos principais: Introdução, Estado de Arte, Arquitetura do Trabalho, Implementação do Trabalho, Casos de Usos e Conclusão e Trabalhos Futuros.

O segundo capítulo designado por **Estado de Arte** contextualiza o trabalho desenvolvido de acordo com os paradigmas mais recentes da manufatura, como também as ferramentas de simulação desenvolvidas, discutindo também a sua importância no mundo atual.

De seguida, o capítulo **Arquitetura do Trabalho** descreve a arquitetura do trabalho baseando-se no paradigma Multiagente e na área de simulação.

O quarto capítulo designado por **Implementação do Trabalho** descreve a implementação da arquitetura ilustrada no capítulo anterior, apresentando a ferramenta desenvolvida deste trabalho.

O capítulo seguinte designado por **Casos de Usos** ilustra e explica os resultados obtidos através das simulações de três sistemas diferentes.

Por último, o capítulo **Conclusão e Trabalhos Futuros** serve para enumerar as conclusões obtidas na realização deste trabalho e apontar novas inovações para o futuro do mesmo.

2

Estado da Arte

Este capítulo tem como objetivo ilustrar a literatura existente sobre o âmbito em que o trabalho desenvolvido se enquadra.

Este capítulo encontra-se subdividido em quatro subcapítulos. No primeiro subcapítulo é apresentada a definição de Simulação e o seu papel no âmbito da Indústria. No subcapítulo seguinte, são apresentadas e descritas algumas ferramentas de simulação de sistemas de manufatura tradicionais e ainda são enumeradas e explicadas as suas limitações face a sistemas baseados em agentes. No terceiro subcapítulo, são explicadas as características de um Agente no âmbito da Indústria e ainda são apresentadas algumas ferramentas de simulação baseadas em agentes. E no último subcapítulo, é apresentada a importância deste trabalho e a sua contextualização na arquitetura do projeto **IDEAS**, *Instantly Deployable Evolvable Assembly Systems*.

2.1 Simulação

A Simulação permite abstrair um sistema real e estudá-lo num ambiente virtual, onde o seu comportamento pode ser explorado de uma forma mais rápida, segura e sem custos associados ao teste do sistema físico [4].

Dito por outras palavras, a Simulação consiste numa ferramenta que permite realizar um conjunto de experiências num determinado modelo de um sistema real ou abstrato, para que se possa proceder à sua avaliação e realizar melhorias no seu desempenho [5].

Um modelo é a representação do sistema real ou abstrato e tem como objetivo estabelecer relações entre as variáveis de entrada e de saída, como se pode observar na Figura 2.1.

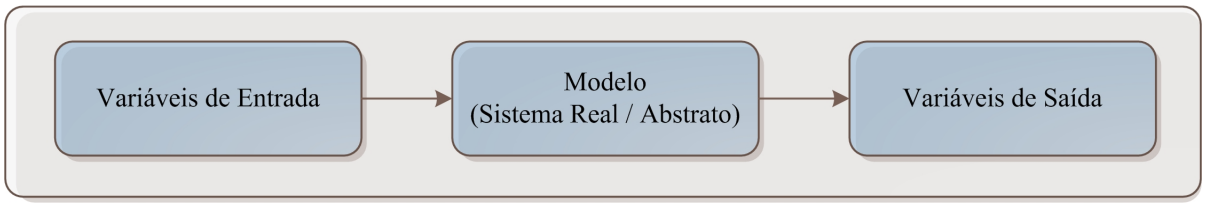

Figura 2.1: Definição de Modelo

Existem duas abordagens na utilização de um modelo de um sistema: **exploratória** e **preditiva**.

A abordagem exploratória visa explorar e criar hipóteses sobre um modelo, ou seja, não se pretende prever o seu comportamento futuro, mas sim criar uma visão geral na qual observações passadas podem ser entendidas como parte de um processo mais amplo. Esta abordagem geralmente foca-se num aspeto específico de um sistema, colocando ênfase em alguns detalhes sobre um acontecimento e ignorando outros. Existe um exemplo de uma simulação de uma colónia de formigas com o uso de uma abordagem exploratória [6].

A abordagem preditiva é usada para observar tendências, avaliar cenários e predizer estados futuros. É uma abordagem utilizada para realizar uma mímica de sistemas do mundo real, e também muito útil para o desenvolvimento de cenários e decisões políticas. Existe um exemplo de uma simulação de epidemias da doença dengue através do uso de diversos parâmetros para caracterizar o avanço da doença [7].

O uso da Simulação permite a deteção de erros durante a fase de análise, pré-implementação e mesmo durante a sua manutenção ou reprogramação. Também proporciona a vantagem de a simulação de um sistema como um todo, ou apenas de uma parte do sistema, permitindo a sua depuração e a sua validação sem que haja necessidade de utilizar equipamentos físicos/reais.

A Simulação permite ainda uma fácil reprodução de diferentes cenários, em condições anormais ou eventualmente até perigosas no mundo real e no entanto seguras no mundo virtual. Os dados recolhidos da simulação podem ser reutilizados para treino de operadores e manutenção do sistema. E, as simulações podem ser repetidas tantas vezes quanto o necessário para uma correta compreensão e ajuste dos sistemas de manufatura e podem ser comprimidas (mediante a aceleração do intervalo de tempo), tornando mais rápida a obtenção de resultados que num ambiente real levariam muito mais tempo.

Resumindo, a Simulação permite descobrir e formalizar novas teorias e modelos, testar as hipóteses dos mesmos, entender as características do sistema, prever comportamentos futuros, analisar e detetar elementos críticos, etc., mas nunca substituindo a interpretação humana [8] [9].

2.2 Ferramentas de Simulação

Atualmente, é possível distinguir dois tipos de simulação que as ferramentas permitem efetuar: **DES**, *Discrete Event Simulation* e **GS**, *Geometric Simulation* [10].

O **DES** consiste na simulação dos comportamentos de todas as entidades do sistema, também descritas como variáveis. Estes comportamentos ocorrem durante um evento numa determinada unidade de tempo de vida do sistema distinta [11]. Esse evento consiste na alteração de estados das variáveis do sistema. Sendo um evento discreto não ocorre linearmente no tempo mas sim em intervalos de tempo irregulares [12]. Através da Tabela 2.1 é possível verificar algumas das ferramentas de simulação atuais que possuem este tipo de comportamento.

O **GS** consiste na simulação geométrica, ou física, de uma simples entidade ou de um sistema complexo, geralmente em três dimensões. A ferramenta de simulação constrói um modelo virtual do sistema de manufatura que inclui o equipamento físico e também o controlo lógico. Através da Tabela 2.2 é possível verificar algumas das ferramentas de simulação atuais que possuem este tipo de comportamento.

Outra maneira de classificar as ferramentas de simulação é através de dois conceitos: **Linguagem** e **Simuladores**.

Uma **Linguagem** consiste numa camada de *Software* em que o seu modelo de desenvolvimento consiste na programação. A vantagem deste conceito é que a modelação é flexível mas que também se traduz numa desvantagem, pois é necessário um conhecimento em programação [13].

Um **Simulador** consiste numa camada de simulação que permite a construção de um modelo de um sistema de manufatura dentro de uma determinada classe de tipo de sistema. Este tipo de ferramenta possui duas grandes características: o seu âmbito consiste na modelação de elementos da manufatura e não é preciso ter um grande conhecimento de programação para a construção do modelo.

Tabela 2.1: Ferramentas de Simulação de Eventos Discretos, adaptado do [1]

Simulador	**Companhia**
Arena	Rockwell Software (http://www.rockwellautomation.com/rockwellsoftware/overview.page)
AutoMod	Brooks Automation (AutoSimulations) (http://www.brooks.com/)
DE3	BYG Systems (http://www.bygsystems.com/index.aspx)
Dosimis3	Simulations Dientleistungs Zentrum GmbH (http://www.sdz.de/)
Enterprise Dynamics(Taylor ED)	Incontrol Enterprise Dynamics (http://www.incontrolsim.com/)
FlexSim(TaylorII)	Flexsim Software Products, Inc. (http://www.flexsim.com/)
GPSS/H	Wolverine Software (http://www.wolverinesoftware.com/)
G2 Rethink	Gensym (http://www.gensym.com/)
Micro Saint	Micro Analysis and Design (http://www.maad.com/)
MMS	nHance Technologies (http://www.enhance-tech.com/)
Quest	Delmia Corp.(Deneb Robotics) (http://www.3ds.com/pt/products/delmia/)
Schedula	Codework (http://www.codework-systems.com/)
ShowFlow	Incontrol Simulation Software B.V. (http://www.incontrolsim.com/)
SimBax	AICOS Technologies AG (http://www.aicos.com/front_content.php)
SimFlex	Flextronics (http://www.flextronics.com/)
Simprocess	CACI Products Company (http://www.caciasl.com/)
SIMUL8	SIMUL8 Corporation Products (http://www.simul8.com/)
SLX	Wolverine Software (http://www.wolverinesoftware.com/)
Spar	Clockwork Solutions (http://www.clockwork-solutions.com/)
Witness	Lanner Group (http://www.lanner.com/)

Tabela 2.2: Ferramentas de Simulação Geométrica, adaptado do [1]

Simulador	Companhia
AMESim	Imagine (http://www.lmsintl.com/LMS-Imagine-Lab-AMESim)
CimStation Robotics(CSR)	Applied Computing & Engineering LTD (http://www.acel.co.uk/)
CMMSimulator	Applied Computing & Engineering LTD (http://www.acel.co.uk/)
Delmia	Delmia Corp. (http://www.3ds.com/pt/products/delmia/)
FoCs	alphaWorks-IBM (https://www.ibm.com/developerworks/community/groups/service/html/communityview?communityUuid=18d10b14-e2c8-4780-bace-9af1fc463cc0)
GRASP	BYG Systems (http://www.bygsystems.com/index.aspx)
HCADWin	NeM (http://www.hcadwin.com/)
IGrip	Delmia(Deneb Robotics) (http://www.3ds.com/pt/products/delmia/)
ProDyn	Ingenious Inc. (http://www.ingeniousinc.com/)
Softmachines	Applied Computing & Engineering LTD (http://www.acel.co.uk/)
Universal Mechanism 2.0	Universal Mechanism Software Lab (http://www.universalmechanism.com/en/pages/index.php?id=6)

Algumas destas ferramentas de simulação possuem uma linguagem de apoio à simulação que permite ao utilizador desenvolver modelos de simulação mais complexos e não específicos.

2.2.1 Arena e ExtendSim

De seguida, são apresentadas as interfaces de duas ferramentas de simulação **DES** frequentemente usadas na Indústria, em particular o **ExtendSim** e **Arena**. Ambas são semelhantes em objetivos e modo de utilização. Do ponto de vista da Indústria permitem por exemplo simular filas, *Conveyor's*, atividades com ou sem horário de funcionamento, decisão, etc..

A ferramenta **ExtendSim** [14] e a ferramenta **Arena** facilitam todas as etapas de projeção de um modelo de simulação de um sistema, desde da sua criação, validação e verificação. Permitem a modelação de eventos discretos ou contínuos. Esta modelação é feita através da construção de modelos, por meio de ações do tipo **Drag & Drop** sobre blocos já modelados. É possível adicionar novos blocos à base de dados, e modificar a informação desses mesmos blocos. O meio de conexão entre os blocos é efetuada através de linhas que representam o fluxo lógico do modelo. As interfaces destas ferramentas

são representadas através das Figuras 2.2 e 2.3.

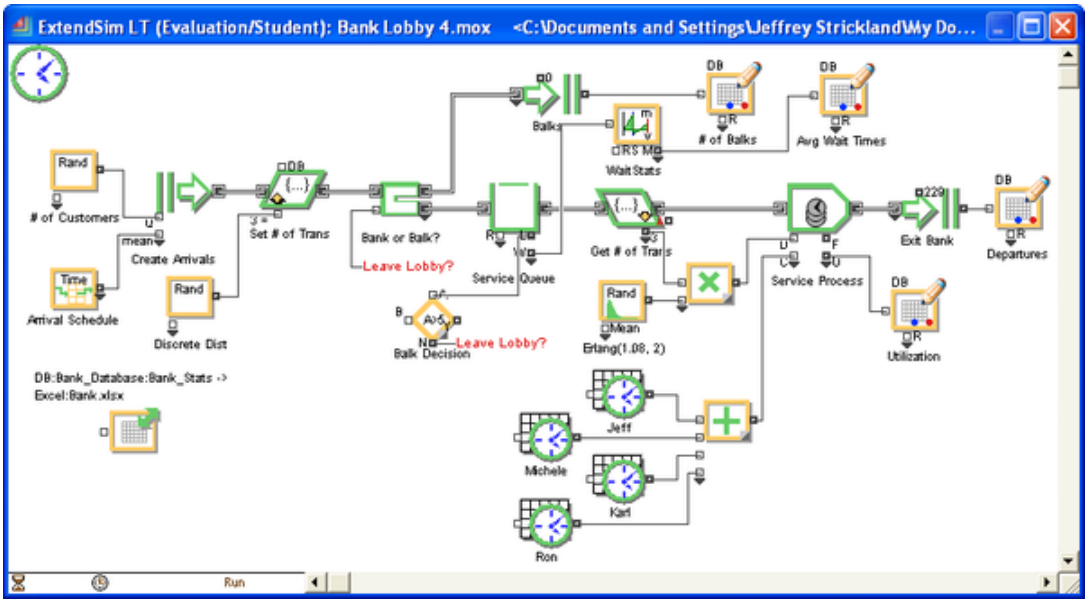

Figura 2.2: ExtendSim GUI

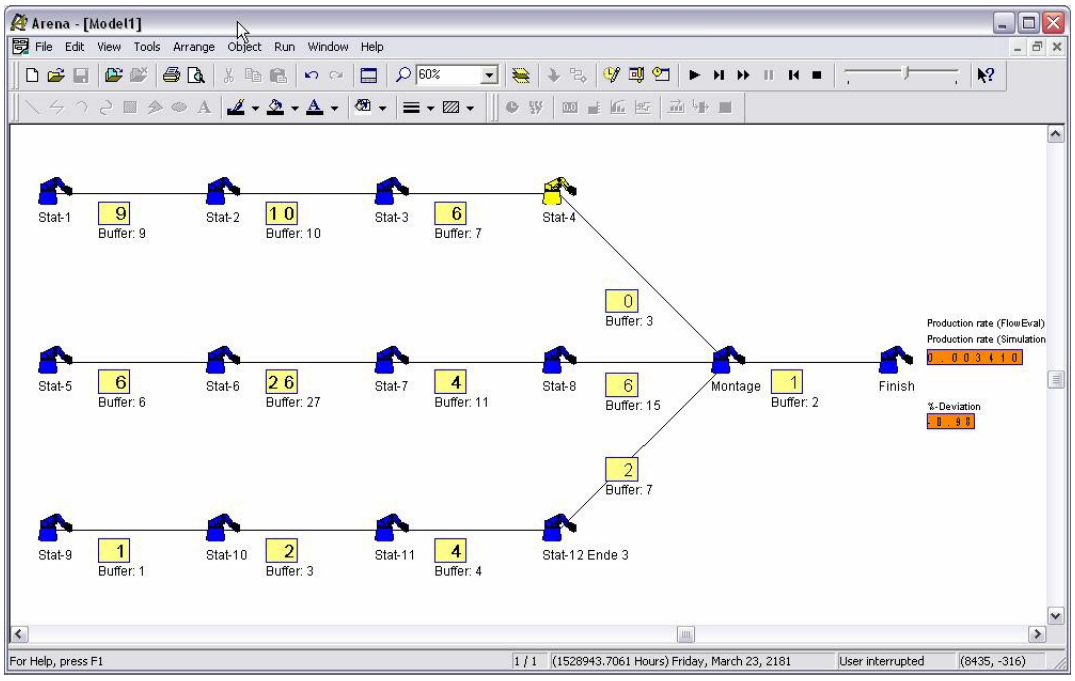

Figura 2.3: Arena GUI

Contundo ambas as ferramentas estão desenhadas para a simulação de sistemas em que não se estende a possibilidade de alterar componentes durante a execução.

Atualmente, diversos paradigmas de produção têm vindo a surgir como fundamento teórico na desenvolvimento de sistemas mecatrónicos baseados em entidades inteligentes, autónomas e distribuídas nomeadamente: Bionic Manufacturing Systems, **BMS** [15], Holonic Manufacturing Systems, **HMS** [16], Reconfigurable Manufacturing Systems, **RMS** [17], Evolvable Assembly Systems, **EAS** e Evolvable Production Systems, **EPS** [18], [19]. Apesar de alguns detalhes específicos, esses paradigmas defendem o uso de sistemas inteligentes distribuídos que interagem e que se auto-organizam para alcançar um objetivo específico de produção.

2.3 MAS, *Sistemas MultiAgente* no âmbito da Indústria

Os **MAS** são importantes na Indústria porque permitem a modelação de sistemas através de unidades distribuídas heterogéneas, chamadas Agentes, em que cada uma delas gere as suas próprias atividades com base no seu estado local e na informação recebida de outras unidades, através de troca de mensagens.

Um agente é um sistema de computação que se encontra situado num determinado ambiente, e que é capaz de efetuar ações autónomas nesse ambiente de forma a atingir determinados objetivos [20].

Existe uma lista de capacidades que um agente inteligente deve apresentar [21]:

1. Reatividade - um agente reage às mudanças de ambiente de maneira a atingir os objetivos.

2. Pró-atividade - um agente que possui um comportamento que atinge os seus objetivos diretamente.

3. Sociável - um agente tem de interagir com outros agentes para que possam atingir os seus objetivos em comum.

Os **MAS** possuem uma arquitetura de controlo descentralizada que oferece vantagens, tais como: a heterogeneidade, a modularidade, a flexibilidade e robustez contra falhas, pelo que, desta forma, a abordagem parece a mais adequada face à gestão dos ambientes de manufatura e da natureza dinâmica e portando construção de sistemas de manufatura mais complexos.

Os **MAS** oferecem então métodos inteligentes para o desenvolvimento e manufatura de produtos, bem como coordenação e cooperação entre os vários domínios em todo o ciclo de vida do produto.

2.3.1 Ferramentas de Simulação baseadas em Agentes

Atualmente, no âmbito da Industria, as ferramentas de simulação baseadas em Agentes têm vindo a ser desenvolvidas e utilizadas pelas características mencionadas anteriormente. As quais se podem destacar **MADSP** [22], **MAST** [23] e **IMSAT** [24]. De seguida são apresentadas e explicadas essas mesmas ferramentas de simulação.

2.3.1.1 MADSP, *Multi-Agent Oriented Distributed Simulation Platform*

Em relação à plataforma **MADSP** [22], é permitido a criação dos modelos dos agentes e suporta a simulação de inúmeros casos num meio distribuído. Esta plataforma possui três camadas que permitem a modelação dos agentes, o centro de *Deployment* que permite instanciar os agentes modelados em qualquer computador de forma eficiente, e o motor de execução que permite executar simulações de modelos.

O módulo que permite a modelação de agentes consiste num ambiente muito favorável para um utilizador familiarizado em conhecimentos de programação, mas separa os utilizadores normais dos programadores, sendo que os últimos podem mudar o modelo dos agentes enquanto os outros usam modelos já definidos e constroem o cenário a simular, graficamente.

O conceito de simulação classificado pelo **MADSP** é expresso pelas seguintes regras, existe um modelo de um sistema constituído por agentes e que contem determinadas características. Dito de outra forma, a simulação consiste numa função que possui parâmetros de entrada, tais como número de agentes e as suas características e o sistema que engloba os mesmos, e que devolve os resultados da simulação. Resumindo a simulação, no **MADSP** é uma caixa negra com *inputs* e *outputs*.

No **MADSP** existem dois tipos de agentes os reativos e os informativos. Os reativos são os agentes que interagem face a alterações vindas do exterior, já os informativos são agentes que apresentam os vários estados dos agentes reativos.

Após a modelação de cada um dos agentes necessários para o cenário pretendido, esses modelos são agrupados num documento para mais tarde ser consultado e estudado. A informação necessária para o **MADSP** é armazenada na seguinte maneira:

1. Um ficheiro com uma descrição geral.
2. Os vários modelos dos agentes envolvidos.
3. O resultado da simulação.

Visto a tecnologia **MAS** permitir a integração com outros sistemas Multiagente, esta plataforma possui um módulo que permite essa integração e distribuição dos agentes modelados. Consiste num portal *Web* o qual os utilizadores se registam e efetuam o *upload*

dos modelos dos seus agentes, modelos esses que podem ser feitos *online* ou *off-line* ao sistema.

Como já foi referido anteriormente o **MADSP** separa os profissionais dos utilizadores regulares, o que quer dizer que os últimos podem utilizar os modelos não projetados por eles, visto não terem os conhecimentos necessários para tal tarefa, e apenas têm de criar o cenário em que esses modelos se inserem, indicando algumas características necessárias para tal cenário.

Após a criação do cenário de simulação que se pretende, a plataforma **MADSP** inicializa os agentes necessários para a simulação ocorrer, distribuindo-os da melhor maneira possível para que não haja carga sobre a rede de telecomunicações. Durante o decorrer da simulação é necessário que todos os agentes estejam dentro do mesmo vetor de tempo, ou seja, se passaram dez segundos de simulação todos os agentes devem estar no décimo segundo de simulação. Por esse facto, o **MADSP** possui um gestor de *Clock* que sincroniza a unidade de tempo de todos os agentes envolvidos, sendo que as ações dos agentes são acionadas quando essa sincronização ocorre.

De acordo com esta arquitetura o agente pode apenas saber o estado do mundo e o seu próprio estado, e baseando-se nisso planeia o seu comportamento durante a simulação. Esses estados são afetados pelo gestor de *Clock*, como já foi referido, e nessa afetação o agente sofre algumas destas alterações:

1. Incrementar o número de *Clock* recebidos.
2. Receber de outros agentes e enviar mensagens para outros agentes, consoante as suas configurações e planos.
3. Colocar os agentes sobre o controlo de outros sistemas, ou seja, migrar os agentes que necessitam de uma grande capacidade computacional para outro sistema.

Devido ao facto de que nem todos os agentes são iguais em termos de modelação e de capacidade computacional, o **MADSP** possui uma camada que permite distribuir os agentes pelos diversos sistemas interligados com o **MADSP**. Essa camada verifica a capacidade de processamento disponível que o computador possui e executa uma estimativa de quanto de espaço e de recursos é que esse agente irá necessitar para que possa executar os seus comportamentos sem que haja sobrecarga no computador. Esta camada encontra-se sempre a ser executada visto que a carga estimada e alocada para cada agente nunca ser uniforme ao longo do decorrer da simulação, logo é necessário verificar o estado interno do computador e se necessário migrar os agentes provenientes dessa sobrecarga para outro computador.

Para efetuar a migração dos agentes, o **MADSP** utiliza uma tecnologia chamada "TerraCota" [25] que basicamente migra informação da memória física de uma máquina virtual para outra máquina virtual, tudo através da rede de dados. Esta tecnologia usa uma

arquitetura servidor-cliente em que o servidor é responsável por descodificar a informação necessária para iniciar o cenário a simular e ativar a camada de distribuição dos agentes, com o objetivo de equilibrar a carga computacional dos agentes pelos vários sistemas interligados. Enquanto o cliente apenas tem de serializar a informação necessária para que o servidor possa efetuar uma melhor gestão da carga computacional.

Como já foi referido, o **MADSP** permite executar múltiplas simulações de casos iguais ou diferentes, o que indica que a camada de distribuição dos agentes necessita sempre de estar a verificar a carga computacional de cada sistema que está interligado com a plataforma, de modo a melhorar o desempenho do sistema como um todo. Esta migração de agentes pode ocorrer a qualquer momento da simulação, o que leva a uma paragem da simulação até que essa transferência esteja completa. Logo é necessário que o gestor de *Clock* e a camada de distribuição dos agentes estejam também em sintonia de modo a que não haja conflitos entre a comunicação entre um agente que não esteja a migrar e outro que realmente necessite de migrar, pois enquanto o agente que está a migrar de um sistema para outro não existe realmente no sistema, só quando o sistema acabar de efetuar a migração é que o agente se encontra ativo e a partir daí é que o sistema pode arrancar de novo, pois caso contrário haveriam erros.

2.3.1.2 MAST, *Manufacturing Agent Simulation Tool*

Relativamente à plataforma **MAST** [23], é uma ferramenta que permite construir sistemas que processam produtos, programada em **JAVA** e utilizando a tecnologia **JADE**. O foco desta plataforma consiste apenas no transporte de produtos por um sistema de estações de produção, usando diferentes meios de transporte, como por exemplo *Conveyors* ou **AGV's**, *Automated Guided Vehicles*, sendo que cada componente do sistema é um agente.

Como cada agente não se encontra fisicamente ao sistema, foi necessário implementar um agente de simulação que recebe a ordem de operação que outros agentes ordenam e este simula o comportamento que seria efetuado se esses agentes tivessem ligados ao mundo físico. Este comportamento não apenas se enquadra aos atuadores mas também aos sensores que possam existir no sistema.

O **MAST** possui uma representação gráfica da simulação como também um *design* específico para a construção de um sistema de manufatura. Com o início da simulação todos os agentes responsáveis pelo transporte dos produtos começam a mover os mesmos até às estações pretendidas para que os produtos possam ser processados. Este processamento de transporte e de produção é efetuado através de trocas de mensagens entre os agentes envolventes.

Com o **MAST** é possível provocar falhas em qualquer dos componentes envolvidos na linha e ver o resultado do transporte dos produtos de maneira a evitar o componente

com a falha técnica. A configuração do sistema pode ser alterado durante a simulação, ou seja, um componente pode ser removido, caso já exista no sistema, ou pode ser adicionado um novo componente.

Existe um componente designado de *Workcell* que representa uma estação de um sistema de manufatura, e uma estação pode ter várias tarefas, um recurso que opera sobre o produto modificando-o, um armazém de produtos, etc.. Para que seja possível fazer um rastreio ao ponto de origem do produto, ou seja, de que estação é que entrou no sistema, o produto é identificado por um *id* único, mas não sendo só essa a tarefa do *id* em todo o sistema, também indica qual o tipo de produto que entrou na estação e qual a melhor ação a tomar perante este produto. Tal como num sistema real uma estação deve conter um leitor de *id* dos produtos, e é esse leitor é implementado no agente de estação.

O agente *Conveyor* é um meio de transporte entre outros dois componentes que interliga. Este agente tem algumas características que o especifica, tais como velocidade e o seu custo, em que o último serve para o cálculo de rotas ótimas dentro do sistema. Este agente envia uma mensagem para o componente que o recebe como entrada, ou seja para onde os produtos vão quando chegam ao final da linha.

Para além destes dois agentes existem ainda outros que representam outros componentes essenciais numa linha de montagem tais como intersecções e *Diverters*. A diferença entre estes dois é a seguinte a intersecção é um agente que possui múltiplas entradas e apenas uma saída enquanto o *Diverter* é um agente que possui um papel mais activo no sistema pois possui múltiplas entradas e múltiplas saídas o que permite escolher o melhor caminho para um determinado componente, alternando os produtos entre os seus componentes de saída.

O *Diverter* possui uma tabela de encaminhamento com os possíveis destinos através das suas saídas. Essa tabela de encaminhamento é obtida através da troca de mensagens entre todos os agentes envolvidos no transporte e produção para um melhor conhecimento do estado interno de cada um e para uma melhor escolha de saídas nos *Diverters*. Desta maneira, mesmo que haja uma falha num componente esta tabela, que cada agente possui, é atualizada e remove o componente da lista, o que confirma o que foi dito anteriormente, que caso haja uma falha nenhum produto será encaminhado para nem por esse componente, até que este seja reparado. O mesmo se passa com a adição de um novo componente, ou seja, mal se interligue um novo componente a um outro é adicionado às várias tabelas de encaminhamento de cada componente.

Para além dos *Conveyors* como meio de ligação aos outros componentes existem ainda os **AGV's**, como já foram mencionados anteriormente. Estes permitem que não haja ligação física entre componentes que se situem em localizações muito próximas. Como se pode verificar o comportamento deste agente é bastante complexo face aos anteriores

pois estes têm de se deslocar pelo sistema o mais rápido possível sem que ocorram colisões com outros **AGV's** ou com outros componentes. Para tal é necessário que cada **AGV** saiba a constituição do sistema, como características e localizações e também as localizações de outros **AGV's**.

Para evitar algumas possíveis colisões entre **AGV's**, quando tentam aceder a um componente partilhável é necessário enviar uma mensagem a esse componente para saber se encontra disponível para interagir com este ou não. Caso esteja o componente recusa todas as outras mensagens de outros, pois foi alocado a interagir com aquele componente naquele momento e por isso é considerado um componente partilhável mas com exclusividade apenas a um outro componente. Outra maneira de evitar colisões é os **AGV's** saberem a posição global e as ações de todos os outros **AGV's**.

Com o **MAST** é possível provocar falhas também nos **AGV's** para verificar o comportamento dos outros agentes. Um tipo de falha, a mais comum no mundo industrial, como perda de produtos é possível simular, em que o comportamento dos outros **AGV** é evitar o trajeto feito pelo **AGV** que indicou a perda de produto, para que não haja colisões do tipo **AGV's**-Produtos.

Como já foi referido anteriormente, o *Workcell* pode representar inúmeras tarefas no **MAST** podendo também representar uma estação de montagem de produtos, o que indica que simula o tempo de uma única operação de montagem. Cada *Workcell* desse tipo pode receber inúmeros produtos do mesmo ou de vários *Conveyors* para que se possa efetuar a montagem de um produto complexo. Mas para tal acontecer é necessário especificar as operações e os tipos de produtos que serão necessários para essa tarefa. Quando todos os requerimentos estiverem concluídos, a *Workcell* começará a simular o tempo de operação.

2.3.1.3 IMSAT, *Intelligent Manufacturing Simulation Agent Tool*

A plataforma **IMSAT** [24] permite simular sistemas de manufatura, baseando-se no conhecimento sobre o estado atual do sistema a simular. O seu objetivo é desenvolver uma arquitetura que incorpore funções de controlo de manufatura como parte da simulação. Essas funções recebem o estado atual do sistema, comparam-no com os objetivos finais do sistema e baseando-se nisso tomam as decisões apropriadas para tal. Este tipo de simulação consiste numa junção da Inteligência Artificial com a simulação tradicional, a qual **IMSAT** se enquadra.

Sendo um sistema Multiagente, a **IMSAT** possui agentes em que cada um representa uma parte de um sistema de controlo de manufatura. O conceito de segmentação de um sistema em componentes individuais baseia-se num modelo de **Quadro Preto** que consiste numa maneira de armazenar informação de todo o sistema, a qual é representada por um conjunto de regras de cada simulação [26].

A arquitetura da ferramenta **IMSAT** é constituída por quatro camadas. A camada com os agentes de simulação inteligentes, gestão da simulação, a camada com a especificação do fluxo de produção do produto e outra com a especificação hierárquica da estrutura dos agentes.

A camada de agentes de simulação inteligentes é uma extensão de uma biblioteca chamada **ISTS-IE**, que consiste num conceito de um **Quadro Preto**. Sendo que cada agente possui um **Quadro Preto** e um conjunto de regras, sendo esta é a característica particular da **IMSAT**.

Os agentes inteligentes representam uma estrutura que controla qualquer sistema de manufatura, esta representação modela a informação necessária, o conhecimento e as interações necessárias para controlar qualquer sistema. Esta camada possui uma estrutura organizacional, ou seja, existem uns agentes que controlam as camadas de mais baixo nível (*Hardware*) e existem outros que comandam esses mesmos agentes.

No modelo geral da **IMSAT** existem os seguintes componentes organizacionais, **Gestão de Divisão**, **Gestão de Produção**, **Gestão de Fabricação**, **Gestão de Montagem**, **Gestão de Ajudantes**, **Gestão de Manutenção**.

Cada componente contém uma ou mais funções de decisão que realiza a tarefa de comunicação entre agentes, havendo dois tipos: uma comunicação do tipo topo para o fundo são da forma de objetivos que altera os parâmetros de operação ou as regras do sistema e uma comunicação do tipo fundo para o topo são da forma de relatórios de desempenho.

A **Gestão de Divisão** é um agente responsável pela supervisão do desempenho de produção baseando-se nos relatórios do agente da **Gestão de Produção**, que são gerados na execução de uma montagem final de um produto. De acordo com estes relatórios a **Gestão de Divisão** consegue prever situações com melhor desempenho alertando a **Gestão de Produção** para esses casos.

A **Gestão de Produção** é um agente responsável pelo controlo de produção tanto no fabrico como na montagem, comunicando com **Gestão de Divisão**, **Gestão de Produção**, **Gestão de Fabricação** e **Gestão de Montagem**. Este agente recebe um relatório, já mencionado anteriormente, o qual permite enumerar os subprodutos necessários para uma montagem final, informando no final os departamentos de produção apropriados, sendo que também recebe relatórios desses mesmos departamentos durante a simulação.

A **Gestão de Fabricação** é um agente responsável pela produção, planeamento e agendamento de estações de serviços, sendo que a **Gestão de Ajudantes** interage com este agente. Os planos de processos, que são disponibilizados a este agente, são representados por uma série de nós de operação, os quais permitem realizar a decisão entre a escolha de estações alternativas e até mesmo caminhos até essas estações. Este agente

escolha uma estação baseando-se num conjunto pré-especificados, como por exemplo o número de produtos em lista de espera, etc..

A **Gestão de Montagem** é um agente similar ao agente anterior, só que baseia-se nas estações de montagem. Sendo que este agente controla os produtos que são subprodutos de um final, sendo depois enviadas para as estações de montagem.

A **Gestão de Ajudantes** é um agente designado por Ajudante, o qual é responsável por iniciar ou parar as operações das estações, ou seja, monitorizar as estações.

A **Gestão de Manutenção** é um agente que controla todos os agentes anteriores, realizando várias interações com os outros gestores do sistema. Este agente recebe um pedido dos ajudantes e baseando-se na disponibilidade de ajudantes que reparam estações, aloca-os a esse pedido.

Atualmente, a **IMSAT** encontra-se a generalizar o conceito de agentes inteligentes, estendendo a capacidade de adicionar outras representações de modelos de agentes.

2.4 Arquitetura do *Instantly Deployable Evolvable Assembly Systems*, IDEAS

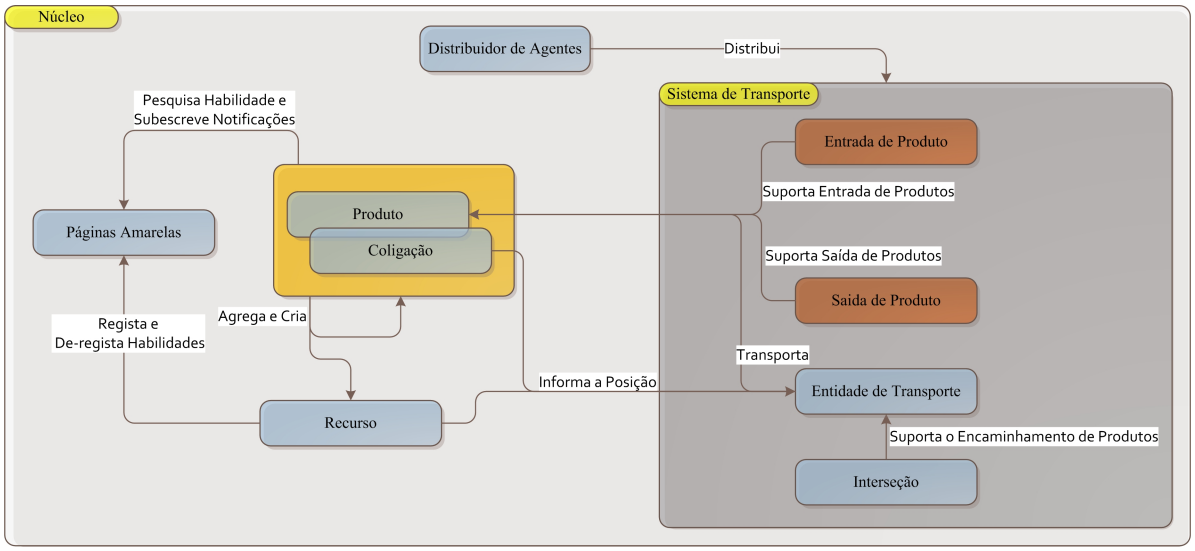

Figura 2.4: Arquitetura do **IADE**

Este projecto **IDEAS** foi implementado na forma de IDEAS Agent Development Environment, **IADE**, o qual já foi validado e integrado num ambiente industrial [27].A Figura 2.4 ilustra a arquitetura do **IADE** [28].

O **IADE** é composto por nove agentes mecatrónicos, **MA**. Esses agentes são: Agente Distribuidor, **DA**, Agente Páginas Amarelas, **YPA**, Agente Recurso, **RA**, Agente Coligação, **CLA**, Agente Produto, **PA**, Agente Ponto de Entrada de Produto, **PSoA**, Agente

Ponto de Saída de Produto, **PSiA**, Agente Transporte, **TEA** e Agente Intersecção, **HUA**. Esses Agentes encontram-se divididos de acordo com o seu papel no sistema e serão explicados mais à frente. Os Agentes conseguem reconhecer e comparar os serviços (Habilidades de acordo com a terminologia do **IADE**) oferecidos por todos os agentes do sistema. A descrição do modelo de informação pode ser encontrada em [29].

Estes agentes são extensões de um conceito de Agente Mecatrónico que agrupa as principais características de todos os tipos de agentes do sistema, de maneira a assegurar de uma forma genérica a interoperabilidade dos mesmos.

Habilidades são oferecidas pelos agentes num âmbito de execução de processos. Os agentes Produto e Coligação oferecem Habilidades compostas. Neste contexto uma Habilidade composta define um fluxo de execução que pode conter decisões e fluxos paralelos ou em série.

O Agente Distribuidor inicializa a plataforma de agentes e os controladores do sistema e permite ao utilizador distribuir qualquer agente para um controlador específico

O Agente Páginas Amarelas possui um registo de todas as habilidades que cada agente oferece no sistema. Os outros agentes podem consultar a localização de outros como também as suas habilidades e subscrever um serviço de notificações que informa quando um agente sai do sistema.

O Agente Recurso interage diretamente com os controladores do sistema e representa o mais baixo nível de abstração do **IADE**. O seu objetivo é traduzir Habilidades para código nativo e assegurar a execução e sincronização com a plataforma de agentes. Ainda informa o agente Transporte acerca da sua localização física.

O Agente Coligação não interage com o sistema, mas coordena a execução de processos dos agentes Recurso e outros agentes Coligação. Implementa os subprocessos que são utilizados pelos agentes Produto durante a execução do seu fluxo de processos. Informa ainda o agente Transporte acerca da sua localização no sistema.

O Agente Produto representa o mais alto nível de abstração no sistema. Cada agente Produto tem a capacidade de gerir o seu próprio fluxo de processos e tomar decisões sobre que Recurso deve executar esse fluxo.

O Agente Ponto de Entrada de Produto efetua a gestão dos Produtos que entram no sistema, associando também um Produto a cada Agente Produto.

O Agente Ponto de Saída de Produto possui um trabalho oposto ao agente anterior. Neste contexto retira Produtos do sistema sempre que um Agente Produto termina o seu fluxo de processos.

O Agente Transporte consiste na abstracção de um *Conveyor* do sistema. É responsável pelo transporte dos Produtos entre Intersecções. Cada *Conveyor* possui múltiplas *Docking Station's* onde podem ser conectados vários Recursos.

O Agente Intersecção controla as Intersecções que existem no sistema e calcula os custos de transporte para atingir qualquer destino no sistema. Esta informação é partilhada com o TEA.

Este trabalho surge no contexto da presente arquitectura como um potencial mecanismo para a sua validação e como uma ferramenta para a simulação de sistemas semelhantes desenvolvidos com base em agentes.

Este trabalho propõe-se simular, com base numa tecnologia Multiagente, qualquer sistema de manufactura que seja controlado por qualquer sistema de transporte externo, apresentando este ponto uma verdadeira inovação, uma vez que, na sua generalidade as ferramentas ou plataformas de simulação são normalmente desenvolvidas caso a caso, levando em consideração as particularidades de cada sistema a simular. Sendo que o sistema de transporte externo escolhido integra o projeto **IDEAS**.

3

Arquitetura do Trabalho

3.1 Arquitetura

A Arquitetura Global do trabalho desenvolvido é representado através da Figura 3.1. A arquitetura possui diferentes tipos de elementos: os **Elementos Estáticos** (Verde), os **Agentes** (Azul), o **Adaptador de Controlo Externo** (Castanho Claro) e ainda a **Aplicação de Modelação Gráfica** (Castanho Escuro).

Os **Elementos Estáticos** consistem em modelos de alguns componentes principais de um sistema de manufatura: o Produto, a *Docking Station*, o Recurso, a Habilidade e Descrição do Sistema.

Os **Agentes** consistem num grupo de entidades autónomas que efetuam a gestão de um sistema de manufatura: o *Conveyor*, a Intersecção, o Ponto de Entrada, o Ponto de Saída e os Distribuidores de Agentes, sendo que algumas destas entidades interagem com uma base de dados, de forma a guardar um histórico de várias simulações já efetuadas.

O **Adaptador de Controlo Externo** consiste numa biblioteca de funções que permite a qualquer sistema de transporte externo construir e controlar um sistema de manufatura que se pretenda simular.

A **Aplicação de Modelação Gráfica** representa a ferramenta desenvolvida neste trabalho que permite construir o modelo lógico do sistema de manufatura e ainda obter resultados estatísticos da sua simulação.

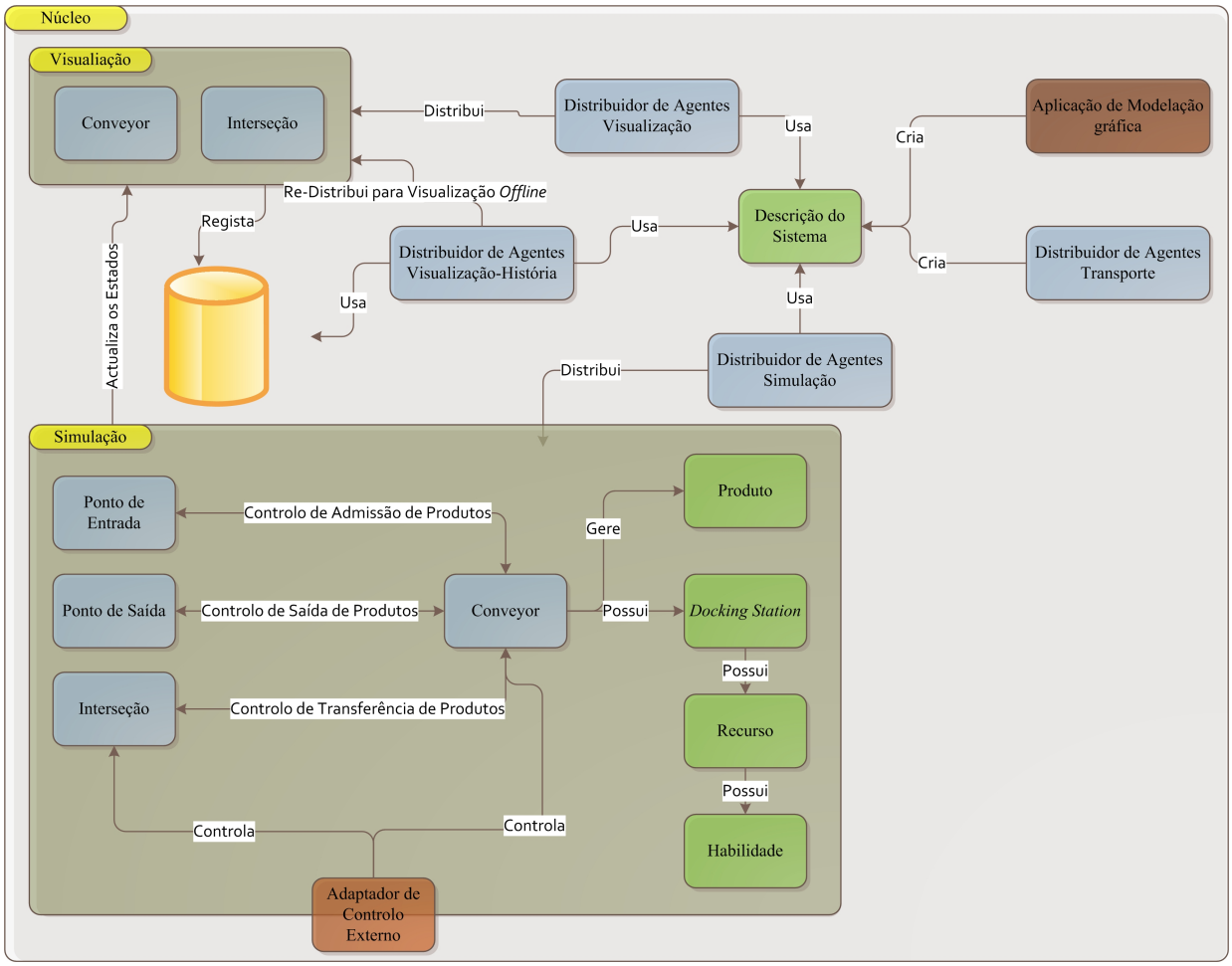

Figura 3.1: Arquitetura Global

3.2 Elementos Estáticos

Existem modelos estáticos que permitem representar alguns dos elementos pertencentes de um sistema de manufatura, como já foi referido anteriormente, sendo de seguida apresentados.

3.2.1 Habilidade

A Habilidade consiste na modelação de ações simples ou complexas que podem ser executadas pelos Recursos existentes no sistema de manufatura. Sendo que possui ainda a seguinte informação necessária para tal modelação, **Nome**, **Tempo** e **Localização**. Em que o **Nome** é utilizado para identificar a Habilidade, o **Tempo** consiste no tempo necessário para completar a mesma e a **Localização** permite indicar a posição que deve realizar a Habilidade. Esta informação é apresentada na Tabela 3.1 e detalhada no Anexo 1.

Tabela 3.1: Variáveis Internas da Habilidade

Nome	Descrição
myName	Contém o Nome da Habilidade
myTime	Contém o tempo necessário para realizar a Habilidade
myIdStation	Contém o *id* da posição associada

3.2.2 Recurso

O Recurso é um elemento que possui determinadas Habilidades, tendo de ser associado a uma localização num *Conveyor*. O Recurso possui a seguinte informação, **Nome**, **Id**, **Lista de Habilidades** e **Localização**. Em que o **Nome** é utilizado para identificar o Recurso, o **Id** consiste na numeração do Recurso no sistema, sendo que este campo é preenchido consoante o número de Recursos existentes, a **Lista de Habilidades** consiste num conjunto de Habilidades que o Recurso pode realizar, a **Localização** permite indicar a interligação entre o Recurso e a sua posição no *Conveyor*. Esta informação é apresentada na Tabela 3.2 e detalhada no Anexo 2.

Tabela 3.2: Variáveis Internas do Recurso

Nome	Descrição
myName	Contém o Nome do Recurso
myId	Contém o *id* do Recurso
mySkillList	Contém a lista de Habilidades
myIdStation	Contém o *id* da *Docking Station* associada

3.2.3 *Docking Station*

A *Docking Station* define posições no *Conveyor* onde um Recurso pode ser adicionado. Possui dois travões para os Produtos localizados na entrada, para evitar acumulação de Produtos dentro da *Docking Station*, e na saída da mesma, para que os Recursos possam operar sobre o Produto, sem que este se mova durante a realização da Habilidade. A *Docking Station* possui a seguinte informação: **Id**, **Lista de Recursos**, **Número de *Clock's***, **Localização**, **Estado do Travão de Entrada**, **Estado do Travão de Saída**, **Recurso Selecionado** e **Habilidade Selecionada**. Em que o **Id** consiste na numeração da *Docking Station* no sistema, sendo que este campo é preenchido consoante o número de *Docking Station's* existentes, a **Lista de Recursos** consiste num conjunto de Recursos que se encontram ligados à *Docking Station*, o **Número de *Clock's*** consiste na contagem de *Clock's* (Número de mensagens de sincronização) que foram enviados durante a simulação, a **Localização** permite indicar a sua posição no *Conveyor*. O **Estado do Travão de Entrada** e **Estado do Travão de Saída** possuem um objetivo comum, a indicação do estado de travões que a *Docking Station* possui, sendo que alteram de valor entre ativado e desativado. O **Recurso Selecionado** indica qual o Recurso que permite operar sobre o Produto que se

encontra dentro da *Docking Station* e a **Habilidade Selecionada** indica qual a Habilidade a executar pelo Recurso selecionado anteriormente. Esta informação é apresentada na Tabela 3.3 e detalhada no Anexo 3.

Tabela 3.3: Variáveis Internas da *Docking Station*

Nome	Descrição
myId	Contém o *id* da *Docking Station*
myResourceList	Contém a lista de Recursos
myCyclicUnit	Contém o número de *Clock* dados durante a simulação
myLocation	Contém a localização no *Conveyor*
myEntryStopper	Variável que indica se o travão de entrada da *Docking Station* está ativado ou não
myExitStopper	Variável que indica se o travão de saída da *Docking Station* está ativado ou não
myResourceIndex	Contém o índice do Recurso da lista anterior
mySkillIndex	Contém o índice da Habilidade da lista que o Recurso possui

3.2.4 Produto

O Produto também consiste num elemento complexo de um sistema de manufatura, pois possui inúmeras características que permitem identificá-lo, classificá-lo e transportá-lo até às *Docking Station's* destinadas a processá-lo. Por isso possui a seguinte informação: **Id**, **Localização**, **Tamanho**, **Lista de Habilidades**, **Id Global**, **Id de *Docking Station***, **Caso da Localização** e **Id de Ponto de Entrada**. Em que o **Id** consiste na identificação do tipo de Produto, a **Localização** permite indicar a sua posição no *Conveyor*, o **Tamanho** consiste na identificação da dimensão do produto no sistema. A **Lista de Habilidades** consiste num conjunto de Habilidades necessárias a serem realizadas sobre o Produto, o **Id Global** indica o número de Produtos enviados pelo Ponto de Entrada que introduziu este Produto, sendo que este campo representa uma contagem de Produtos. O **Id de *Docking Station*** representa o *id* da próxima *Docking Station* para a qual o Produto deve ser encaminhado, o **Caso da Localização** consiste na indicação da localização do Produto em relação à *Docking Station* anteriormente indicada e o **Id de Ponto de Entrada** serve para indicar o Ponto de Entrada que introduziu o Produto no sistema. Esta informação é apresentada na Tabela 3.4 e detalhada no Anexo 4.

Tabela 3.4: Variáveis Internas do Produto

Nome	Descrição
myId	Contém o *id* do Produto
myLocation	Contém a localização no *Conveyor*
myLength	Contém o tamanho do Produto
mySkillList	Contém a lista de Habilidade
myGlobalId	Contém o número de Produtos enviados pelo Ponto de Entrada durante a simulação
myIdStation	Contém o *id* da próxima *Docking Station* que o Produto tem de entrar
myStationCase	Variável que indica se o Produto encontra-se à Esquerda, Dentro, ou à Direita da *Docking Station*
myEntryPointId	Contém o *id* do Ponto de Entrada que inicializou o Produto

3.2.5 Descrição do Sistema

A Descrição do Sistema consiste na modelação dos componentes que compõem o sistema: *Conveyor*, Intersecção, Produto, etc.. Também modelando as suas relações com outros objetos. Esta descrição possui a seguinte informação: **Lista de Pontos de Entrada**, **Lista de Pontos de Saída**, **Lista de *Conveyor's***, **Lista de Intersecções** e **Lista de Ligações entre os Elementos**. Estes novos elementos serão explicados com mais detalhe a seguir. Esta informação é apresentada na Tabela 3.5 e detalhada no Anexo 5.

Tabela 3.5: Variáveis Internas do Sistema

Nome	Descrição
myEntryList	Contém a lista de Ponto de Entrada
myConveyorList	Contém a lista de *Conveyor*
myExitList	Contém a lista de Ponto de Saída
myJunctionList	Contém a lista de Intersecção
myComponentConnectionsList	Contém a lista de ligações dos elementos do sistema

3.3 Agentes

Como já foi referido anteriormente, a arquitetura é constituída por agentes, os quais possuem distintas tarefas e informação que vão mudando ao longo do decorrer da simulação. Através da Figura 3.1 é possível identificar os vários tipos de agentes e a forma como interagem no contexto da presente arquitetura.

3.3.1 Ponto de Entrada

O Ponto de Entrada possui um único comportamento cíclico, cujo objetivo é a introdução de um Produto no sistema. Este agente possui a seguinte informação: **Id**, **Velocidade**, **Lista de Produtos**, **Produto Selecionado**, **Número de *Clock's***, **Repetição**, **Valor Temporal do *Clock***, **Término** e **Id Global**. Em que o **Id** é utilizado para identificar o Ponto de Entrada, a **Velocidade** consiste no tempo que é necessário esperar até que o Ponto de Entrada possa introduzir um novo Produto no sistema. A **Lista de Produtos** consiste num conjunto de Produtos a serem introduzidos no sistema, o **Produto Selecionado** indica qual o Produto que deve ser introduzido da lista anterior. O **Número de *Clock's*** consiste na contagem de *Clock's* (Número de mensagens de sincronização) que foram enviados durante a simulação, a **Repetição** indica se no fim de percorrer a **Lista de Produtos** é necessário percorrer novamente a mesma lista para que se possa introduzir os mesmos Produtos no sistema, o **Valor Temporal do *Clock*** indica qual o valor de cada *Clock* dado durante a simulação, ou seja, se um *Clock* corresponde a um segundo na realidade, se um minuto, etc.. O **Término** indica se o Distribuidor de Agentes de Simulação terminou a simulação e se por sua vez o Ponto de Entrada deve também cessar os seus serviços e o **Id Global** consiste no número total de Produtos enviados pelo Ponto de Entrada. Esta informação é apresentada na Tabela 3.6 e detalhada no Anexo 6.

Tabela 3.6: Variáveis Internas do Ponto de Entrada

Nome	Descrição
myId	Contém o *id* do Ponto de Entrada
myBoxList	Contém a lista de Produtos a serem lançados no sistema
mySpeed	Contém o tempo de espera até um próximo lançamento de um novo Produto da lista anterior
myBoxIndex	Contém o índice do Produto da lista anterior
myCyclicUnit	Contém o número de *Clock* dados durante a simulação
myCyclic	Variável que indica se a lista de elementos é para repetir ao longo da simulação ou se apenas uma vez
myKill	Variável que indica se o agente é marcado para término
myCyclicValue	Contém o valor temporal de cada *Clock* dado pela simulação
myGlobalIdBox	Contém o número de Produtos enviados durante a simulação

Como já foi referido, o Ponto de Entrada representa um agente que introduz um Produto no sistema a um ritmo definido pelo utilizador. Este agente possui os seguintes comportamentos que são essenciais para a sua simulação:

1. Matar o agente.
2. Enviar novamente o Produto selecionado.
3. Seleccionar o próximo Produto a ser introduzido.
4. Actualizar o seu estado.

Estes comportamentos serão representados e explicados a seguir.

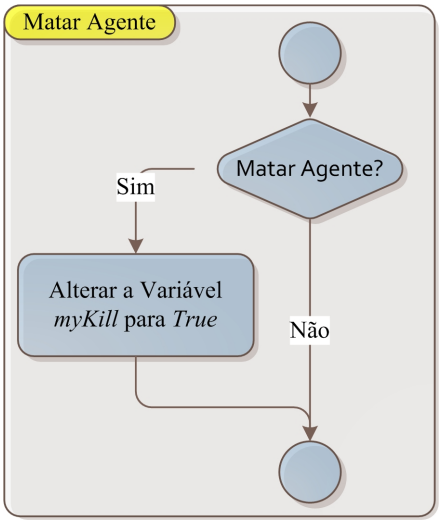

Figura 3.2: Matar o Agente

O comportamento representado pela Figura 3.2 é inicializado através da receção de uma mensagem de término por parte do distribuidor de agentes de simulação. O Ponto de Entrada altera a variável *myKill* para *True*, para que recuse os outros tipos de mensagens recebidas, pois indica que o agente encontra-se marcado para término.

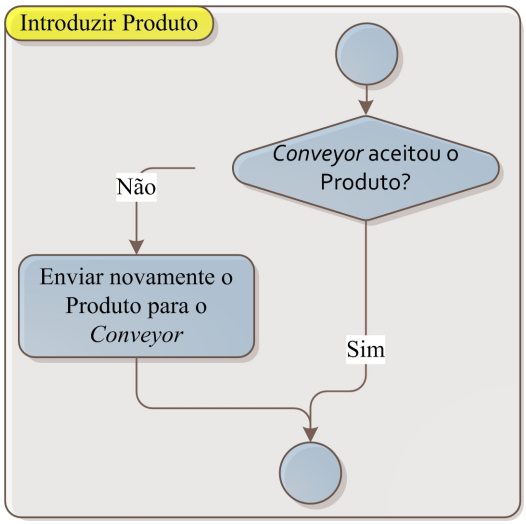

Figura 3.3: Enviar novamente o Produto selecionado

O comportamento representado pela Figura 3.3 é inicializado através da receção de uma mensagem com a aceitação ou a rejeição do Produto por parte do *Conveyor*. Depois da verificação do conteúdo da mensagem recebida, e se consiste numa aceitação ou numa rejeição, o Ponto de Entrada reenvia ou não o Produto para o *Conveyor*, para que se possa introduzir um novo Produto no sistema.

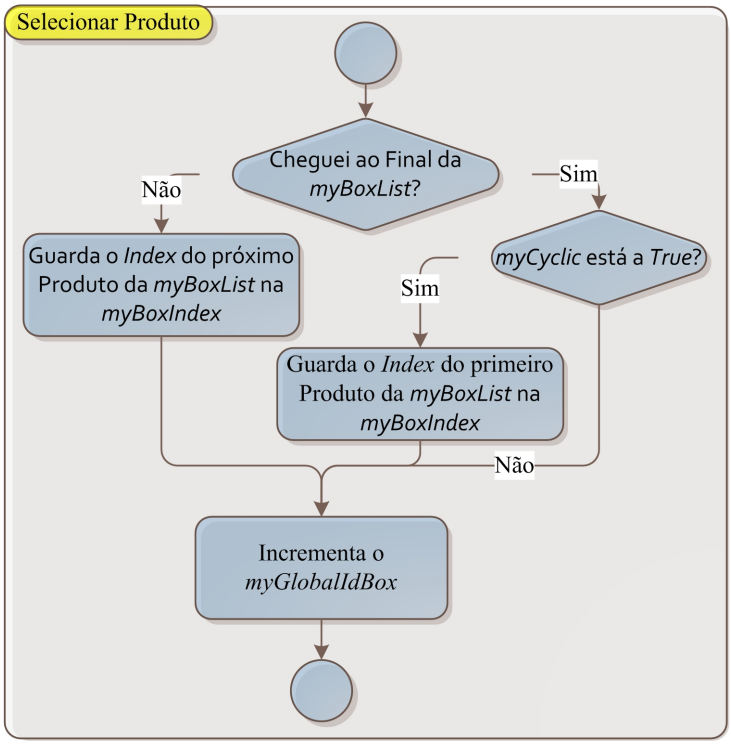

Figura 3.4: Selecionar o próximo Produto a ser introduzido

O comportamento representado pela Figura 3.4 é inicializado através da recepção de uma mensagem com a aceitação do Produto por parte do *Conveyor*. Depois da verificação do conteúdo da mensagem recebida, o Ponto de Entrada confirma se percorreu a Lista de Produtos até ao final, e em caso afirmativo, se a lista é para ser repetida guarda-se na variável *myBoxIndex* o *índice* do primeiro Produto dessa lista, ou em caso negativo guarda-se nessa mesma variável o *índice* do próximo elemento da lista.

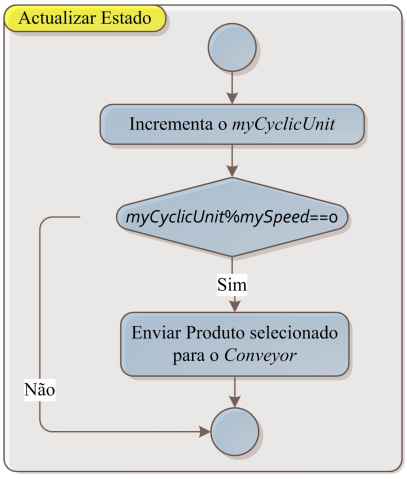

Figura 3.5: Actualizar o Estado

O comportamento representado pela Figura 3.5 é inicializado através da recepção de uma mensagem por parte do distribuidor de agentes de simulação. O Ponto de Entrada incrementa a variável *myCyclicUnit*, pois serve para indicar a contagem de *Clock's* dados pela simulação, como já foi referido anteriormente. De seguida, o Ponto de Entrada verifica se é possível enviar um novo produto para o *Conveyor*, ao qual se encontra associado.

3.3.2 Ponto de Saída

O Ponto de Saída possui um único objectivo, a remoção de um Produto do sistema colocando-o numa lista ordenada por entrada de Produtos que ele possui. Este agente possui a seguinte informação: **Id**, **Lista de Produtos** e **Número de *Clock's***. Em que o **Id** é utilizado para identificar o Ponto de Saída, a **Lista de Produtos** consiste num conjunto de Produtos foram retirados do sistema e o **Número de *Clock's*** consiste na contagem de *Clock's* (Número de mensagens de sincronização) que foram enviados durante a simulação. Esta informação é apresentada na Tabela 3.7 e detalhada no Anexo 7.

Tabela 3.7: Variáveis Internas do Ponto de Saída

Nome	Descrição
myId	Contém o *id* do Ponto de Saída
myBoxList	Contém a lista de Produtos que saíram do sistema
myCyclicUnit	Contém o número de *Clock* dados durante a simulação
myKill	Variável que indica se o agente é marcado para término

O Ponto de Saída representa um agente que retira um Produto do sistema a simular. Este agente possui os seguintes comportamentos que são essenciais para a simulação deste elemento:

1. Matar o agente.
2. Adicionar Produto à lista.
3. Actualizar o seu estado.

Estes comportamentos serão representados e explicados a seguir.

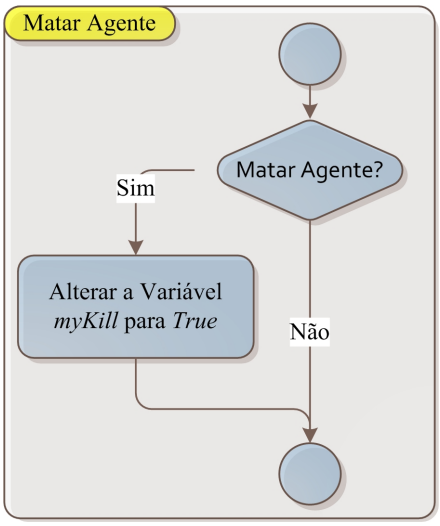

Figura 3.6: Matar o Agente

O comportamento representado pela Figura 3.6 é inicializado através da recepção de uma mensagem de término por parte do distribuidor de agentes de simulação. Depois da confirmação do conteúdo da mensagem recebida, o Ponto de Saída altera a variável *myKill* para *True*, para que recuse os outros tipos de mensagens recebidas, pois indica que o agente encontra-se marcado para término.

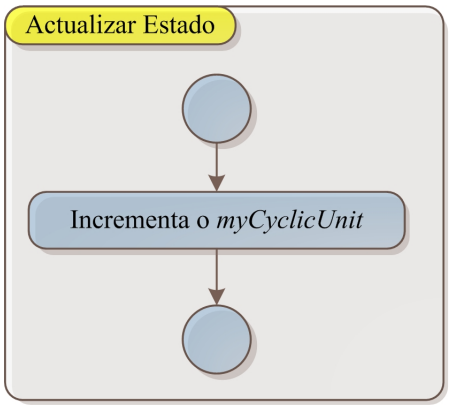

Figura 3.7: Actualizar o seu estado

O comportamento representado pela Figura 3.7 é inicializado através da recepção de uma mensagem por parte do distribuidor de agentes de simulação. Depois da verificação do conteúdo da mensagem recebida, o Ponto de Saída incrementa a variável *myCyclicUnit*, que serve para indicar a contagem de *Clock's* dados pela simulação. Como consiste num elemento que retira Produtos do sistema e só executa essa tarefa quando recebe outro tipo de mensagem, este agente não possui mais tarefas a realizar quando recebe a mensagem de sincronização.

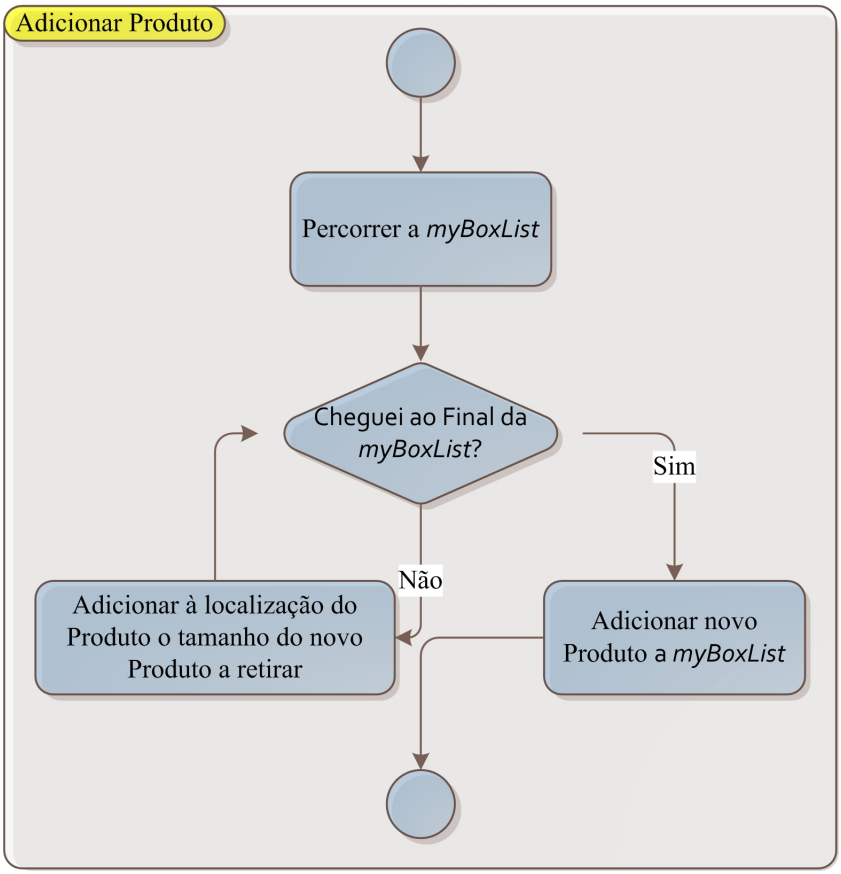

Figura 3.8: Adicionar a palete à lista

O comportamento representado pela Figura 3.8 é inicializado através da recepção de uma mensagem por parte do *Conveyor*. O Ponto de Saída percorre a lista de Produtos com o objectivo de adicionar à Localização de todos os elementos dessa lista a dimensão do novo Produto, para simular a deslocação dos mesmo no sistema. Para além dessa deslocação, ainda adiciona o novo Produto à lista.

3.3.3 *Conveyor*

O *Conveyor* possui os seguintes objectivos neste sistema: transportar Produtos de um elemento para outro e informar as *Docking Station's* que o Produto se encontra pronto para ser operado. Este agente possui a seguinte informação: **Id**, **Tamanho**, **Velocidade**, **Lista de Produtos**, **Número de *Clock's***, **Valor Temporal do *Clock***, **Orientação**, **Direção**, **Término** e **Lista de *Docking Station's***. Em que o **Id** é utilizado para identificar o *Conveyor*, a **Velocidade** relaciona a variação da posição no espaço em relação ao tempo, ou seja, qual a distância percorrida por um Produto num determinado intervalo de tempo no sistema e o **Tamanho** consiste na dimensão que o *Conveyor* possui para unir dois elementos do sistema. A **Lista de Produtos** consiste num conjunto de Produtos a serem deslocados pelo *Conveyor*, o **Número de *Clock's*** consiste na contagem de *Clock's* (Número de

mensagens de sincronização) que foram enviados durante a simulação, o **Valor Temporal do *Clock*** indica qual o valor de cada *Clock* dado durante a simulação, ou seja, se um *Clock* corresponde a um segundo na realidade, se um minuto, etc.. O **Término** indica se o Distribuidor de Agentes de Simulação terminou a simulação e que por sua vez o *Conveyor* deve também cessar os seus serviços, tanto como a **Orientação** como a **Direção** auxiliam na representação gráfica do *Conveyor*, sendo que a primeira indica se se encontra na posição Horizontal ou na Vertical e a segunda indica se os Produtos se deslocam da esquerda para a direita ou de cima para baixo, e vice-versa e a **Lista de *Docking Station's*** consiste num conjunto de *Docking Station* que se encontram interligados com o *Conveyor*. Esta informação é apresentada na Tabela 3.8 e detalhada no Anexo 8.

Tabela 3.8: Variáveis Internas do *Conveyor*

Nome	Descrição
myId	Contém o *id* do *Conveyor*
myLenght	Contém o tamanho do *Conveyor*
mySpeed	Contém a velocidade com que os Produtos se deslocam
myBoxList	Contém a lista de Produtos
myCyclicUnit	Contém o número de *Clock* dados durante a simulação
myHorinzontal	Variável que indica se a orientação do agente é Horizontal ou Vertical
myDirection	Variável que indica a direcção do movimento dos Produtos
myStationsList	Contém a lista de *Docking Station's*
myCyclicValue	Contém o valor temporal de cada *Clock* dado pela simulação
myKill	Variável que indica se o agente é marcado para término

Neste trabalho, o *Conveyor* possui três modos de funcionamento:

1. **Visualização**- Representa o estado interno do componente, como por exemplo, a localização dos Produtos e das *Docking Station's* no mesmo.

2. **Simulação Autónoma**- Realiza um comportamento autónomo sobre as acções das *Docking Station's* e dos travões, ou seja, se um Produto estiver dentro de uma *Docking Station* e se estiver pronto para sair da mesma, o *Conveyor* activa o travão de Saída para que o Produto possa ser encaminhado para outra *Docking Station*, ou para o Ponto de Saída ou até mesmo para uma Intersecção.

3. **Simulação Semiautónoma**- Realiza apenas a deslocação dos Produtos pelo *Conveyor*, sendo que neste modo não possui qualquer tipo de controlo sobre os travões, pois encontra-se interligado com um sistema de transporte externo que efectua a gestão do mesmo.

O *Conveyor* representa um agente que transporta Produtos entre dois elementos, como já foi referido, baseando-se numa configuração prévia pelo utilizador. Este agente possui os seguintes comportamentos que são essenciais para a sua simulação:

1. Matar o agente.
2. Activar o travão de entrada ou de saída da *Docking Station* X ou o travão de saída do *Conveyor*.
3. Adicionar ou remover Produto na lista.
4. Actualizar o seu estado.

Estes comportamentos serão representados e explicados a seguir.

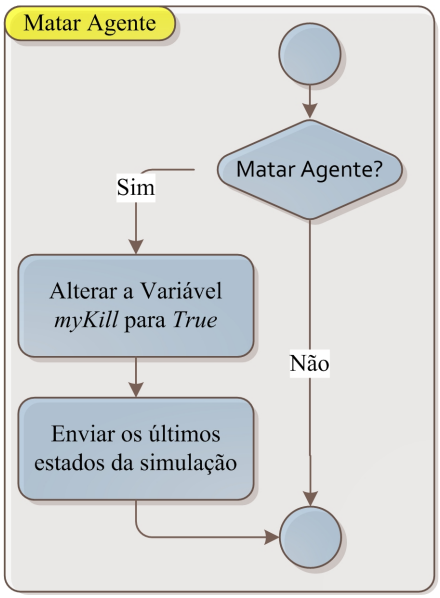

Figura 3.9: Matar o Agente

O comportamento representado pela Figura 3.9 é inicializado através da recepção de uma mensagem de término por parte do distribuidor de agentes de simulação. Depois da confirmação do conteúdo da mensagem recebida, o *Conveyor* altera a variável *myKill* para *True* para que recuse os outros tipos de mensagens recebidas, pois indica que o agente encontra-se marcado para término e ainda envia os seus últimos estados de simulação para o distribuidor de agentes de visualização.

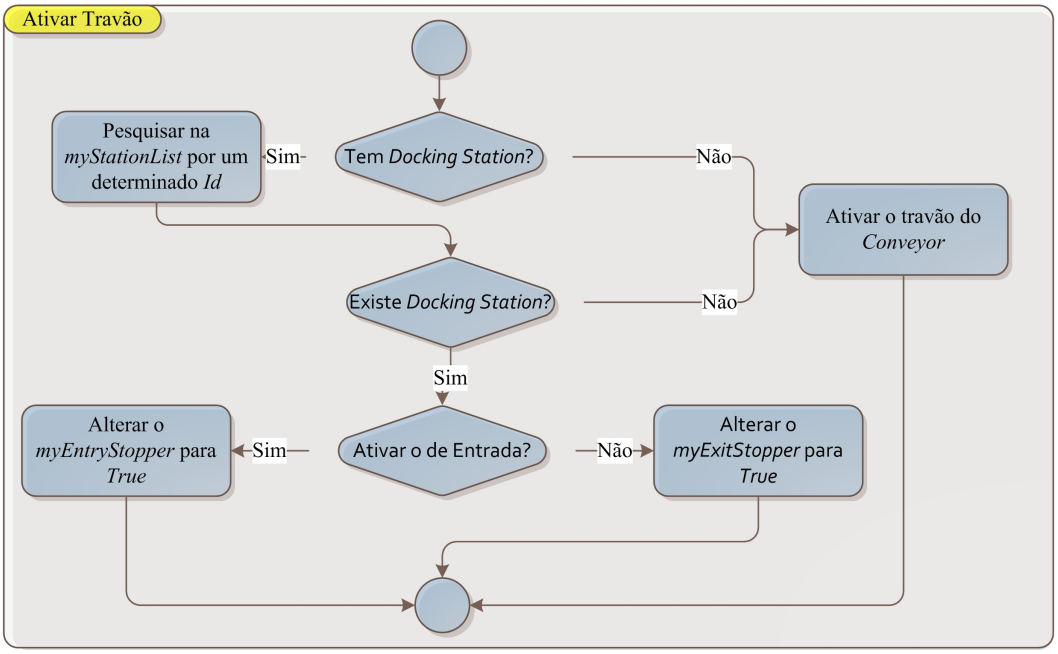

Figura 3.10: Activar Travão

O comportamento representado pela Figura 3.10 é inicializado através da recepção de uma mensagem por parte do controlador externo do sistema ou, como já foi referido, por iniciativa própria do *Conveyor*. Depois da confirmação do conteúdo da mensagem recebida, o *Conveyor* altera as variáveis de controlo dos travões, quer da *Docking Station* quer do próprio *Conveyor* para *True*, para que os Produtos continuem a deslocar-se pelo mesmo.

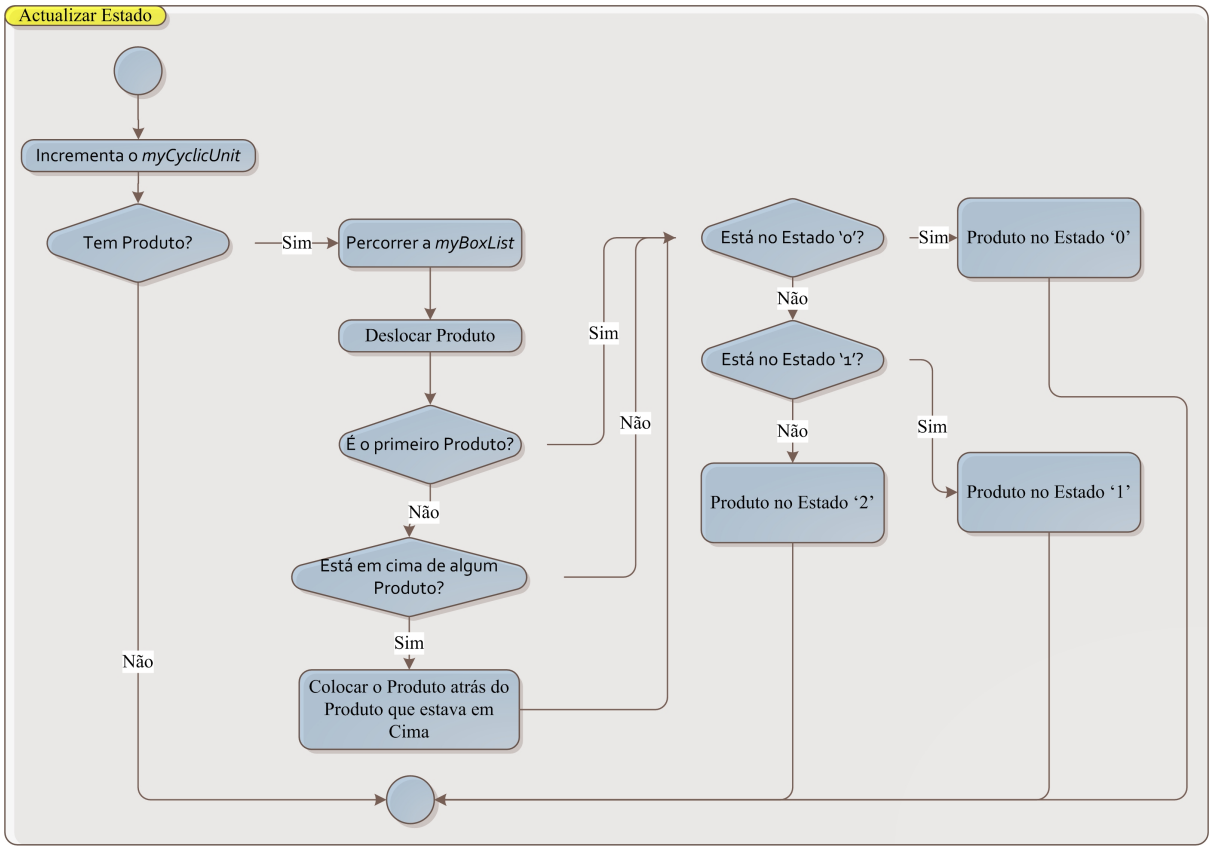

Figura 3.11: Actualizar o seu estado

O comportamento representado pela Figura 3.11 é inicializado através da recepção de uma mensagem por parte do distribuidor de agentes de simulação. O *Conveyor* incrementa a variável *myCyclicUnit*, para indicar a contagem de *Clock's* dados pela simulação, de seguida verifica se possui Produtos na lista, para que possa efectuar a deslocação de cada uma e actualizar a variável *myStationCase* de acordo com determinados casos, que serão explicadas a seguir. Verificam-se ainda as sobreposições de valores de localização para garantir a ordem correta dos Produtos.

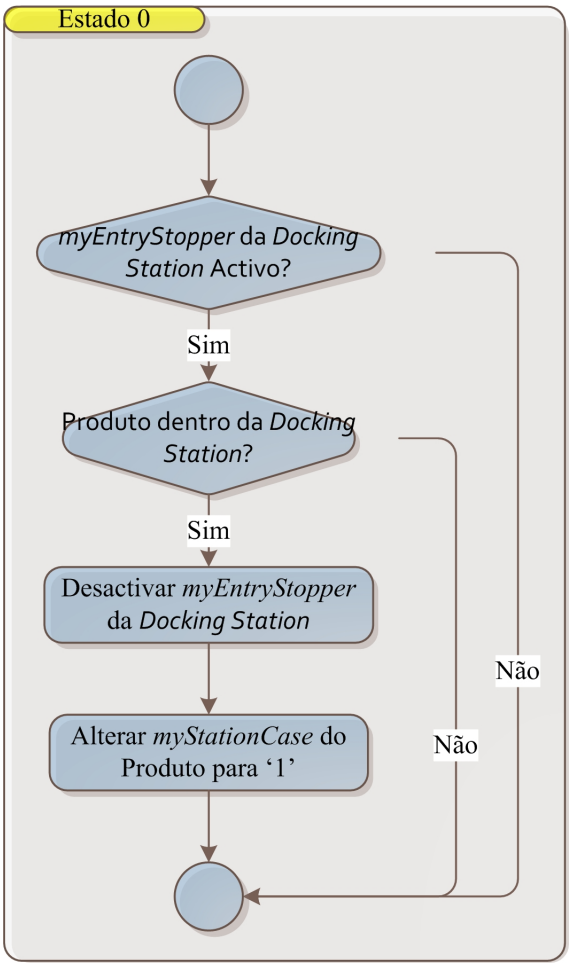

Figura 3.12: Produto no Estado 0

O comportamento representado pela Figura 3.12 é inicializado quando a variável *myStationCase* do Produto se encontra no Estado 0, sendo que nesta situação verifica-se se a variável *myEntryStopper* está a *True* e se o Produto já atravessou o travão de entrada da *Docking Station*. Caso tudo se confirme a variável *myEntryStopper* passa para *False* e a variável *myStationCase* do Produto passa para o Estado 1.

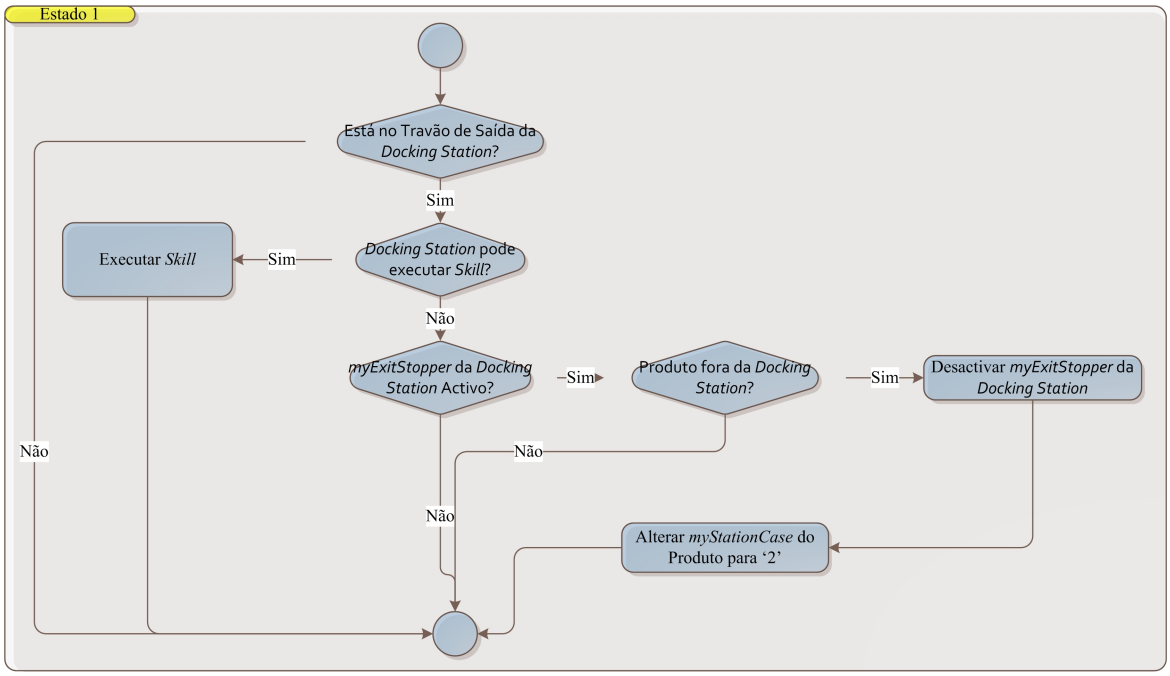

Figura 3.13: Produto no Estado 1

O comportamento representado pela Figura 3.13 é inicializado quando a variável *myStationCase* do Produto se encontra no Estado 1, sendo que nesta situação verifica-se se o Produto ja chegou ao travão de saída da *Docking Station* e se é possível executar a Habilidade. Caso se confirme a possibilidade de executar a Habilidade, o *Conveyor* simula o tempo de execução da mesma, caso contrário verifica-se de seguida se variável *myExitStopper* está a *True* e se o Produto atravessou o travão de saída da *Docking Station*. No caso disso acontecer a variável *myExitStopper* passa para *False* e altera-se a variável *myStationCase* do Produto passa para o Estado 2.

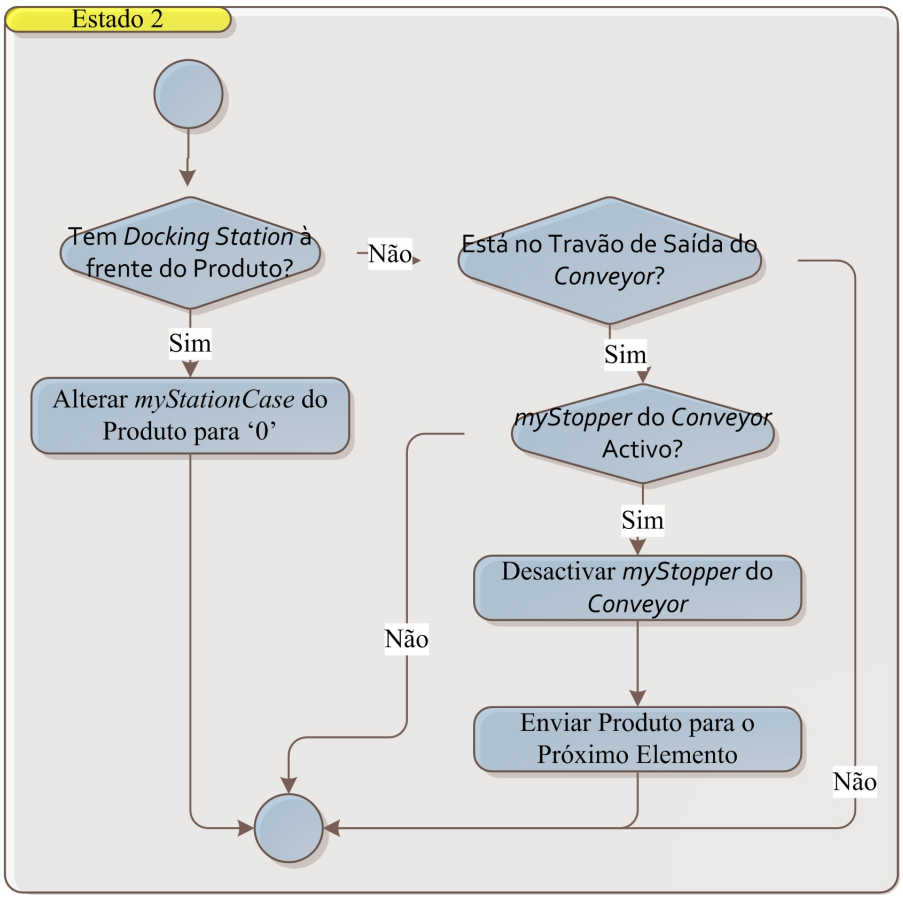

Figura 3.14: Produto no Estado 2

O comportamento representado pela Figura 3.14 é inicializado quando a variável *myStationCase* do Produto se encontra no Estado 2, sendo que nesta situação se existirem mais *Docking Station's* para o Produto percorrer no *Conveyor*, a variável *myStationCase* do Produto passa para o Estado 0. Caso contrário, o Produto encontra-se perto do travão de saída do *Conveyor* e então verifica-se se o Produto está nesse mesmo travão e se a variável *myStopper* está a *True*. Confirmando-se as condições anteriormente mencionadas, a variável *myStopper* passa para *False* e envia-se o Produto para o elemento de destino.

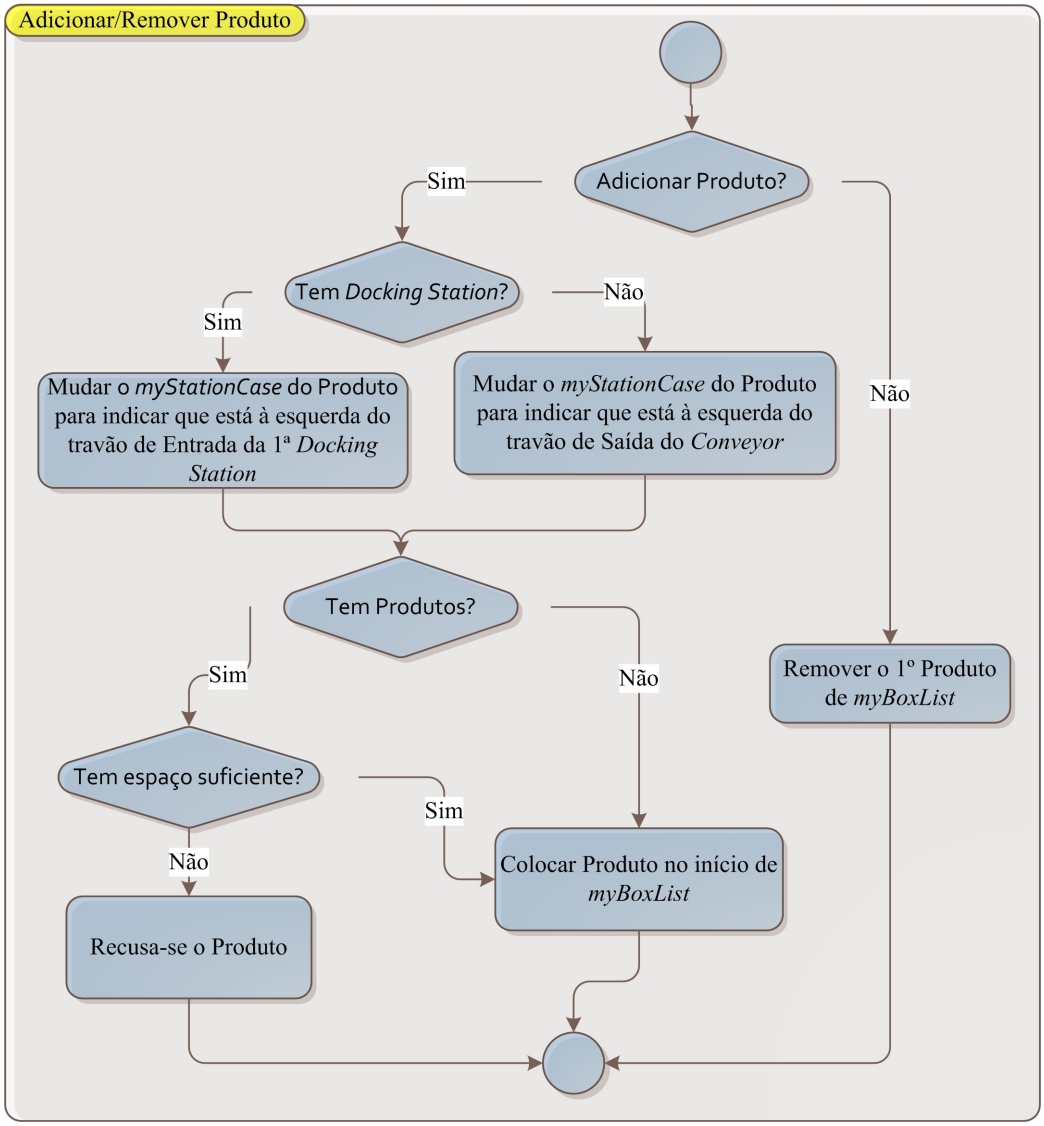

Figura 3.15: Adicionar ou Remover Produto

O comportamento representado pela Figura 3.15 é inicializado através da recepção de uma mensagem por parte do elemento que se encontra interligado à entrada do *Conveyor*. O *Conveyor* retira o primeiro Produto da lista, ou adiciona-o à lista. Verifica-se se possui espaço suficiente para que o Produto possa entrar totalmente e não parcialmente no *Conveyor* e ainda valida-se se existem *Docking Station*de modo a alterar a variável *myStationCase* de acordo com o caso em que se encontre.

3.3.4 Intersecção

A Intersecção tem como objectivo encaminhar os Produtos para os *Conveyor's* de saída com o menor custo de transporte, através de calculo de métricas. Este agente possui a seguinte informação: **Id**, **Tempo**, **Produto**, **Tipo**, **Número de *Clock's***, **Tipo de Encaminhamento**, **Lista de *Rácios* de Produtos**, **Valor Temporal do *Clock*** e **Lista de Tipos de Produtos**. Em que o **Id** é utilizado para identificar a Intersecção, o **Tempo** consiste no tempo necessário para a Intersecção calcular uma nova rota para um determinado Produto e o **Tipo** identifica o tipo de Intersecção baseando-se no número de saídas e entradas da mesma. O **Número de *Clock's*** consiste na contagem de *Clock's* (Número de mensagens de sincronização) que foram enviados durante a simulação, o **Valor Temporal do *Clock*** indica qual o valor de cada *Clock* dado durante a simulação, ou seja, se um *Clock* corresponde a um segundo na realidade, se um minuto, etc. e o **Tipo de Encaminhamento** permite identificar a forma como a Intersecção efectua a distribuição dos Produtos, se por número ou por tipo de produtos, ou seja, definir um número fixo de produtos para cada *Conveyor* ou então definir os tipos de produtos que devem ser encaminhados para cada *Conveyor*. A **Lista de *Rácios* de Produtos** e **Lista de Tipos de Produtos** consistem num conjunto de informação do número e dos tipos de Produtos, respectivamente, que devem ser encaminhados para cada *Conveyor*. Esta informação é apresentada na Tabela 3.9 e detalhada no Anexo 9.

Tabela 3.9: Variáveis Internas da Intersecção

Nome	Descrição
myId	Contém o *id* do agente
myTime	Contém o tempo que o agente necessita de esperar antes de enviar o elemento do tipo Palete para o agente de destino
myBox	Contém o Produto
myCyclicUnit	Contém o número de *Clock* dados durante a simulação
myType	Contém o tipo de Intersecção é, de acordo com as entradas e saídas do mesmo
myRatio	Variável que indica qual o método de distribuição dos Produtos
myRatios	Contém uma lista de distribuição dos Produtos, de acordo com o número de elementos necessários a serem enviados a cada elemento de destino
myFilters	Contém uma lista de distribuição dos Produtos, de acordo com o tipo de elementos necessários a serem enviados a cada elemento de destino
myCyclicValue	Contém o valor temporal de cada *Clock* dado pela simulação
myKill	Variável que indica se o agente é marcado para término

Neste trabalho, a Intersecção também possui três modos de funcionamento:

1. **Visualização**- Representa o estado interno do componente, como por exemplo, se a Intersecção está ocupada ou não.
2. **Simulação Autónoma**- Realiza um comportamento autónomo sobre o encaminhamento dos Produtos, ou seja, baseando-se na configuração definida pelo utilizador a Intersecção redirecciona os Produtos.
3. **Simulação Semiautónoma**- Realiza apenas o encaminhamento dos Produtos através dos comandos dados pelo sistema de transporte externo, pois este é que efectua a gestão do sistema.

A Intersecção representa um agente que redirecciona os Produtos pelo sistema baseando-se numa configuração prévia pelo utilizador, ou pelo sistema de transporte externo capaz de se interligar com o trabalho desenvolvido. Este agente possui os seguintes comportamentos que são essenciais para a sua simulação:

1. Matar o agente.
2. Saber se a Intersecção está ocupada.
3. Adicionar ou remover Produto.
4. Actualizar o seu estado.

Estes comportamentos serão representados e explicados a seguir.

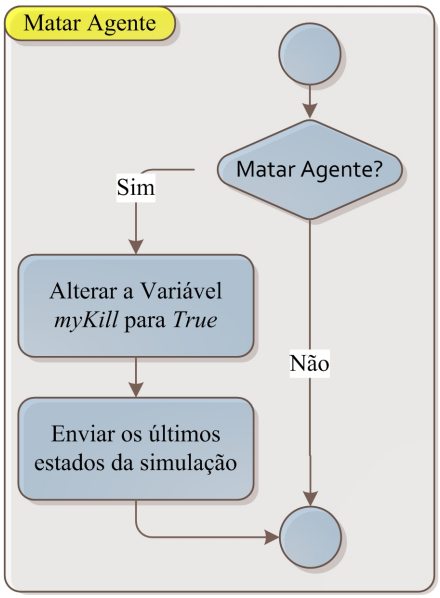

Figura 3.16: Matar o agente

O comportamento representado pela Figura 3.16 é inicializado através da recepção de uma mensagem de término por parte do distribuidor de agentes de simulação. Depois da confirmação do conteúdo da mensagem recebida, a Intersecção altera a variável *myKill* para *True*, para que recuse os outros tipos de mensagens recebidas, pois indica que o agente encontra-se marcado para término e ainda envia os seus últimos estados de simulação para o distribuidor de agentes de visualização.

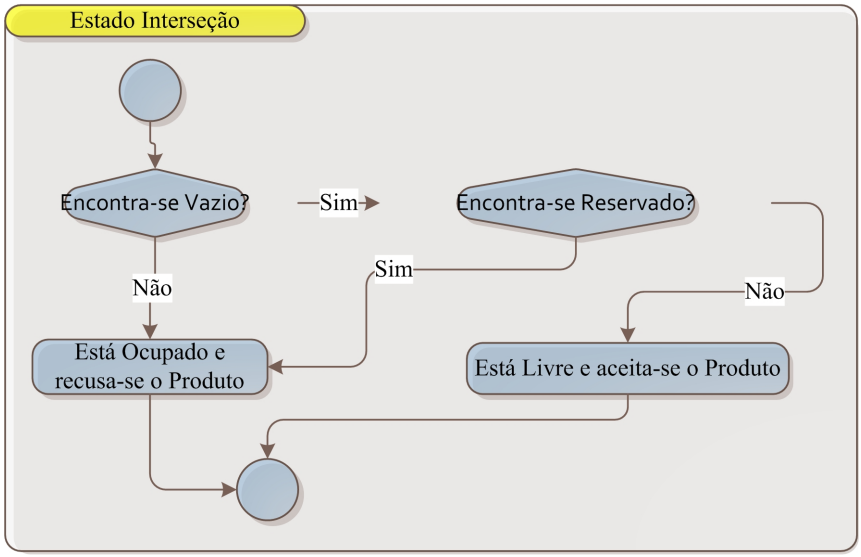

Figura 3.17: Estado da Interseção

O comportamento representado pela Figura 3.17 é inicializado através da recepção de uma mensagem por parte do *Conveyor*. Depois da verificação do conteúdo da mensagem recebida, a Intersecção verifica se está vazia e se não está reservada, caso estas condições se confirmem a Intersecção aceita o Produto, caso contrário recusa-o.

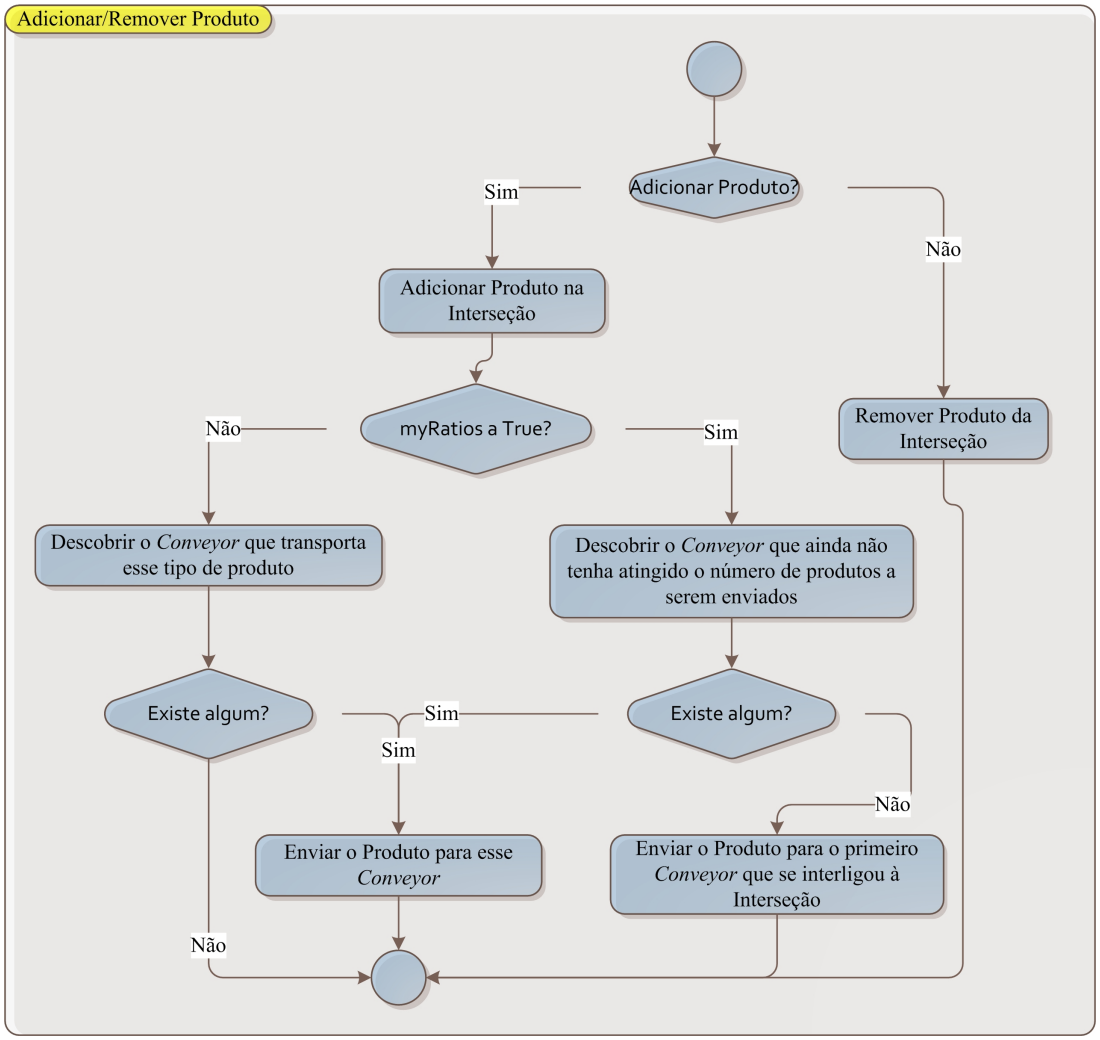

Figura 3.18: Adicionar ou Remover Produto

O comportamento representado pela Figura 3.18 é inicializado através da recepção de uma mensagem por parte do *Conveyor*. A Intersecção pode remover o Produto da mesma ou então adicionar o Produto, sendo que de seguida processa o custo de transporte de cada *Conveyor* de acordo com o tipo de Produto e com a configuração da Intersecção.

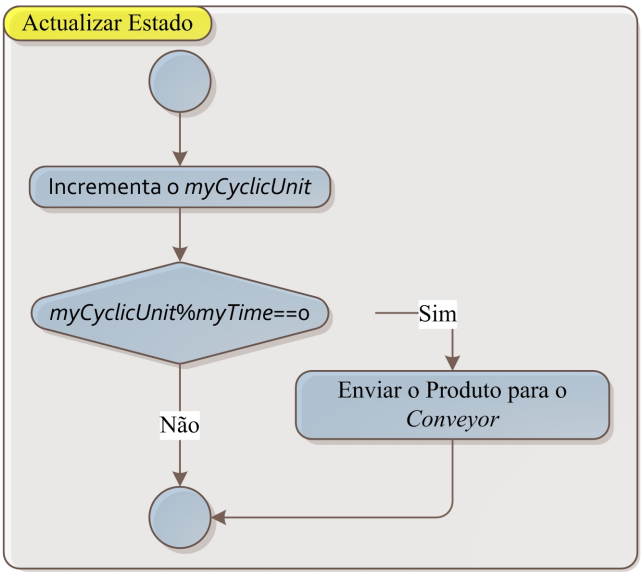

Figura 3.19: Actualizar o seu Estado

O comportamento representado pela Figura 3.19 é inicializado através da recepção de uma mensagem por parte do distribuidor de agentes de simulação. A Intersecção incrementa a variável *myCyclicUnit*, pois serve para indicar a contagem de *Clock's* dados pela simulação, de seguida verifica se passou o tempo necessário para que a Intersecção distribua o Produto para um determinado *Conveyor*, caso se confirme o Produto é enviado para o *Conveyor* de destino.

3.3.5 Distribuidor de Agentes de Visualização

O Distribuidor de Agentes de Visualização tem como objectivo receber a estrutura e características do sistema que se pretende simular que provêm da Aplicação Gráfica ou do Distribuidor de Agentes de Transporte, visto que o sistema encontra-se adaptado para interagir com qualquer tipo de sistema de Transporte externo e ainda inicializar, configurar e sincronizar todos os agentes de visualização, caso sejam necessários para representar o estado interno de cada elemento do sistema de manufactura. Esta informação é enviada através de uma mensagem para o Distribuidor de Agentes de Simulação para que sejam tomadas as respectivas acções no sistema.

3.3.6 Distribuidor de Agentes de Simulação

O Distribuidor de Agentes de Simulação tem como objectivo receber a estrutura e características do sistema que se pretende simular, que provém do Distribuidor de Agentes de Visualização e ainda inicializar, configurar e sincronizar os agentes de simulação. Este Distribuidor possui um registo de todos os agentes deste tipo, para que se possa efectuar uma gestão de carga de processamento uniforme entre os sistemas computacionais ligados à ferramenta desenvolvida.

3.3.7 Distribuidor de Agentes de História

O Distribuidor de Agentes de História é inicializado com o objectivo de receber a estrutura e características do sistema de manufactura e dos vários estados dos seus componentes durante uma determinada simulação, que provêm da Aplicação Gráfica. Esta informação é essencial para que os agentes necessários sejam inicializados e configurados com os seus estados históricos e para que posteriormente seja possível visualizar os resultados da simulação, efectuando a sincronização de todos os agentes envolvidos.

3.3.8 Distribuidor de Agentes de Transporte

O Distribuidor de Agentes de Transporte tem como objectivo construir a estrutura e definir as características do sistema que se pretende simular e controlar, sendo que esta mensagem é enviada para o Distribuidor de Agentes de Visualização para que este possa realizar as tarefas anteriormente referidas.

3.4 Adaptador de Controlo Externo

Como já foi referido anteriormente, o trabalho desenvolvido disponibiliza uma biblioteca de funções que permitem interligar qualquer sistema de transporte externo. Essa biblioteca, por sua vez, permite construir um modelo do sistema de manufactura que se pretenda simular e ainda controlar os *Conveyor* e as Intersecções que esse sistema possa vir a conter.

3.4.1 Funções de Construção do Sistema

Estas funções permitem ao sistema de transporte externo criar um modelo do sistema de manufactura de acordo com determinadas características e ligações. Neste caso, as funções são **LaunchSimulation()** (Permite enviar a descrição do sistema para o distribuidor de agentes de visualização através de troca de mensagens), **StartSimulation()** (Permite inicializar a simulação do sistema, enviando uma mensagem para o distribuidor de agentes de visualização), **StopSimulation()** (Permite parar a simulação do sistema, enviando uma mensagem para o distribuidor de agentes de visualização), **addConveyor(int id, double speed, double length)** (Permite a adição de um *Conveyor* no sistema com determinada velocidade e dimensão), **remConveyor(int id)** (Permite remover um qualquer *Conveyor* com determinado *id*), **modConveyor(int id, double speed, double length)** (Permite a alteração das características de um qualquer *Conveyor* com determinado *id*), **addHandover(int id, double time)** (Permite a adição de uma Intersecção no sistema com determinado tempo de processamento), **remHandover(int id)** (Permite remover uma qualquer Intersecção com determinado *id*), **modHandover(int id, double time)** (Permite a alteração do tempo de processamento de uma qualquer Intersecção com determinado *id*), **addStationConveyor(int id, double location, int idConveyor)** (Permite a

adição de uma *Docking Station* numa localização especifica de um qualquer *Conveyor* existente no sistema), **remStationConveyor(int id, int idConveyor)** (Permite a remoção de uma *Docking Station* de um qualquer *Conveyor* existente no sistema), **modStationConveyor(int id, double location, int idConveyor)** (Permite a modificação da localização de uma *Docking Station* de um qualquer *Conveyor* existente no sistema), **addConnectionConveyorHandover(int idOrigin, int idDestination)** (Permite adicionar uma ligação do tipo origem-destino entre um *Conveyor* e uma Intersecção, respectivamente), **remConnectionConveyorHandover(int idOrigin, int idDestination)** (Permite remover uma ligação entre um *Conveyor* e uma Interseção), **addConnectionHandoverConveyor(int idOrigin, int idDestination)** (Permite adicionar uma ligação do tipo origem-destino entre uma Intersecção e um *Conveyor*, respectivamente) e **remConnectionHandoverConveyor(int idOrigin, int idDestination)** (Permite remover uma ligação entre uma Intersecção e um *Conveyor*).

3.4.2 Funções de Controlo da Intersecção

Estas funções permitem ao sistema de transporte externo efectuar a gestão do comportamento da Intersecção. Neste caso, as funções são **routeBox(int idConveyorDestination)** (Permite enviar o *id* do *Conveyor* para onde deve ser encaminhado o Produto que se encontra na Intersecção, através de trocas de mensagens para a Intersecção que está no modo de simulação) e **haveBox()** (Permite verificar se a Intersecção está ocupada por um Produto ou não).

3.4.3 Funções de Controlo do *Conveyor*

Estas funções permitem ao sistema de transporte externo efectuar a gestão do comportamento do *Conveyor*. Neste caso, as funções são **activateStopper()** (Permite activar o travão de saída do *Conveyor*, através de trocas de mensagens para o *Conveyor* que está no modo de simulação), **haveBoxStopper()** (Permite verificar se o *Conveyor* possui um Produto ou não no travão de saída do *Conveyor*), **activateEntryStopperStation(int id)** (Permite activar o travão de entrada da *Docking Station* de um determinado *Conveyor*, através de trocas de mensagens para o *Conveyor* que está no modo de simulação), **haveBoxEntryStation(int id)** (Permite verificar se uma determinada *Docking Station* possui um Produto ou não no travão de entrada da mesma), **activateExitStopperStation(int id)** (Permite activar o travão de saída da *Docking Station* de um determinado *Conveyor*, trocas de mensagens para o *Conveyor* que está no modo de simulação), **haveBoxExitStation(int id)** (Permite verificar se uma determinada *Docking Station* possui um Produto ou não no travão de saída da mesma), **haveBoxStation(int id)** (Permite verificar se uma determinada *Docking Station* possui um Produto ou não), **addBoxStation(int id)** (Permite adicionar um Produto na *Docking Station* de um determinado *Conveyor*, através de trocas de mensagens para o *Conveyor* que está no modo de simulação) e **remBoxStation(int id)** (Permite remover um Produto da *Docking Station* de um determinado *Conveyor*, através

de trocas de mensagens para o *Conveyor* que está no modo de simulação).

3.5 Aplicação Gráfica

A Aplicação Gráfica consiste numa ferramenta capaz de representar o modelo lógico de qualquer sistema de manufactura e simular esse mesmo modelo sobre determinadas condições. Esta ferramenta possui dois modos de funcionamento de simulação: Autónoma ou Semiautónoma.

A Simulação Autónoma consiste em atribuir o controlo e gestão da simulação ao sistema de controlo da ferramenta, ou seja, os travões dos *Conveyor's* e também das *Docking Station's* e a distribuição dos Produtos é feita automaticamente, baseando-se na configuração definida pelo utilizador.

Enquanto que na Simulação Semiautónoma esse controlo e gestão é efectuado através do sistema de transporte externo, ou seja, é esse sistema que activa os travões dos *Conveyor's* e também das *Docking Station's* e ainda realiza o encaminhamento dos Produtos através de cálculos de métricas baseados no custos de transporte.

Apesar de haver esta diferenciação na maneira como a simulação é controlada, o modo de visualização é opcional em ambos os casos, ou seja, é possível adicionar ou retirar a componente de visualização do sistema antes da inicialização da simulação, com o intuito de reduzir a carga de processamento.

4

Implementação do Trabalho

Este capítulo apresenta e descreve os problemas e as soluções da implementação da arquitetura apresentada no Capítulo 3.

No trabalho desenvolvido foi usado um ambiente de programação designado por **NetBeans IDE**, *Netbeans Integrated Develepment Environment*, na linguagem de programação **JAVA**.

Para além deste ambiente de desenvolvimento e da linguagem de programação foi também usada a biblioteca **JADE**. Como foi referido, a arquitetura é constituída por um sistema de agentes, ou um **MAS**, para tal a arquitetura foi implementada efetuando a extensão de classes providenciadas pelo **JADE**.

O trabalho desenvolvido permite a criação de vários cenários de simulação que possibilitam testar diferentes abordagens de produção e transporte. Esta ferramenta possui três agentes que tratam da inicialização da biblioteca **JADE** e de outros agentes, criação e visualização da simulação da linha de montagem.

No início do desenvolvimento do trabalho efetuou-se a virtualização da célula **NOVAFLEX**, representada na Figura 4.1, visto ser uma linha de montagem complexa que possui inúmeros *Conveyor's*, estações, intersecções e produtos diferentes.

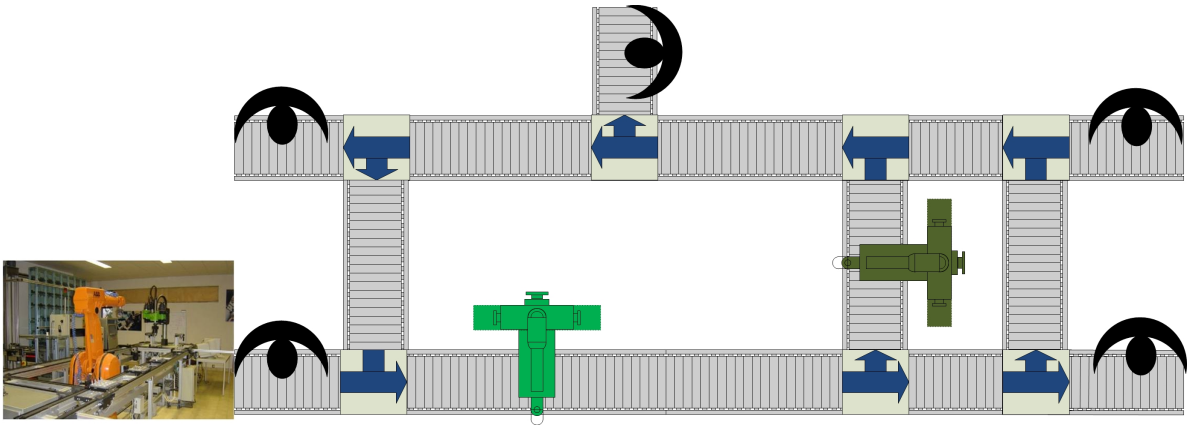

Figura 4.1: Célula NovaFlex

De acordo com a Figura 4.1 e com a Tabela 4.1, é possível verificar os tipos de agentes e o seu número necessários para simular a célula **NOVAFLEX**.

Tabela 4.1: Tipo e Número de Agentes na Célula NOVAFLEX

Tipo	**Instanciação**	**Número**
Pontos de Entrada	*Entry*	2
Pontos de Saída	*Exit*	3
Conveyors	*Conveyor*	13
Interseções	*Junction*	7
Simulação	*Simulation*	2
Visualização	*Visualization*	1
Total	-	28

Para que o trabalho efetue a virtualização desta célula é necessário criar o modelo lógico, ou seja, indicar os seus componentes, como *Conveyor's*, intersecções, estações, recursos e produtos que entram na célula, e as suas respetivas características. Esse modelo é convertido numa informação com um formato **XML**, como já foi detalhada no Capítulo 3.2.5. Através do modelo lógico, o sistema é capaz de estabelecer interações entre qualquer elemento adicionado à linha de montagem, cumprindo um conjunto de regras.

Este processo é facilitado usando a interface principal representada pela Figura 4.2.

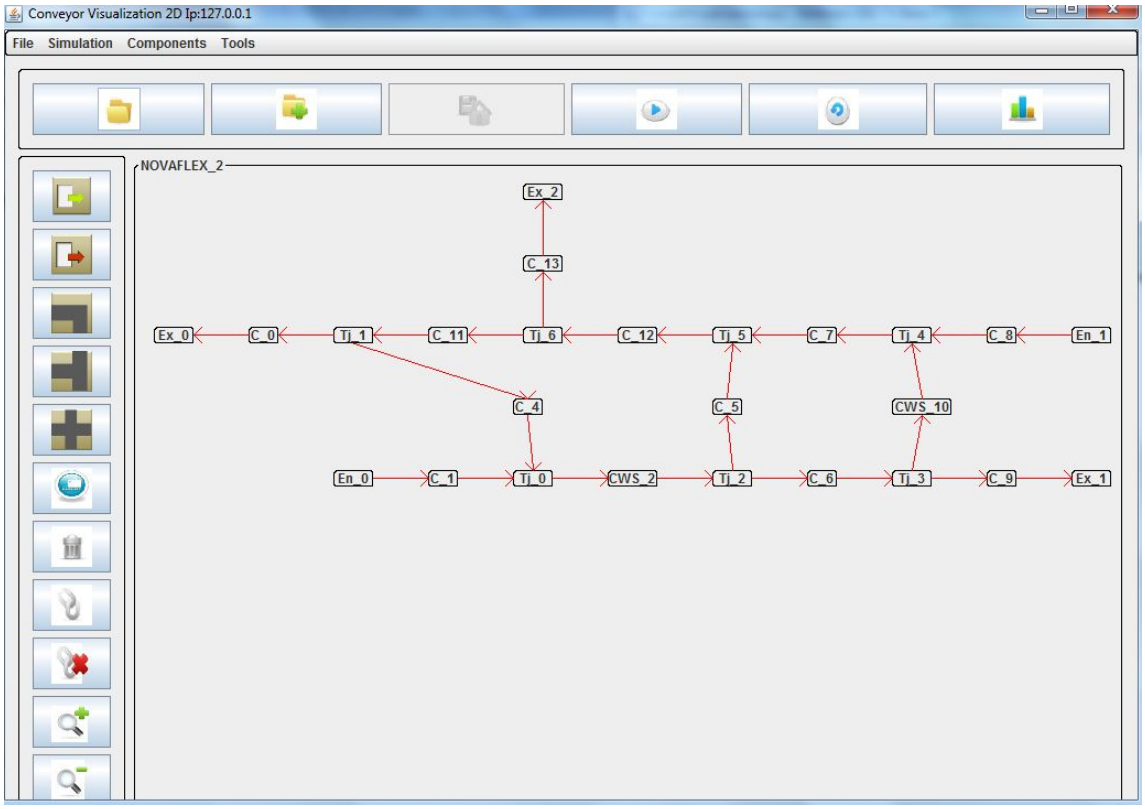

Figura 4.2: Interface Principal

Como se pode verificar pela Figura 4.2, através das ligações entre os componentes o utilizador consegue criar o modelo lógico para que se possa efetuar a simulação da linha da montagem. Para além de indicar quais os componentes existentes no sistema e as suas ligações, é necessário também definir os parâmetros de configuração de cada um.

4.1 Configuração de Componentes

Como já foi referido, é necessário configurar cada componente antes de ser lançado o sistema de simulação.

4.1.1 Ponto de Entrada

Através desta ferramenta, o utilizador possui determinadas características para o Ponto de Entrada que necessita de configurar. Características essas que são **Tempo de intervalo**, **Tipos de produtos** e se é **Cíclico**. Em que o **Tempo de intervalo** consiste num intervalo de tempo que o Ponto de Entrada tem de aguardar até que um novo produto seja disponibilizado para que este seja introduzido no sistema, este intervalo de tempo deve ser introduzido em segundos.

Os **Tipos de produtos** consistem em modelos de produtos pré-definidos que são apresentados em forma de lista ao utilizador. Sendo que, neste caso, o utilizador tem de escolher os tipos de produtos que pretende utilizar no sistema, à medida que for escolhendo os produtos, a ferramenta assume a ordem de seleção como a ordem de introdução no sistema, como se pode visualizar através da Figura 4.3, o primeiro produto a ser introduzido no sistema é o produto com o *id* **1**.

A característica **Cíclica** do Ponto de Entrada consiste num atributo que permite indicar se a lista de produtos é para ser percorrida novamente ou não, com o intuito de introduzir os produtos no sistema.

Estas definições necessárias para o Ponto de Entrada são exibidas pela Figura 4.3

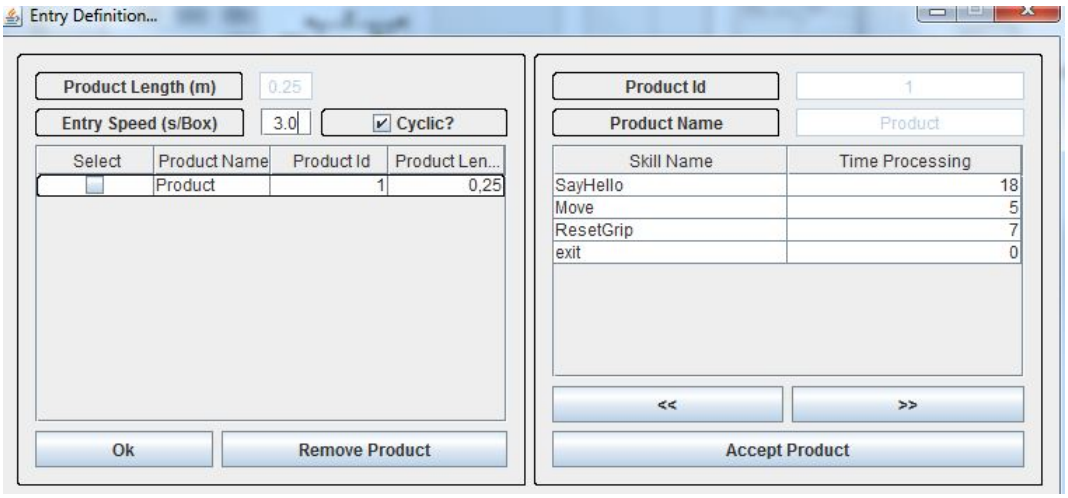

Figura 4.3: Definição das características do Ponto de Entrada

4.1.2 *Conveyor*

O Conveyor precisa também de ser configurado, como tal, contem determinadas características, tais como **Comprimento**, **Velocidade**, **Orientação** e **Lista de *Docking Stations***. Em que o **Comprimento** consiste no tamanho que o **Conveyor** possui para unir outros dois componentes do sistema, esta característica deve ser introduzido em metros. A **Velocidade** relaciona a variação da posição no espaço em relação ao tempo, ou seja, neste caso, qual a distância percorrida por um produto num determinado intervalo de tempo, este valor deve ser introduzido em metros/segundo.

A **Orientação** permite à ferramenta posicionar o desenho gráfico do *Conveyor* no modelo físico do sistema, sendo que as opções variam entre posicionamento **Horizontal** ou **Vertical**.

Por último, a **Lista de *Docking Stations*** permite a introdução, remoção ou modificação de *Docking Stations* que o *Conveyor* possa conter, sendo que este assunto será abordado a seguir.

Estas definições necessárias para o *Conveyor* são exibidas pela Figura 4.4.

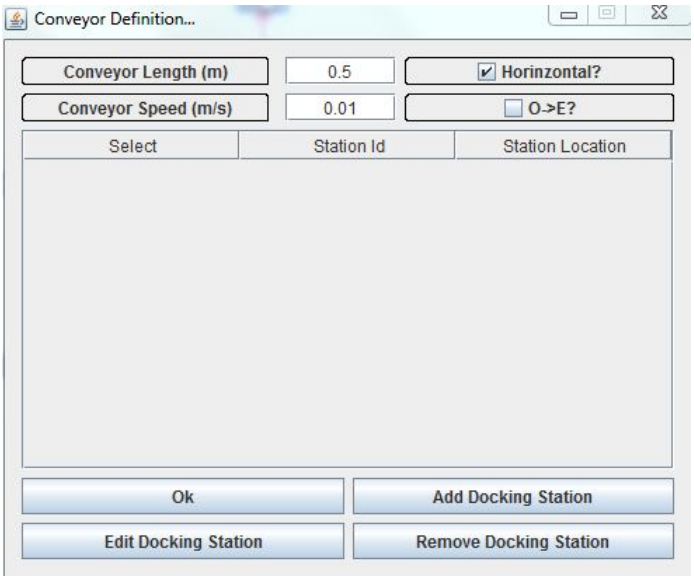

Figura 4.4: Definição de características do *Conveyor*

4.1.3 *Docking Station*

Como já foi referido, o *Conveyor* pode conter *Docking Stations* com o objetivo de adicionar recursos ao sistema, de maneira a melhorar a produção de um produto. Para além, de adicionar recursos pré-definidos, que são apresentados em forma de lista, é possível também definir a localização da *Docking Station* no *Conveyor* que pertence, este valor é introduzido em metros, como se pode verificar na Figura 4.5.

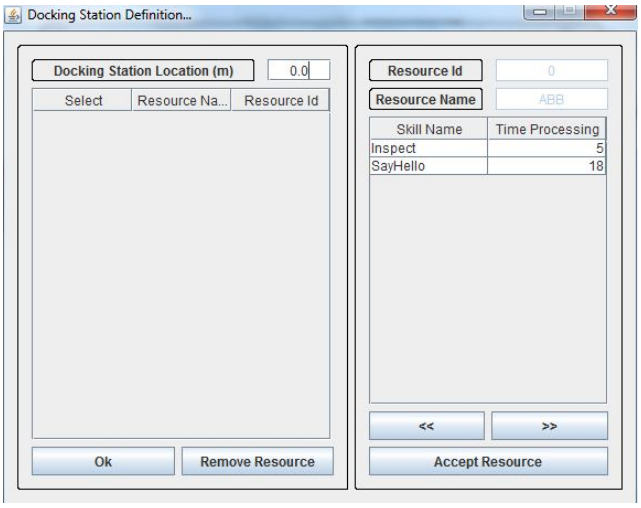

Figura 4.5: Definição das características do *Docking Station*

4.1.4 Intersecção

Para as intersecções, as características necessárias são um bocado mais complexas, devido ao encaminhamento das paletes serem feitas neste componente, ou seja, é aqui que é feita a distribuição das paletes pelos *Conveyors*. Através desta ferramenta, o utilizador possui determinadas características para a Intersecção que necessita de configurar. Características essas que são **Tempo de intervalo** e **Modo de Encaminhamento**. Em que o **Tempo de intervalo** consiste num intervalo de tempo que a Intersecção necessita para calcular uma nova rota para o produto, este valor é introduzido em segundos.

Relativamente ao **Modo de Encaminhamento**, a Intersecção possui dois modos de funcionamento que são por **Número** ou por **Tipo** de produtos. Em que o modo de encaminhamento por **Número** consiste em definir um número fixo de produtos que devem ser distribuídas para cada *Conveyor*. Enquanto que o modo de encaminhamento por **Tipo** consiste em definir quais os tipos de produtos que devem ser distribuídos para cada *Conveyor*. De acordo com a Figura 4.6, é possível verificar que a Intersecção, neste caso, encontra-se configurado para o modo de encaminhamento por **Número** em que os *Conveyor's* com os *id's* 11 e 13 têm de receber pelo menos um produto de qualquer tipo. Se a Intersecção fosse configurada para o encaminhamento por **Tipo**, como é apresentada na Figura 4.6, apenas o *Conveyor* com o *id* 11 consegue receber produtos com o *id* 2.

Estas definições necessárias para a Intersecção são exibidas pela Figura 4.6.

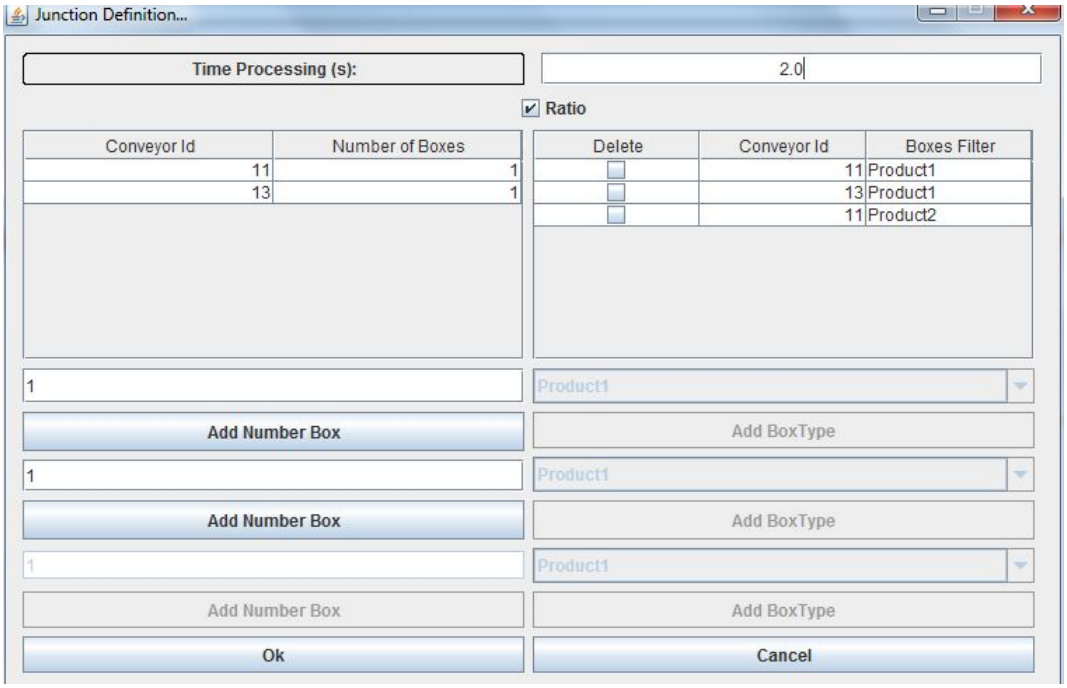

Figura 4.6: Definição das características da Interseção

Tendo estas configurações corretamente introduzidas, o sistema de simulação validará esses parâmetros e o também o modelo físico construído através de regras de ligações e de características dos componentes, ou seja, um Ponto de Entrada não pode estar ligado diretamente a um Ponto de Saída sem que exista um *Conveyor* entre eles, ou um *Conveyor* não pode estar ligado a outro, pois isso significava que era o mesmo, uma Intersecção tem de ter mais do uma ligação, o Ponto de Entrada tem de conter uma lista de produtos, entre outras regras.

4.2 Gestão da Simulação

Após as configurações necessárias do modelo lógico e dos seus componentes estarem completas, o comportamento do sistema é o seguinte: a Simulação, instanciada como *Simulation*, cria o **Main Container** que serve para inicializar e configurar os agentes necessários para a simulação do sistema, e que por sua vez, aguarda pela informação do modelo lógico do sistema a simular que provém da a Visualização, instanciada como *Visualization* ou do registo de novos servidores de Simulação.

Na receção da informação do sistema a simular, a Simulação distribui os agentes de simulação, para um melhor desempenho a nível de processamento, por entres os outros servidores de Simulação, sendo que também possui uma tarefa importante, que consiste em sincronizar todas as ações de todos os agentes de simulação. Esses servidores inicializam e configuram os agentes de simulação, como os Pontos de Entrada, de Saída, *Conveyor's* e intersecções, sendo que de seguida informam outros servidores a localização

desses mesmos agentes no sistema.

Após o envio dessa informação, a Visualização também inicializa os agentes de visualização, para que seja possível verificar o estado interno do sistema a simular, ao longo do tempo. A Visualização ainda possui uma tarefa importante neste sistema, que consiste em sincronizar todas as ações de todos os agentes de visualização.

Todo este processo é possível observar-se na Figura 4.7.

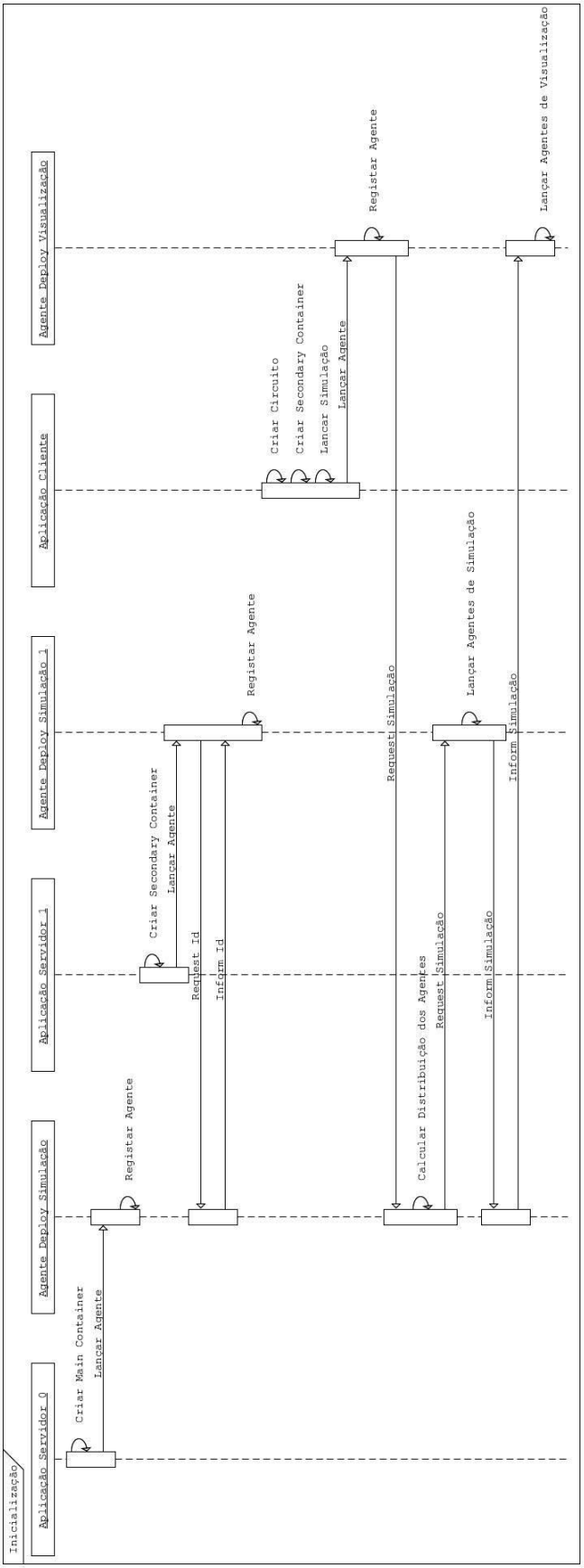

Figura 4.7: Inicialização da simulação

Em comparação entre a simulação definida pelo sistema de transporte externo e o processo anterior, os processos de inicialização do sistema de simulação são muito idênticos, sendo que a construção da linha de montagem é feito pelo sistema de transporte e não pelo sistema de visualização. Que por sua vez, o sistema de transporte envia a informação sobre o sistema a simular para a Visualização, e a partir do momento da receção dessa informação, a Visualização comporta-se da mesma maneira, referida anteriormente.

Existe um painel implementado que possui funções que controlam a biblioteca **JADE**. Em particular permite inicializar o agente de Simulação e os seus agentes de simulação em qualquer máquina. Esse painel é representado na Figura 4.8.

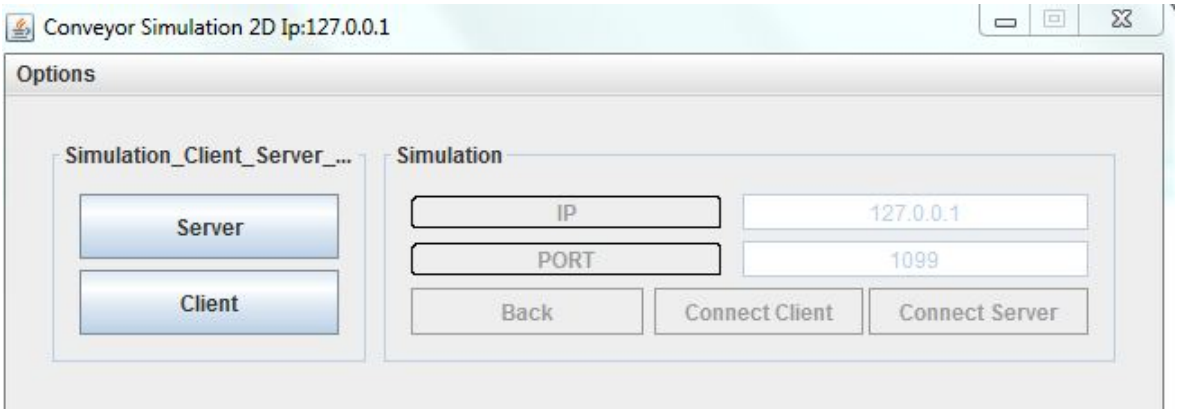

Figura 4.8: Painel Principal da Simulação

De acordo com a Figura 4.8, o utilizador tem a opção de inicializar o **JADE** criar o *Main Container* e inicializar o agente de Simulação, para que seja possível lançar outros agentes de Simulação e registá-los. Podendo apenas haver um *Main Container* e se já existe um no sistema, aparece a opção de introduzir o endereço de IP e o Porto da máquina de destino para que seja possível lançar outros *Containers*.

Após a inicialização dos *Containers* e dos agentes de Simulação, o utilizador pode desenhar o modelo lógico do sistema que pretende simular. Como o projeto permite a interligação de qualquer sistema de transporte alheio, o utilizador possui duas opções para controlar a linha de montagem. No primeiro caso, o utilizador projeta a linha de montagem usando o modelo lógico e define as características das intersecções para orientar os produtos dentro do sistema, como já foi referido. No segundo caso, o utilizador interliga a ferramenta com o sistema de transporte inicializando o agente de Simulação para este caso, que logo será detalhado.

Antes de se inicializar o sistema de visualização é necessário configurar primeiro os parâmetros de simulação, que se encontram na aplicação Cliente, como se pode verificar na Figura 4.9.

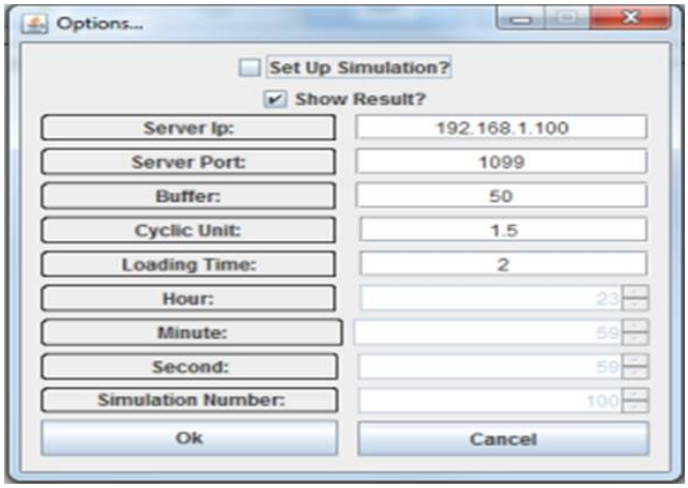

Figura 4.9: Configuração de parâmetros da Simulação

Esses parâmetros são:

1. Visualizar o estado interno da linha de montagem ou apenas simular o comportamento da mesma.
2. Definir se o tipo de simulação é contínuo, ou seja, o sistema não termina a simulação do comportamento da linha, por si própria, ou se é limitado, ou seja, o utilizador define o tempo de simulação pretendido, neste caso o utilizador também pode definir o número de vezes que a linha deve ser simulada dentro do tempo definido.
3. Definir o endereço de IP e o Porto de comunicação da máquina de destino no qual se encontra o sistema de simulação.
4. Definir o tempo que é atribuído a cada ciclo de tempo do sistema de simulação.
5. Definir o tempo de espera que o sistema de visualização deve esperar depois de enviar a linha de montagem para o sistema de simulação, como já foi referido anteriormente.
6. Definir o tamanho do *buffer* de estados de cada componente deve guardar antes de os enviar para o sistema de visualização.

4.3 Modo de Simulação

Toda a linha de montagem, no modo físico, é visualizada num painel principal representada Figura 4.10. Este painel auxilia na perceção do que está a acontecer no interior da linha de montagem projetada.

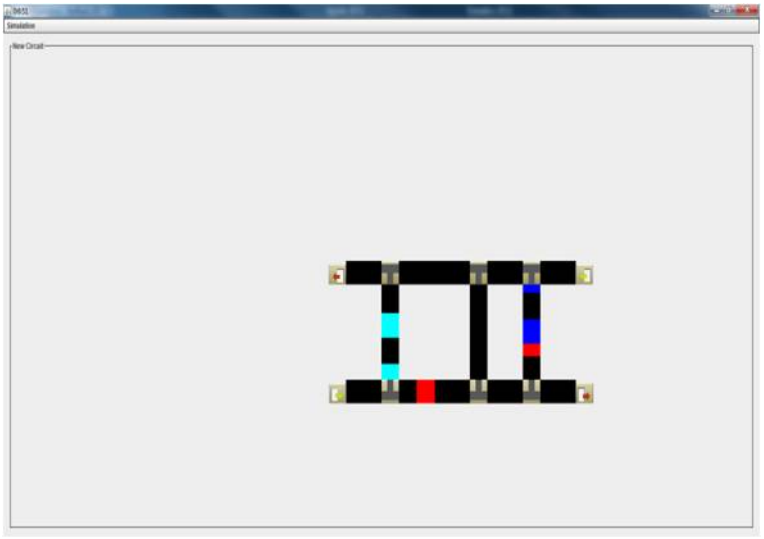

Figura 4.10: Simulação da célula NOVAFLEX

O painel de simulação da Figura 4.10 foi implementado usando uma biblioteca de representação gráfica do **JAVA** e permite uma representação gráfica do estado interno do sistema que se pretende simular. Cada agente é representado por uma figura representada pela Tabela 4.2.

Tabela 4.2: Representação Gráfica de Cada Componente

Tipo	**Desenho**
Ponto de Entrada	
Ponto de Saída	
Conveyor	 
Interseção	

Neste projeto a informação sobre o estado dos componentes é representado na Tabela 4.2, como já foi referido, e essa informação não serve simplesmente para a visualização mas também é importante para analise estatística, pois permite calcular valores estatísticos como o valor médio de produtos que se deslocam e que saem no *Conveyor*, se existe acumulação, etc..

5

Casos de Uso

Neste capítulo são descritos e explicados as técnicas de validação efetuadas sobre este trabalho. São ainda apresentados os testes realizados e os resultados obtidos pelo trabalho em estudo.

A criação das células a simular é feita tendo em conta a quantidade e o tipo de componentes pretendidos e referidos anteriormente e a conectividade pretendida entre eles. Os componentes são adicionados ao modelo lógico e, posteriormente, são atribuídas as suas relações com outros componentes, sendo depois enviado o modelo para a camada de visualização com o objetivo de inicializar o sistema.

No sentido de testar o comportamento do sistema, face a estas células e características, optou-se por interligar o sistema de transporte desenvolvido no projeto **IDEAS** ao trabalho. Este sistema de transporte constrói o modelo lógico da célula a simular, sendo depois enviado ao sistema de visualização para iniciar a tarefa de simulação.

Os nomes dos testes de Casos de Usos são:

1. Célula Nº 1;
2. Célula Nº 2;
3. Célula Nº 3;

Para efetuar a validação deste trabalho escolheram-se os seguintes casos:

5.1 Caso Célula Nº 1

O sistema a simular neste caso é constituída com várias estações, em que cada uma está habilitada a realizar uma distinta e única tarefa no sistema em geral. A direção do fluxo dos produtos são indicados pelas setas azuis, como são demonstradas através da Figura 5.1.

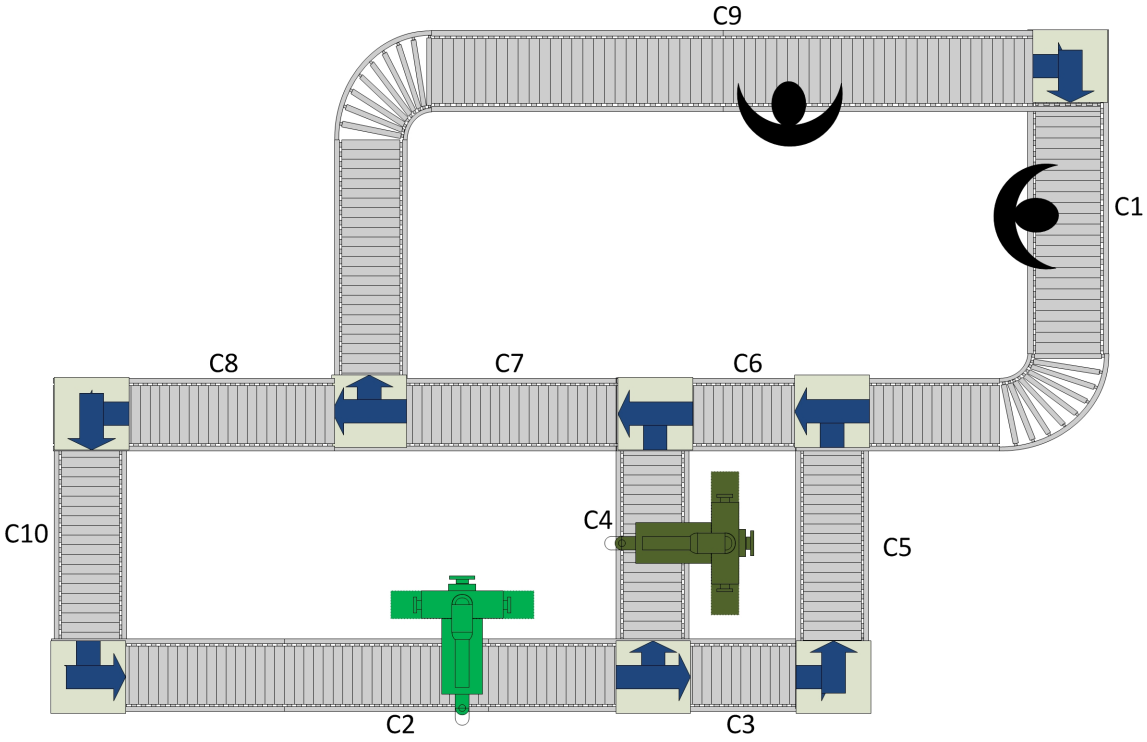

Figura 5.1: Caso Célula Nº 1

Através da Figura 5.1, verifica-se que o sistema possui poucos caminhos alternativos para os tipos de produtos projetados e que existem duas estações que são responsáveis pela introdução e remoção dos produtos no sistema.

Esta célula foi simulada com 51 agentes no total, sendo que na Tabela 5.1 encontram-se enumerados os agentes de simulação e os agentes de transporte necessários para a simulação do mesmo.

Tabela 5.1: Tipo e Número de Agentes no Caso Célula Nº 1

Tipo	Número
Conveyors	20
Interseções	16
Produtos	10
Recursos	2
Simulação	2
Visualização	1
Total	51

5.1.1 Caraterísticas

Cada Estação/Recurso consegue realizar *Skills* em apenas um produto de cada vez, enquanto cada *Conveyor* consegue manipular vários produtos, pois possuem tamanho diferentes.

De acordo com a Tabela 5.2, é possível comprovar as características dos vários componentes, especificamente, os *Conveyors* que constituem a linha de montagem a simular. Sendo que neste caso foi normalizado o tamanho dos produtos para 0.25 m.

Tabela 5.2: Descrição dos *Conveyors* do Caso Célula Nº 1

Conveyor (Id)	Tamanho (Produtos)	Velocidade (m/s)
1	8	0.121
2	8	0.121
3	3	0.121
4	8	0.121
5	8	0.121
6	3	0.121
7	4	0.121
8	3	0.121
9	15	0.121
10	8	0.121

De acordo com a Tabela 5.3 é possível comprovar as características das estações, como localização no *Conveyor* que se encontra conectado, as *Skills* que executa, etc. que constituem a linha de montagem a simular. Sendo possível, realocar as estações em qualquer localização no *Conveyor*.

Tabela 5.3: Descrição das Estações do Caso Célula Nº 1

Estação (Id)	*Skill*	*Conveyor (Id)*	**Localização no *Conveyor* (Produto)**
1	*Load* & *UnLoad*	1	2
2	*UnLoad* & *Load*	9	11
3	B	2	6
4	A	4	6

Para este caso também foram escolhidos dois tipos de produtos a circularem na linha que são representados pela Tabela 5.4, sendo que estes são introduzidos no sistema, assim que for possível, no *Conveyor* a que as estações, que efetuam a introdução de produtos, encontram-se ligadas. Estes tipos de produtos são introduzidos nos sistema com um *rácio* de 1:1.

Tabela 5.4: Descrição dos Produtos do Caso Célula Nº 1

Produto (Id)	*Skills*
1	BAB
2	AB

5.1.2 Resultados e Análise

Neste subcapítulo são apresentados os resultados das simulações efetuadas sobre o caso anterior. Sendo que foram efetuadas 11 simulações, que demoraram 7 minutos. Durante cada simulação a adição dos produtos foi progressiva e sem interrupções, como já foi referido anteriormente. O controlo do sistema foi efetuado através do sistema de transporte do projeto **IDEAS**.

De seguida, são apresentados os gráficos com os valores de *Throughput*, o número de produtos e se ocorre acumulação em cada *Conveyor*. E ainda, é apresentada uma análise dos resultados obtidos.

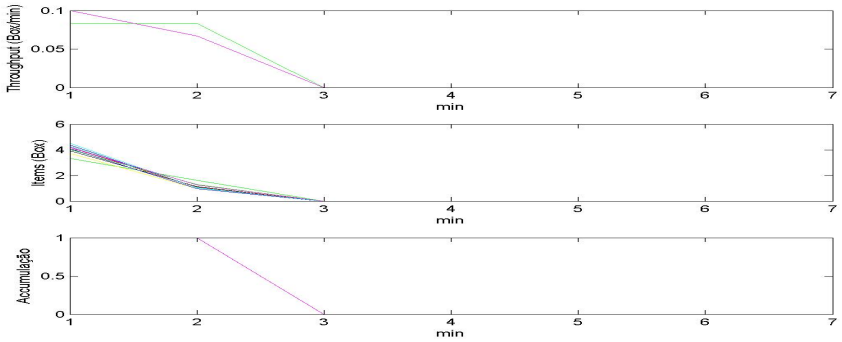

Figura 5.2: Resultados da Simulação do *Conveyor 1* no Caso da Célula Nº 1

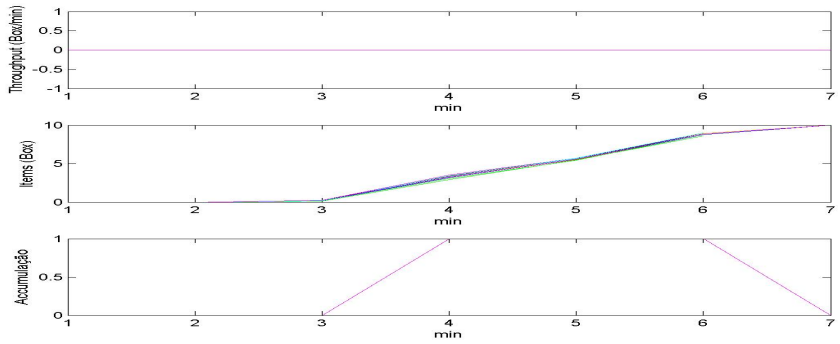

Figura 5.3: Resultados da Simulação do *Conveyor 9* no Caso da Célula Nº 1

É possível verificar que o *Conveyor 1* (Figura 5.2) demorou cerca de 3 minutos a introduzir os 10 produtos no sistema. Sendo que este *Conveyor* durante toda a simulação nunca possui mais produtos, devido ao final de cada produto este ser encaminhado para a estação do *Conveyor 9* (Figura 5.3).

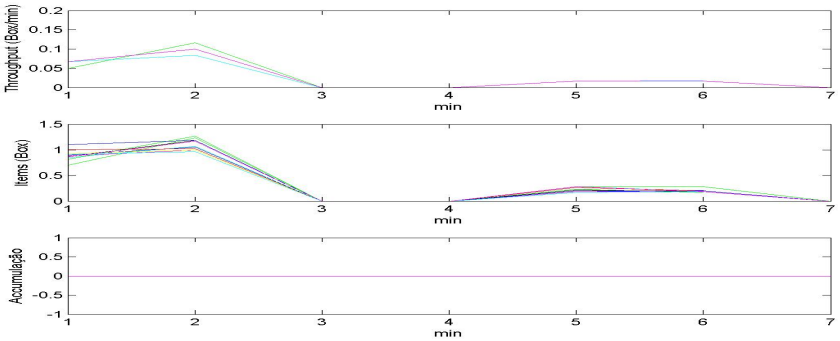

Figura 5.4: Resultados da Simulação do *Conveyor 6* no Caso da Célula Nº 1

No *Conveyor 6 (Figura 5.4)* é possível verificar a recirculação dos produtos no minuto 4, por causa do plano de *Skills* de cada produto e à disposição de cada estação no sistema.

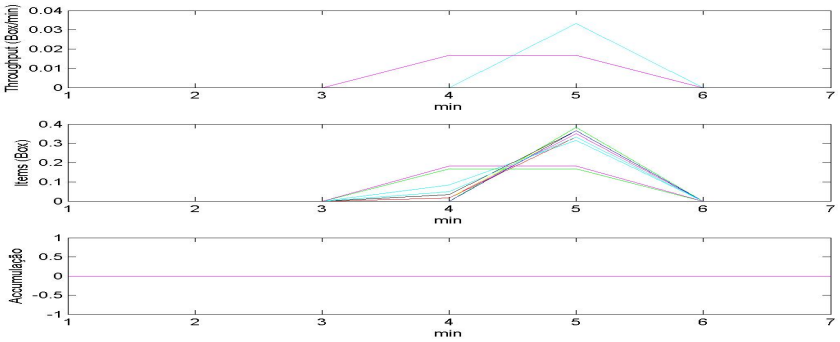

Figura 5.5: Resultados da Simulação do *Conveyor 3* no Caso da Célula Nº 1

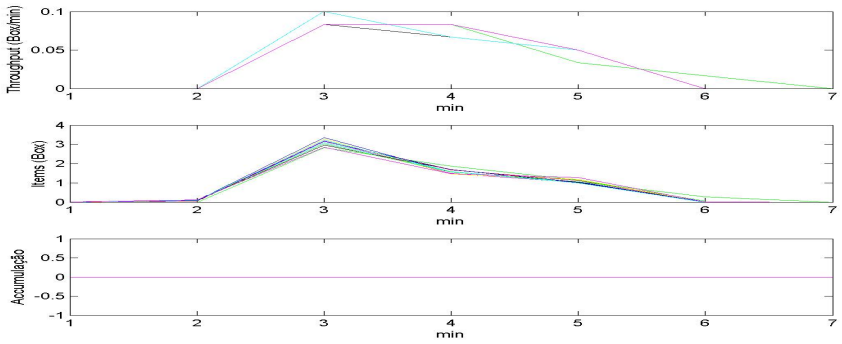

Figura 5.6: Resultados da Simulação do *Conveyor 4* no Caso da Célula Nº 1

Essa recirculação deve-se ao fato do plano do primeiro tipo de produto ter de efetuar mais uma volta ao sistema para executar a *Skill* **B**, sendo que a sua *Skill* anterior faça com que o *Conveyor 4 (Figura 5.6)* esteja mais ocupado nos últimos momentos da simulação. Sendo assim, os produtos do segundo tipo terão de ser encaminhados para o *Conveyor 3 (Figura 5.5)* em vez do *Conveyor 4 (Figura 5.6)*, começando o momento da recirculação que ocorre *Conveyor 6 (Figura 5.4)*.

Nestes casos, o controlador escolheria o *Conveyor 3 (Figura 5.5)* face ao *Conveyor 4 (Figura 5.6)* para os produtos do segundo tipo, pois se não fosse assim o *Conveyor 3 (Figura 5.5)* começaria a acumular produtos.

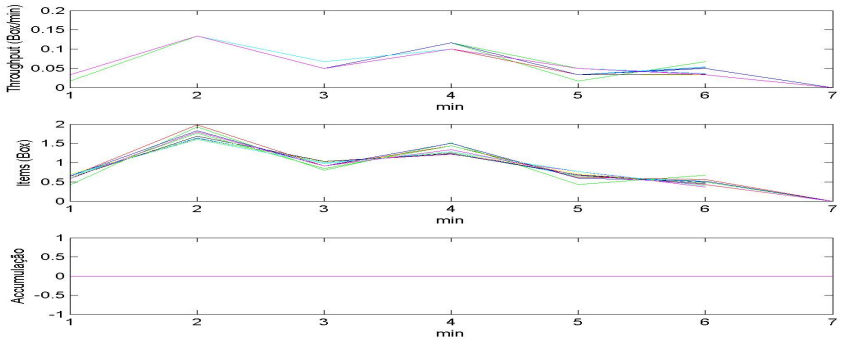

Figura 5.7: Resultados da Simulação do *Conveyor 7* no Caso da Célula Nº 1

O *Conveyor 7 (Figura 5.7)* consiste num canal que une dois *Conveyors*, logo o seu estado ao longo da simulação reflete o efeito de recirculação, como se pode ver no pico ao minuto 4.

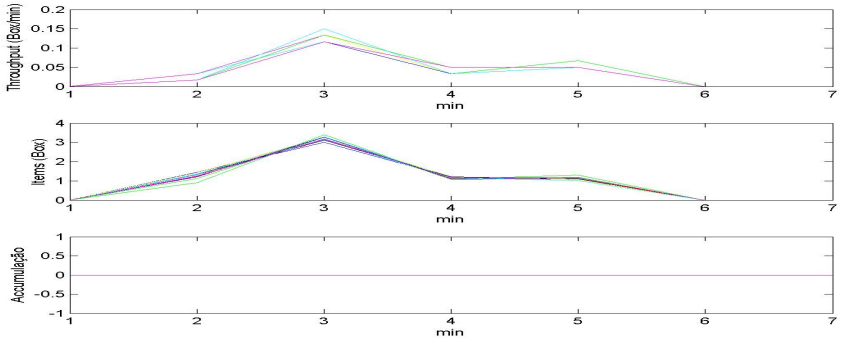

Figura 5.8: Resultados da Simulação do *Conveyor 2* no Caso da Célula Nº 1

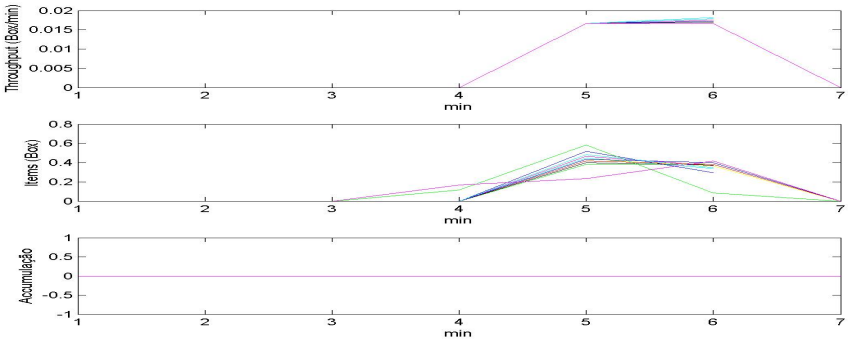

Figura 5.9: Resultados da Simulação do *Conveyor 5* no Caso da Célula Nº 1

Relativamente ao *Conveyor 4 (Figura 5.6)* é o mais usado no início da simulação que o *Conveyor 5 (Figura 5.9)* pois os produtos executam a *Skill* **A** e de seguida voltam para o *Conveyor 2 (Figura 5.8)* para a *Skill* **B**, ocorrendo a recirculação até ao minuto 5.

O primeiro pico de atividade do *Conveyor 2 (Figura 5.8)* acontece devido ao facto de todos os produtos terem de ser encaminhados para o mesmo. Depois verifica-se a recirculação entre o minuto 4 e 5.

A carga entre os *Conveyor 4 (Figura 5.6)*, *5 (Figura 5.9)* é balanceada entre os dois, visto que no final da simulação o *Conveyor 5 (Figura 5.9)* comporta-se como um caminho alternativo, enquanto os produtos não chegam à penúltima *Skill* a ser executada.

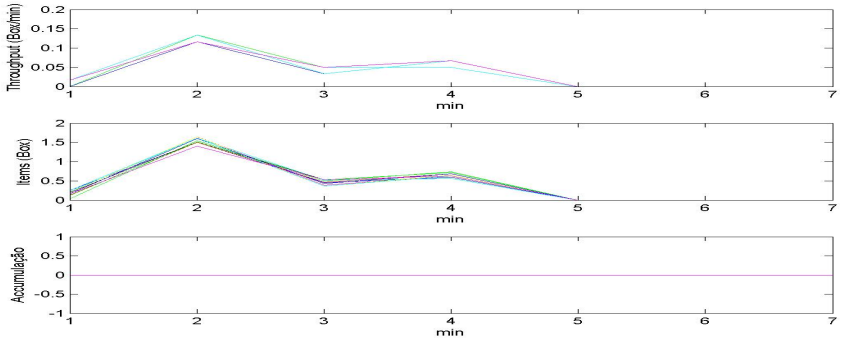

Figura 5.10: Resultados da Simulação do *Conveyor 8* no Caso da Célula Nº 1

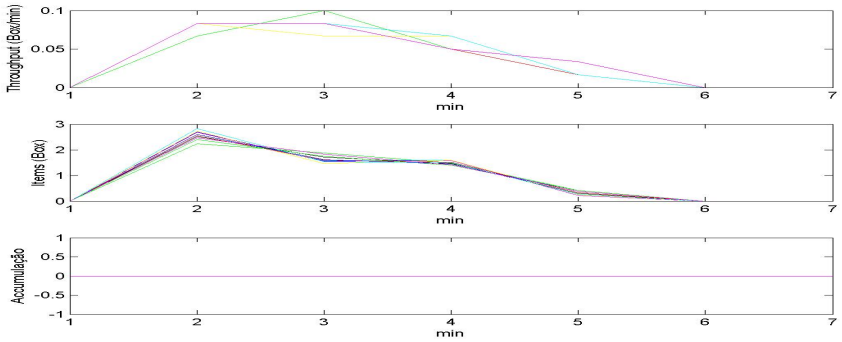

Figura 5.11: Resultados da Simulação do *Conveyor 10* no Caso da Célula Nº 1

De acordo com os gráficos apresentados anteriormente, é possível concluir que o comportamento de todos os agentes de *Conveyor* e do sistema de transporte foi sempre consistente ao longo de todas as simulações, e houve um balanceamento da carga de produtos no sistema.

5.2 Caso Célula Nº 2

Neste caso, o sistema a simular possui mais estações que o caso anterior, em que cada uma está habilitada a realizar uma distinta e única tarefa no sistema em geral. A direção do fluxo dos produtos são indicados pelas setas azuis, como são demonstradas através da Figura 5.12.

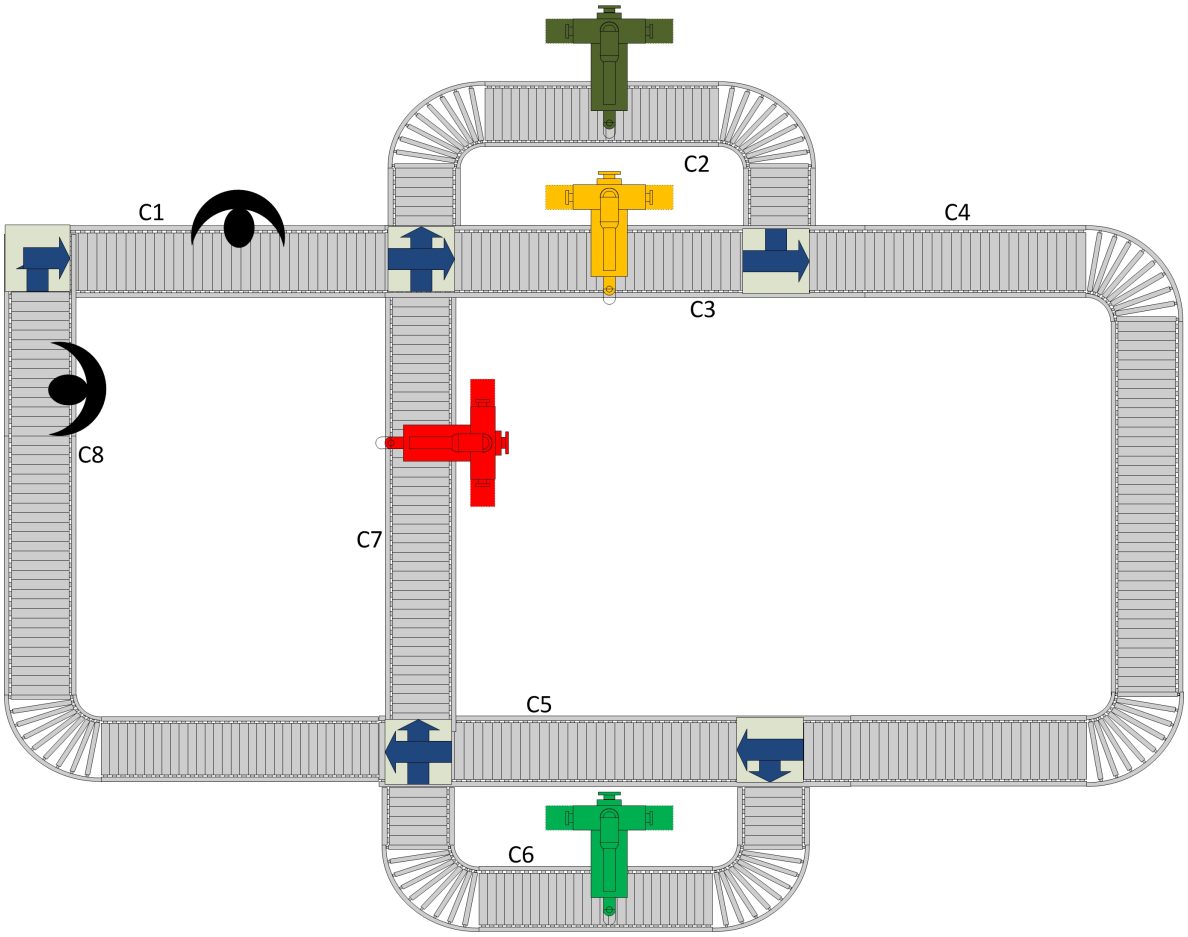

Figura 5.12: Caso Célula Nº 2

Através da Figura 5.12, verifica-se que o sistema possui poucos caminhos alternativos para os tipos de produtos projetados e que existem duas estações que são responsáveis pela introdução e remoção dos produtos no sistema.

Esta célula foi simulada com 48 agentes no total, sendo que na Tabela 5.5 encontram-se enumerados os agentes de simulação e os agentes de transporte necessários para a simulação do mesmo.

Tabela 5.5: Tipo e Número de Agentes no Caso Célula Nº 2

Tipo	**Número**
Conveyors	16
Interseções	10
Produtos	15
Recursos	4
Simulação	2
Visualização	1
Total	48

5.2.1 Caraterísticas

Cada Estação/Recurso consegue realizar *Skills* em apenas um produto de cada vez, enquanto cada *Conveyor* consegue manipular vários produtos, pois possuem tamanho diferentes.

De acordo com a Tabela 5.6, é possível comprovar as características dos vários componentes, especificamente, os *Conveyors* que constituem a linha de montagem a simular. Sendo que neste caso também foi normalizado o tamanho dos produtos para 0.25 m.

Tabela 5.6: Descrição dos *Conveyors* do Caso Célula Nº 2

Conveyor (Id)	**Tamanho (Palete)**	**Velocidade (m/s)**
1	8	0.121
2	8	0.121
3	8	0.121
4	14	0.121
5	8	0.121
6	8	0.121
7	8	0.121
8	20	0.121

De acordo com a Tabela 5.7 é possível comprovar as características das estações, como localização no *Conveyor* que se encontra conectado, as *Skills* que executa, etc. que constituem a linha de montagem a simular. Sendo possível, realocar as estações em qualquer localização no *Conveyor*.

Tabela 5.7: Descrição das Estações do Caso Célula Nº 2

Estação (Id)	***Skill***	***Conveyor (Id)***	**Localização no *Conveyor* (Palete)**
1	*Load* & *UnLoad*	1	2
2	*UnLoad* & *Load*	8	16
3	A	2	6
4	B	3	6
5	C	6	6
6	D	7	6

Para este caso também foram escolhidos três tipos de produtos a circularem na linha que são representados pela Tabela 5.8, sendo que estes são introduzidos no sistema, assim que for possível, no *Conveyor* a que as estações, que efetuam a introdução de produtos, encontram-se ligadas. Estes tipos de produtos são introduzidos nos sistema com um *rácio* de 1:1:1.

Tabela 5.8: Descrição dos Produtos do Caso Célula Nº 2

Produto (Id)	*Skills*
1	ACBD
2	BACD
3	CBAD

5.2.2 Resultados e Análise

Neste subcapítulo são apresentados os resultados das simulações efetuadas sobre o caso anterior. Sendo que foram efetuadas 21 simulações, que demoraram 12 minutos. Durante cada simulação a adição dos produtos foi progressiva e sem interrupções, como já foi referido anteriormente. O controlo do sistema foi efetuado através do sistema de transporte do projeto **IDEAS**.

De seguida, são apresentados os gráficos com os valores de *Throughput*, o número de produtos e se ocorre acumulação em cada *Conveyor*. E ainda, é apresentada uma análise dos resultados obtidos.

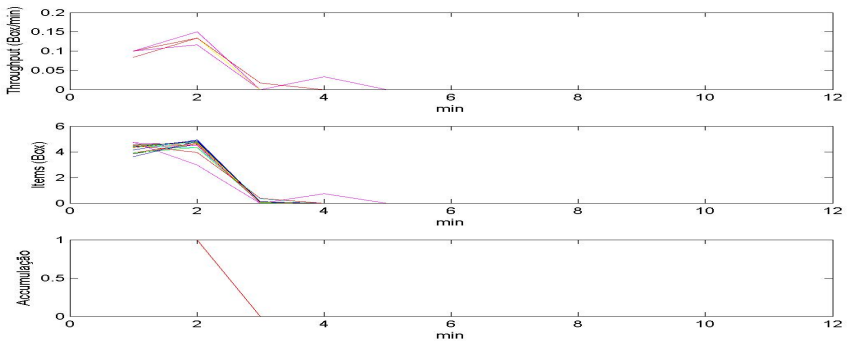

Figura 5.13: Resultados da Simulação do *Conveyor 1* no Caso Célula Nº 2

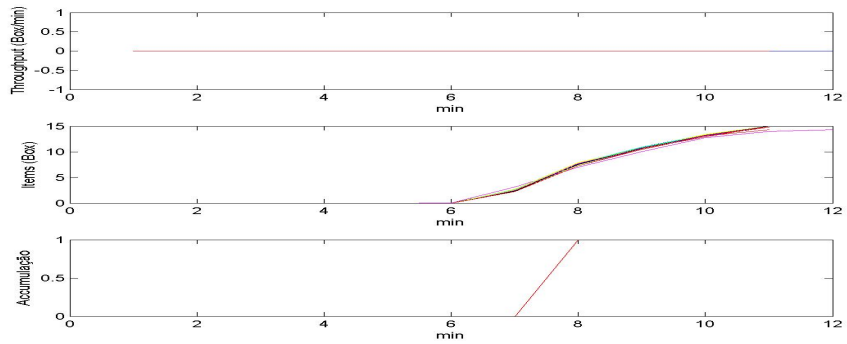

Figura 5.14: Resultados da Simulação do *Conveyor 8* no Caso Célula Nº 2

É possível verificar que o *Conveyor 1* (Figura 5.13) demorou cerca de 4 minutos a introduzir os 15 produtos no sistema. Sendo que este *Conveyor* durante toda a simulação nunca possui mais produtos, devido ao final de cada produto este ser encaminhado para a estação do *Conveyor 8* (Figura 5.14).

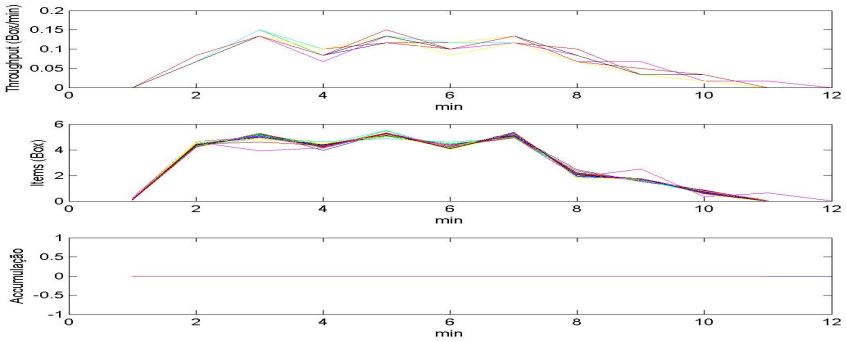

Figura 5.15: Resultados da Simulação do *Conveyor 4* no Caso Célula Nº 2

No *Conveyor 4 (Figura 5.15)* é possível verificar a recirculação dos produtos no minuto 5, por causa do plano de *Skills* de cada produto e à disposição de cada estação no sistema.

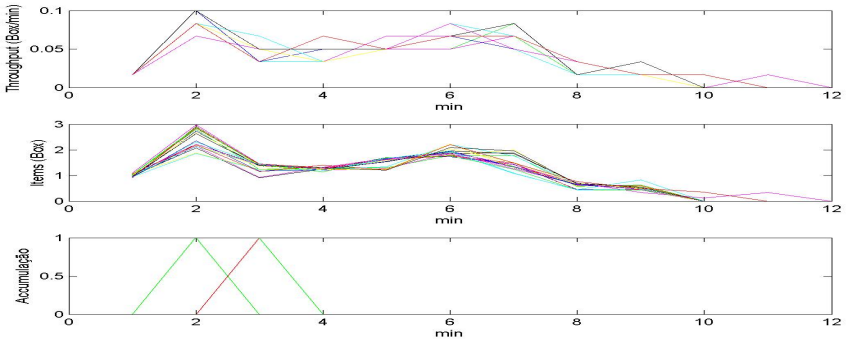

Figura 5.16: Resultados da Simulação do *Conveyor 2* no Caso Célula Nº 2

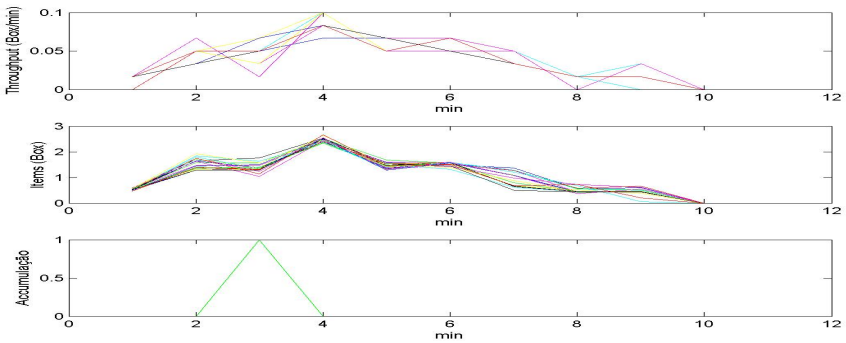

Figura 5.17: Resultados da Simulação do *Conveyor 3* no Caso Célula Nº 2

Essa recirculação deve-se ao fato do plano do primeiro tipo de produto ter de efetuar mais uma volta ao sistema para executar a *Skill* **B**, sendo que o último tipo de produto faça com que o *Conveyor 2 (Figura 5.16)* esteja mais ocupado nos últimos momentos da simulação. Sendo assim, os outros tipos de produtos terão de ser encaminhados para o *Conveyor 3 (Figura 5.17)* em vez do *Conveyor 2 (Figura 5.16)*, começando o momento da recirculação que ocorre *Conveyor 4 (Figura 5.15)*.

Nestes casos, o controlador escolheria o *Conveyor 3 (Figura 5.17)* face ao *Conveyor 2 (Figura 5.16)* para os produtos do primeiro e terceiro tipo, pois se não fosse assim o *Conveyor 2 (Figura 5.16)* começaria a acumular produtos.

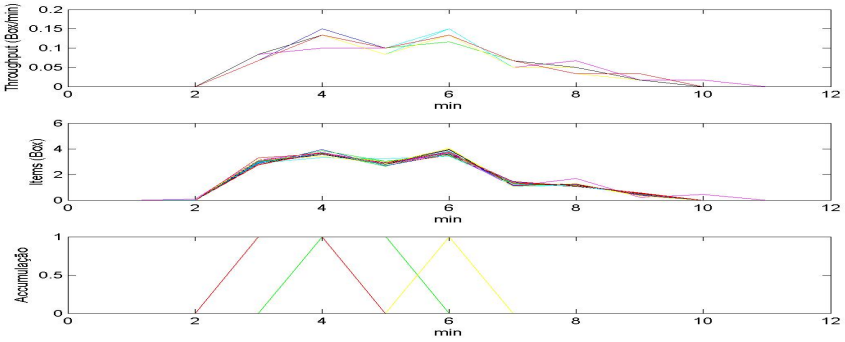

Figura 5.18: Resultados da Simulação do *Conveyor 7* no Caso Célula Nº 2

O *Conveyor 7 (Figura 5.18)* consiste num canal que une dois *Conveyors*, logo o seu estado ao longo da simulação reflete o efeito de recirculação, como se pode ver no pico ao minuto 6. Isto também é válido para o *Conveyor 4 (Figura 5.15)* mas no pico ao minuto 5.

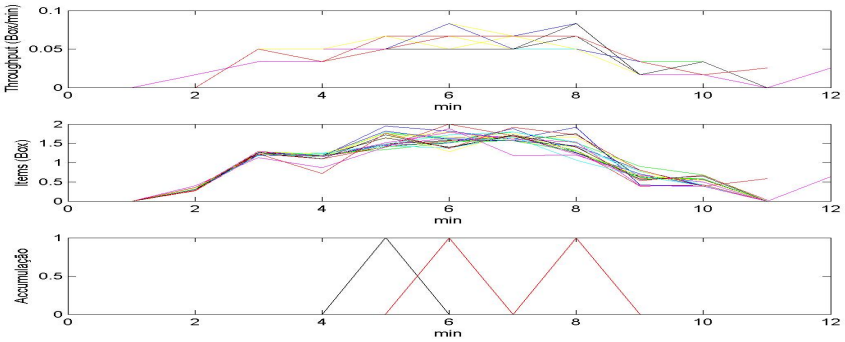

Figura 5.19: Resultados da Simulação do *Conveyor 5* no Caso Célula Nº 2

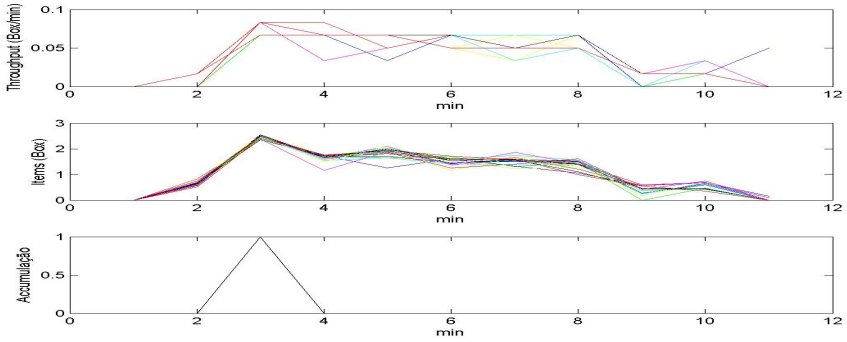

Figura 5.20: Resultados da Simulação do *Conveyor 6* no Caso Célula Nº 2

Relativamente ao *Conveyor 5 (Figura 5.19)* é o mais usado no inicio da simulação que o *Conveyor 6 (Figura 5.20)* pois os produtos são encaminhados devido à métrica aplicada pelo controlador (**Custo=10*Nº de Produto no*Conveyor*+Nº de *Docking Points*+Tamanho do *Conveyor***), em que no início da simulação o custo do *Conveyor 5 (Figura 5.19)* é mais baixo que o do *Conveyor 6 (Figura 5.20)* devido à sua dimensão.

Mas a carga entre os *Conveyor 5 (Figura 5.19)*, *6 (Figura 5.20)* é balanceada entre os dois, visto que no final da simulação o *Conveyor 5 (Figura 5.19)* comporta-se como um caminho alternativo, enquanto os produtos não chegam à penúltima *Skill* a ser executada.

De acordo com os gráficos apresentados anteriormente, é possível concluir que o comportamento de todos os agentes de *Conveyor* e do sistema de transporte foi sempre consistente ao longo de todas as simulações, e houve um balanceamento da carga de produtos no sistema.

5.3 Caso Célula Nº 3

Neste caso, o sistema a simular inclui um complexo sistema de *Conveyor's* e inúmeras estações, em que cada uma está habilitada a realizar uma distinta e única tarefa no sistema em geral. A direção do fluxo dos produtos são indicados pelas setas azuis, como são demonstradas através da Figura 5.21.

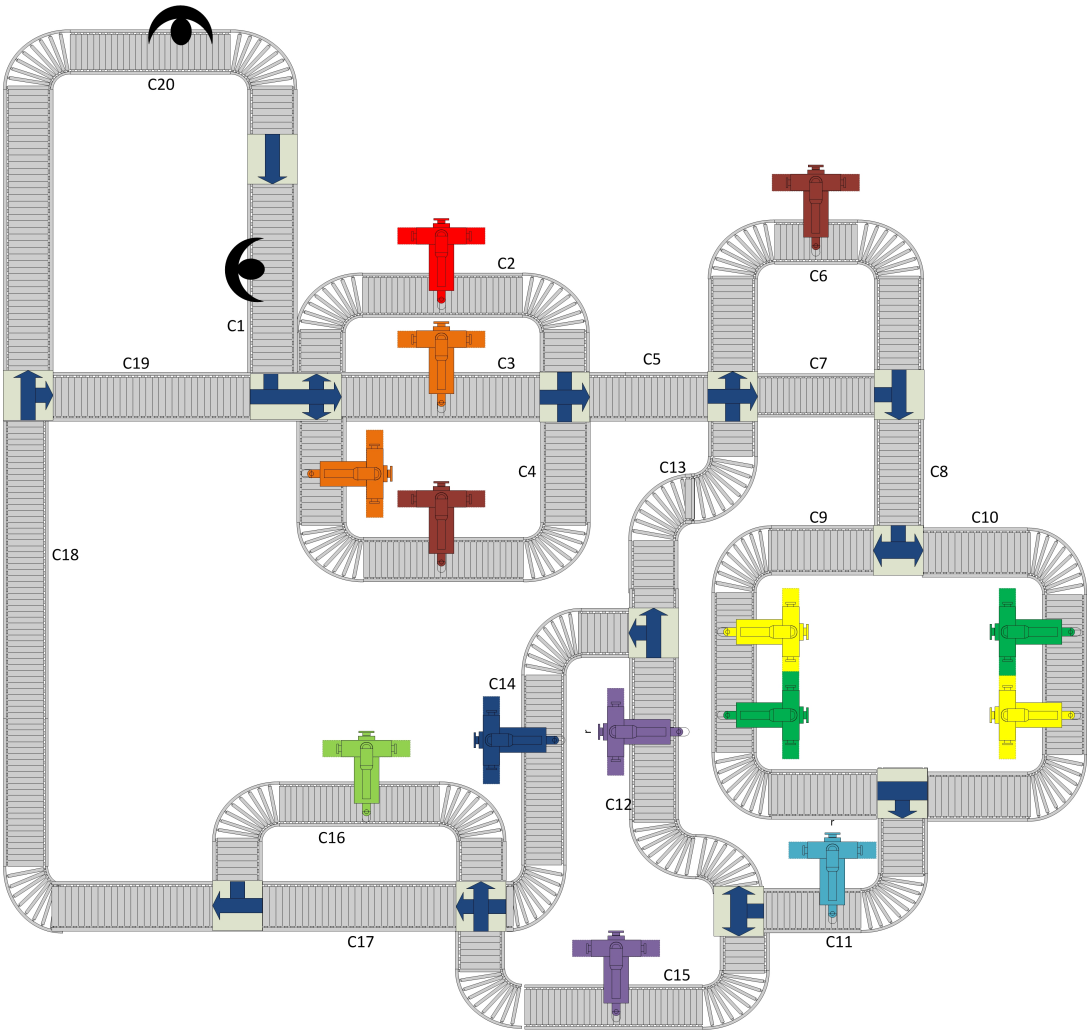

Figura 5.21: Caso Célula Nº 3

Através da Figura 5.21, verifica-se que o sistema possui inúmeros caminhos alternativos para os diferentes produtos projetados para este sistema e que existem duas estações que são responsáveis pela introdução e remoção dos produtos no sistema.

Esta célula foi simulada com 121 agentes no total, sendo que na Tabela 5.9 encontram-se enumerados os agentes de simulação e os agentes de transporte.

Tabela 5.9: Tipo e Número de Agentes no Caso Célula Nº 3

Tipo	Número
Conveyors	40
Interseções	24
Produtos	40
Recursos	14
Simulação	2
Visualização	1
Total	121

5.3.1 Caraterísticas

Cada Estação/Recurso consegue realizar *Skills* em apenas um produto de cada vez, enquanto cada *Conveyor* consegue manipular vários produtos, pois possuem tamanho diferentes.

De acordo com a Tabela 5.10 , é possível comprovar as características dos vários componentes, especificamente, os *Conveyors* que constituem a linha de montagem a simular. Sendo que neste caso também foi normalizado o tamanho dos produtos para 0.25 m.

Tabela 5.10: Descrição dos *Conveyors* do Caso Célula Nº 3

Conveyor (Id)	Tamanho (Palete)	Velocidade (m/s)
1	8	0.121
2	8	0.121
3	8	0.121
4	14	0.121
5	5	0.121
6	8	0.121
7	5	0.121
8	5	0.121
9	14	0.121
10	14	0.121
11	8	0.121
12	8	0.121
13	5	0.121
14	8	0.121
15	8	0.121
16	8	0.121
17	5	0.121
18	28	0.121
19	5	0.121
20	45	0.121

De acordo com a Tabela 5.11, é possível comprovar as características das estações, como localização no *Conveyor* a que se encontra conectado, as *Skills* que executa, etc. que constituem a linha de montagem a simular. Sendo possível, realocar as estações em qualquer localização no *Conveyor*.

Tabela 5.11: Descrição das Estações do Caso Célula Nº 3

Estação (Id)	***Skill***	***Conveyor (Id)***	**Localização no *Conveyor* (Palete)**
1	*Load* & *UnLoad*	1	2
2	*UnLoad* & *Load*	20	41
3	C	2	6
4	A	3	6
5	A	4	6
6	B	4	12
7	B	6	6
8	D	9	6
9	E	9	12
10	E	10	6
11	D	10	12
12	F	11	16
13	G	15	6
14	G	12	6
15	H	14	6
16	I	16	6

Para este caso também foram escolhidos três tipos de produtos a circularem na linha que são representados pela Tabela 5.12, sendo que estes são introduzidos no sistema, assim que for possível, no *Conveyor* a que as estações, que efetuam a introdução de produtos, encontram-se ligadas. Estes tipos de produtos são introduzidos no sistema com um *rácio* de 1:1:1.

Tabela 5.12: Descrição dos Produtos do Caso Célula Nº 3

Produto (Id)	***Skills***
1	ABCDEFGHI
2	ABCDEFGI
3	ABCDI

5.3.2 Resultados e Análise

Neste subcapítulo são apresentados os resultados das simulações efetuadas sobre o caso anterior. Sendo que foram efetuadas 21 simulações, que demoraram 40 minutos. Durante cada simulação a adição dos produtos foi progressiva e sem interrupções, como já foi referido anteriormente. O controlo do sistema foi efetuado através do sistema de transporte do projeto **IDEAS**.

De seguida, são apresentados os gráficos com os valores de *Throughput*, o número de produtos e se ocorre acumulação em cada *Conveyor*. E ainda, é apresentada uma análise dos resultados obtidos.

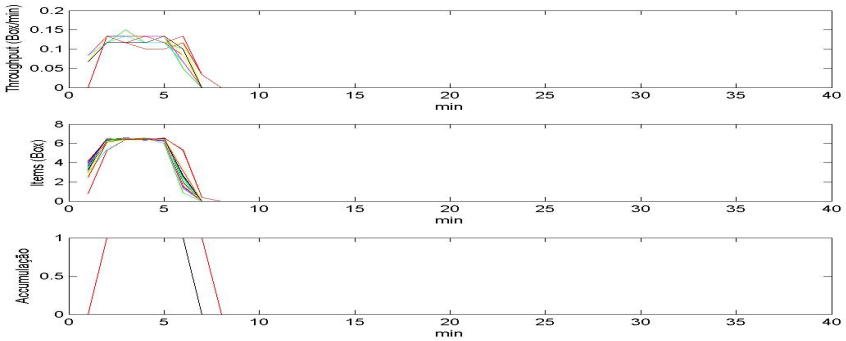

Figura 5.22: Resultados da Simulação do *Conveyor 1* no Caso Célula Nº 3

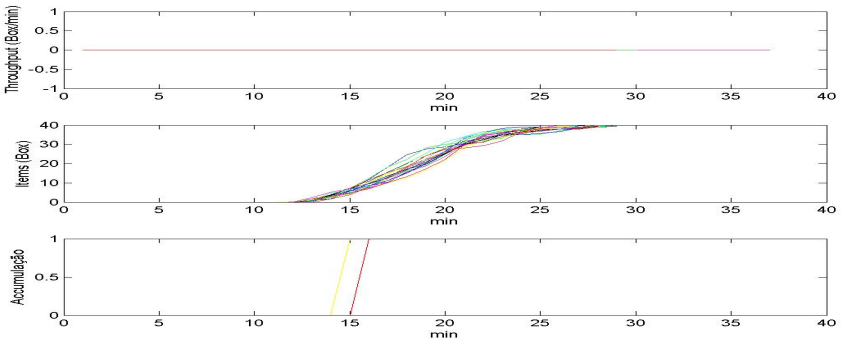

Figura 5.23: Resultados da Simulação do *Conveyor 20* no Caso Célula Nº 3

É possível verificar que o *Conveyor 1* (Figura 5.22) demorou cerca de 6 minutos a introduzir os 40 produtos no sistema. Sendo que este *Conveyor* durante toda a simulação nunca possui mais produtos, devido ao final de cada produto este ser encaminhado para a estação do *Conveyor 20* (Figura 5.23).

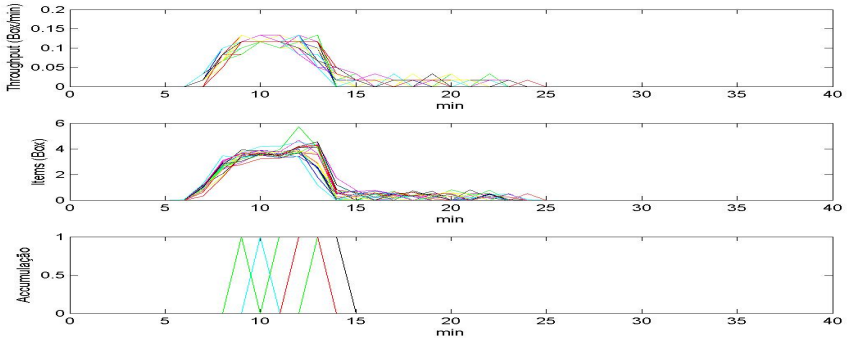

Figura 5.24: Resultados da Simulação do *Conveyor 2* no Caso Célula Nº 3

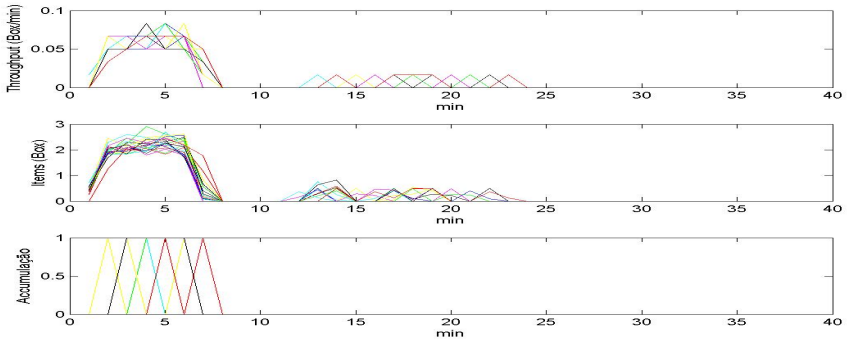

Figura 5.25: Resultados da Simulação do *Conveyor 3* no Caso Célula Nº 3

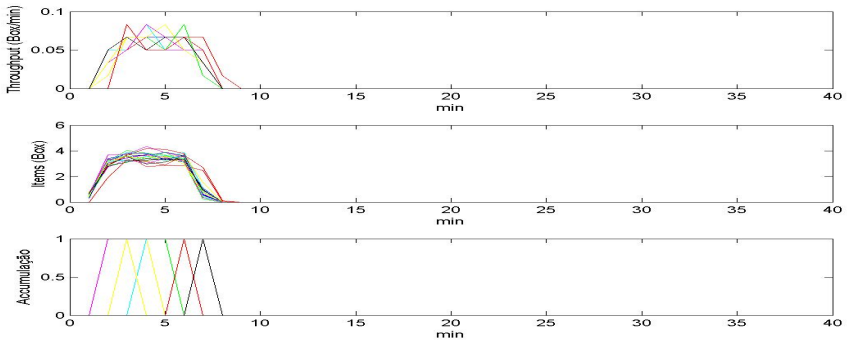

Figura 5.26: Resultados da Simulação do *Conveyor 4* no Caso Célula Nº 3

Nos *Conveyors 2 (Figura 5.24), 3 (Figura 5.25), 4 (Figura 5.26)*, é possível verificar a recirculação dos produtos no minuto 10, por causa do plano de *Skills* de cada produto e à disposição de cada estação no sistema.

O plano de *Skills* de cada produto possui uma *Skill* designada por **C** que só pode ser realizada no *Conveyor 2* (Figura 5.24), o que leva à recirculação de produtos.

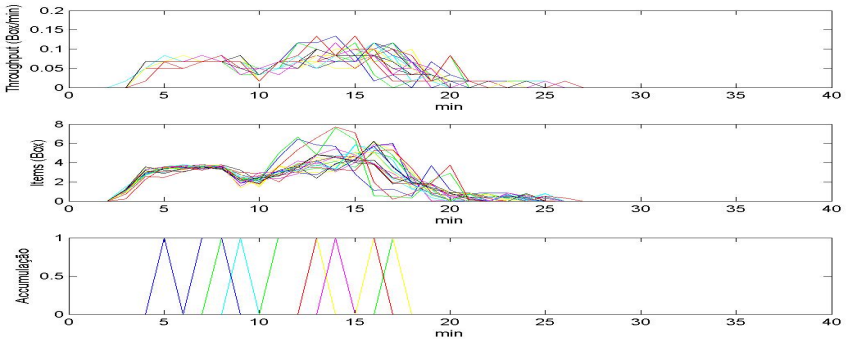

Figura 5.27: Resultados da Simulação do *Conveyor 9* no Caso Célula Nº 3

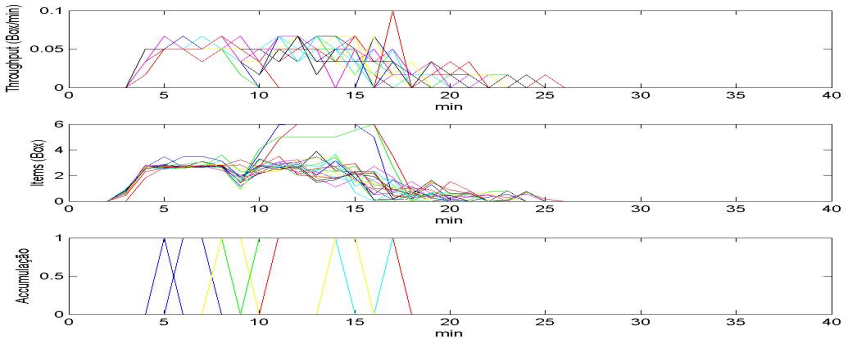

Figura 5.28: Resultados da Simulação do *Conveyor 10* no Caso Célula Nº 3

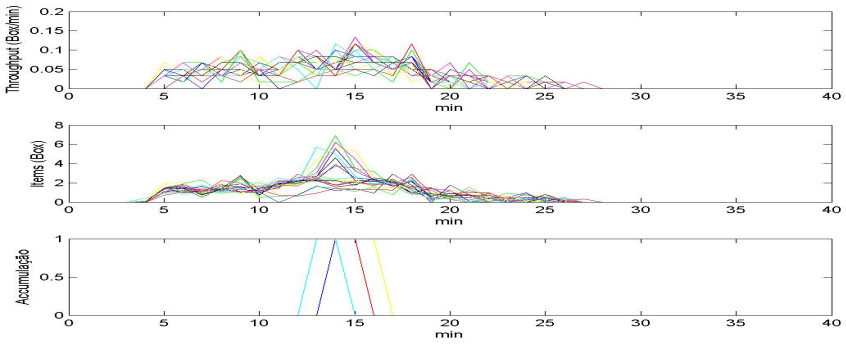

Figura 5.29: Resultados da Simulação do *Conveyor 12* no Caso Célula Nº 3

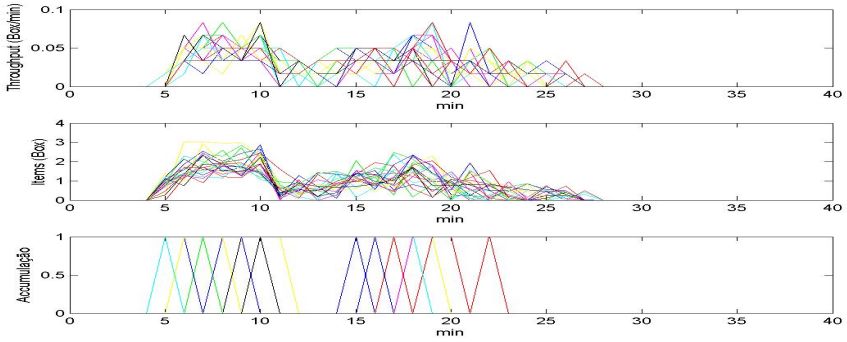

Figura 5.30: Resultados da Simulação do *Conveyor 14* no Caso Célula Nº 3

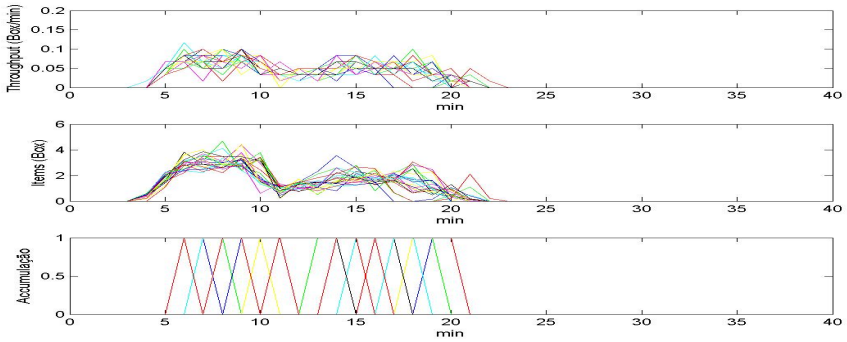

Figura 5.31: Resultados da Simulação do *Conveyor 15* no Caso Célula Nº 3

Este efeito também é causado pelas estações dos *Conveyor 9 (Figura 5.27), 10 (Figura 5.28), 12 (Figura 5.29), 15 (Figura 5.31), 14 (Figura 5.30)*, pois executam as *Skills*, **D** e **E**, **E** e **D**, **G**, **G**, e **H**, respetivamente. Nestes casos, teoricamente, o controlador escolheria o *Conveyor 9* (Figura 5.27) face ao *Conveyor 10* (Figura 5.28), pois no caso de escolher o último executava a *Skill* **D** e de seguida teria de efetuar uma nova circulação na linha para executar a *Skill* **E**. Isto também acontece nos casos da *Skill* **G** e **H**.

Devido à métrica usada no sistema de transporte para o transporte dos produtos (**Custo=10*Nº de Produto no*Conveyor*+Nº de *Docking Points*+Tamanho do *Conveyor***), o *Conveyor 4* (Figura 5.26) não sofre o efeito da recirculação, pois os *Conveyor 2 (Figura 5.24), 3 (Figura 5.25)* nunca acumulam no período de recirculação fazendo com que o custo destes dois seja mais baixo que o do *Conveyor 4* (Figura 5.26).

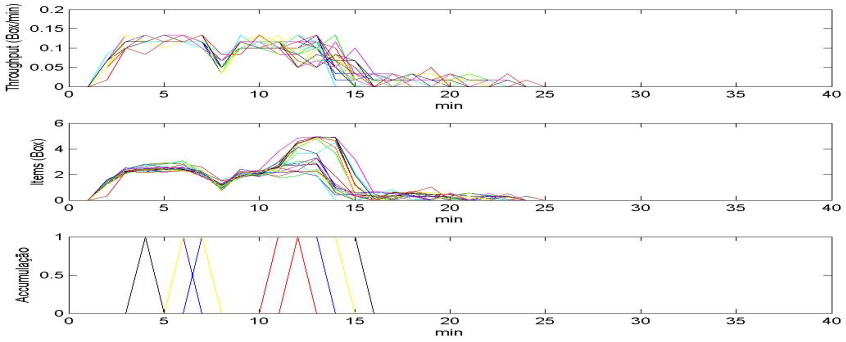

Figura 5.32: Resultados da Simulação do *Conveyor 5* no Caso Célula Nº 3

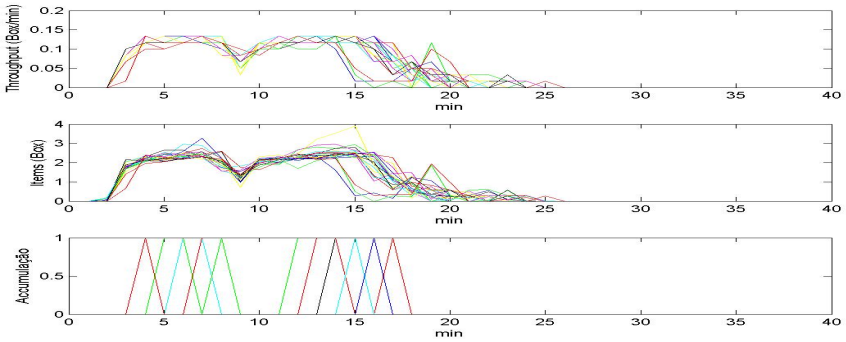

Figura 5.33: Resultados da Simulação do *Conveyor 8* no Caso Célula Nº 3

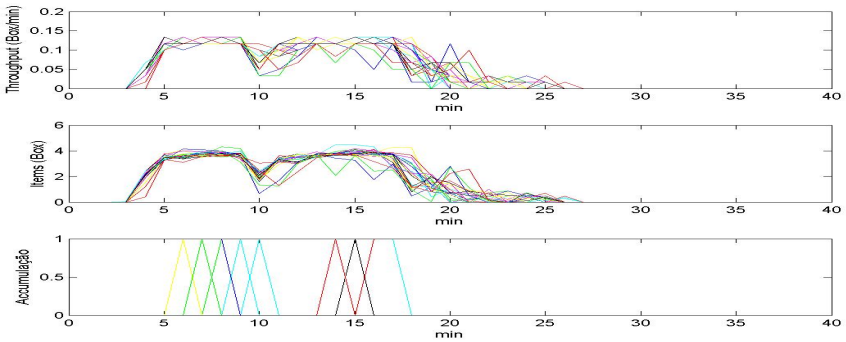

Figura 5.34: Resultados da Simulação do *Conveyor 11* no Caso Célula Nº 3

O *Conveyor 5* (Figura 5.32) consiste num canal que une três *Conveyors*, logo o seu estado ao longo da simulação reflete o efeito de recirculação. Tal como os *Conveyors 8 (Figura 5.33), 11 (Figura 5.34)*.

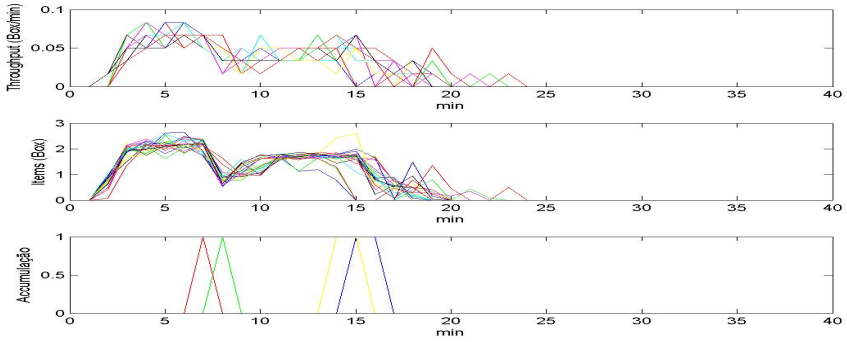

Figura 5.35: Resultados da Simulação do *Conveyor 6* no Caso Célula Nº 3

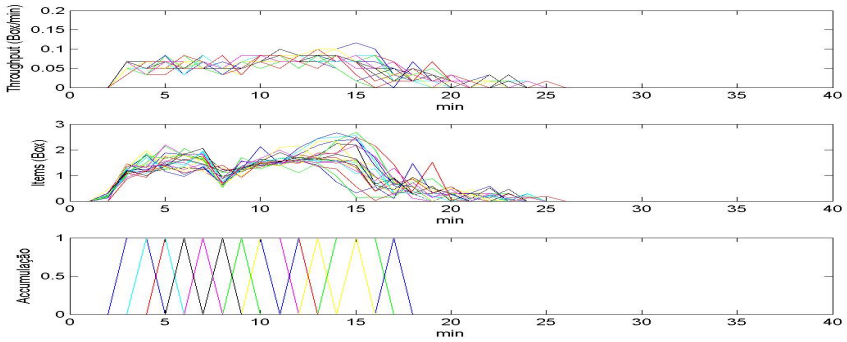

Figura 5.36: Resultados da Simulação do *Conveyor 7* no Caso Célula Nº 3

Relativamente ao *Conveyor 6 (Figura 5.35)*, *7 (Figura 5.36)*, o *Conveyor 6 (Figura 5.35)* é o mais usado no início da simulação que o *Conveyor 7 (Figura 5.36)* pois os produtos executam a *Skill* **A** no *Conveyor 3 (Figura 5.25)* e de seguida vão para o *Conveyor 6 (Figura 5.35)* para a *Skill* **B** em vez de ocorrer a recirculação, caso não siga este trajeto. Caso haja recirculação os dois *Conveyors* possuem comportamentos similares.

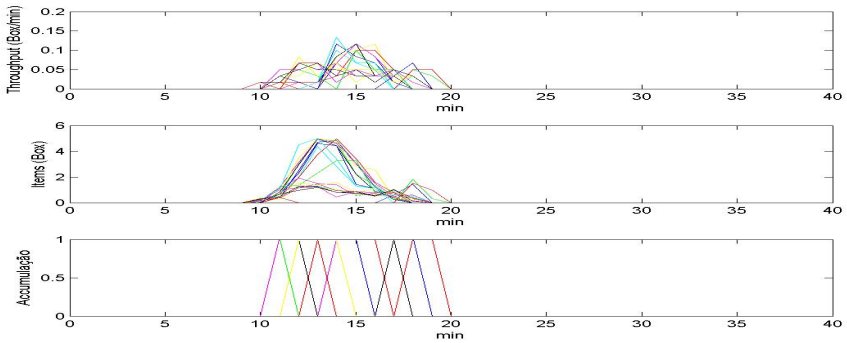

Figura 5.37: Resultados da Simulação do *Conveyor 13* no Caso Célula Nº 3

Os *Conveyor 9 (Figura 5.27)*, *10 (Figura 5.28)* possuem comportamentos muito similares, devido ao fato da rápida recirculação através do *Conveyor 13 (Figura 5.37)* que fica sempre ativo para os dois primeiros tipos de produto da Tabela 5.12. O que leva a um aumento de carga nos *Conveyor 6 (Figura 5.35)*, *7 (Figura 5.36)* no minuto 10.

O conjunto de *Conveyor 12 (Figura 5.29)*, *14 (Figura 5.30)*, *15 (Figura 5.31)* tem comportamentos similares em comparação com o par de *Conveyor 9 (Figura 5.27)*, *10 (Figura 5.28)*. Os *Conveyor 12 (Figura 5.29)*, *15 (Figura 5.31)* são muito parecidos tirando o pico inicial no *Conveyor 15 (Figura 5.31)* e o pico do meio no *Conveyor 12 (Figura 5.29)*. Sendo que o *Conveyor 15 (Figura 5.31)* é o mais usado para a recirculação onde todos os produtos executaram as *Skills* **A** e **B** e de seguida necessitam de executar a *Skill* **C**, sendo que no decorrer da simulação o controlador prefere ir pelo conjunto *Conveyor 12 (Figura 5.29)*, *14 (Figura 5.30)*.

O primeiro pico de atividade do *Conveyor 14 (Figura 5.30)* acontece devido à recirculação, pois nenhum dos produtos executa a *Skill* **H**. Depois da recirculação, existe um novo pico de atividade, por volta do minuto 17, o que indica que os produtos encontram-se na penúltima *Skill* a ser executada.

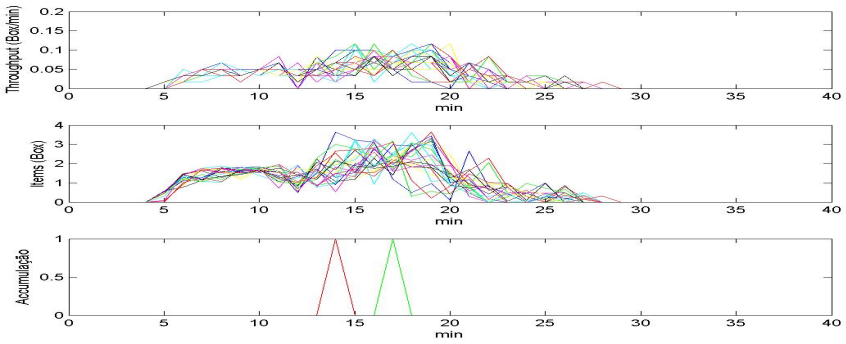

Figura 5.38: Resultados da Simulação do *Conveyor 16* no Caso Célula Nº 3

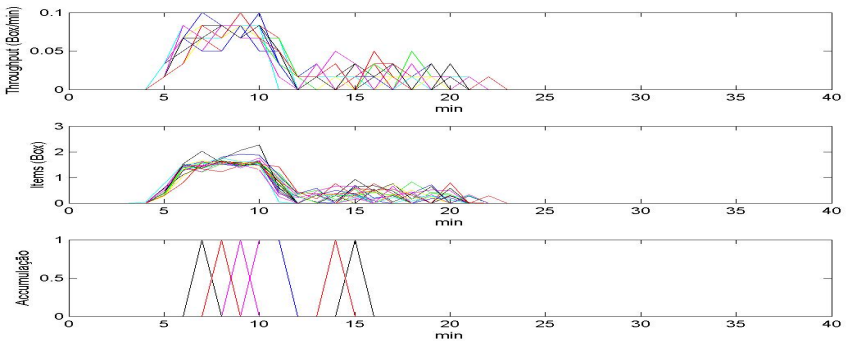

Figura 5.39: Resultados da Simulação do *Conveyor 17* no Caso Célula Nº 3

A carga entre os *Conveyor 17 (Figura 5.39), 16 (Figura 5.38)* é balanceada entre os dois, visto que no início da simulação estes comportarem-se como caminhos alternativos, enquanto os produtos não chegam à penúltima *Skill* a ser executada.

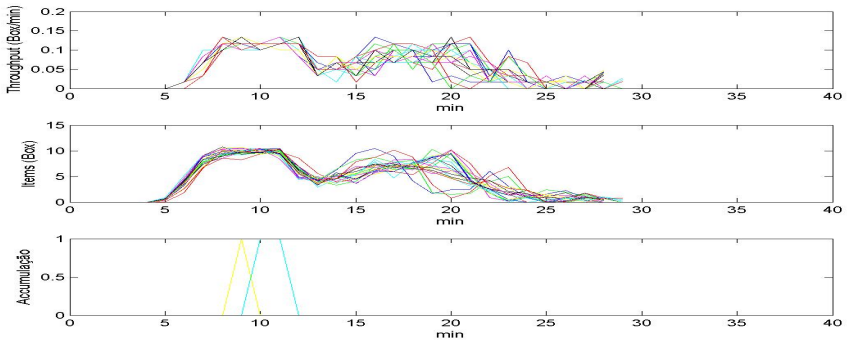

Figura 5.40: Resultados da Simulação do *Conveyor 18* no Caso Célula Nº 3

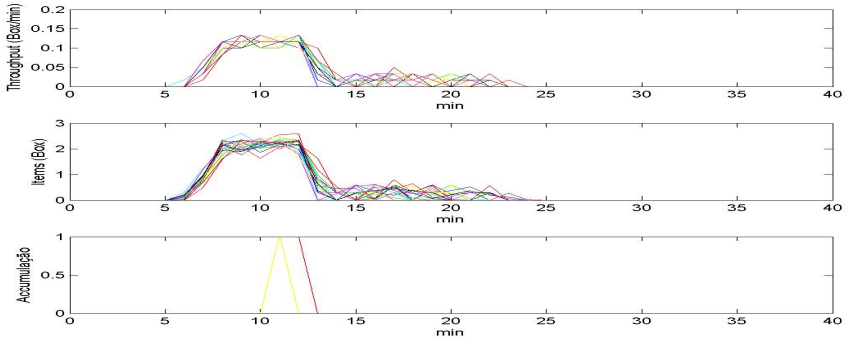

Figura 5.41: Resultados da Simulação do *Conveyor 19* no Caso Célula Nº 3

De acordo com os gráficos apresentados anteriormente, é possível concluir que o comportamento de todos os agentes de *Conveyor* e do sistema de transporte foi sempre consistente ao longo de todas as simulações, e houve um balanceamento da carga de produtos no sistema.

6

Conclusões e Trabalhos Futuros

6.1 Conclusões

Devido ao aumento da complexidade dos sistemas de manufatura, os paradigmas da Indústria que vão surgindo baseiam-se sempre em abordagens distribuídas. À medida que esses paradigmas vão surgindo, o uso de ferramentas de simulação aumenta drasticamente.

As ferramentas de simulação atuais possuem ainda, na sua maioria, uma abordagem centralizada, o que implica um elevado nível de processamento computacional num único controlador. Não permitem realizar uma simulação determinística e não possuem a capacidade de adicionar ou remover componentes durante a execução de uma simulação.

A tecnologia multiagente possui uma arquitetura distribuída e permite fazer a gestão de agentes que se encontram em vários controladores de forma a emular a noção de que cada entidade pode ser encontrada em qualquer plataforma. Por ser um sistema auto-organizável, é possível adicionar ou remover agentes durante a execução do sistema. Por estes motivos, o desenvolvimento deste trabalho foi baseado em agentes.

A ferramenta desenvolvida demonstrou a possibilidade de distribuição de agentes por entre vários controladores de modo a baixar o processamento computacional. A ferramenta ainda permitiu a interligação do sistema de transporte do **IDEAS** com o objetivo de validar e depurar a sua arquitetura. Provou ainda ser consistente nos resultados das simulações nestes casos de interligação, como se pode ver no Capítulo 5.

6.2 Trabalhos Futuros

Um dos objetivos principais a ser desenvolvido no futuro consiste na modelação e diversificação dos tipos de elementos de um sistema de manufatura. E ainda no aperfeiçoamento da *Performance* e análise da distribuição dos agentes do sistema a simular.

Finalmente, seria também interessante, desenvolver um algoritmo que permitisse construir o *layout* gráfico da linha de montagem que o sistema de transporte pretenda simular de uma forma automática, baseando-se nas características de cada elemento e das suas relações.

Bibliografia

[1] N. Ruiz, A. Giret, and V. Botti, "Towards an agent-based simulation tool for manufacturing systems," in *Emerging Technologies and Factory Automation, 2006. ETFA'06. IEEE Conference on*, pp. 797–804, IEEE, 2006.

[2] IRDAC, "Opinion on r&d needs in the field of mechatronics," *Industry R&D Advisory Committee of the Comm. of the EC, Brussels, Belgium*, 1986.

[3] J. Van Amerongen, "The role of control in mechatronics," *Engineering Science and Education Journal*, vol. 9, no. 3, pp. 105–112, 2000.

[4] A. Law, W. Kelton, and W. Kelton, *Simulation modeling and analysis*, vol. 2. McGraw-Hill New York, 2000.

[5] C. M. Moreira, *Estratégias de reposição de estoques em supermercados: avaliação por meio de simulação*. PhD thesis, Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Engenharia de Produção., 2001.

[6] A. Drogoul and J. Ferber, "Multi-agent simulation as a tool for modeling societies: Application to social differentiation in ant colonies," in *Artificial Social Systems*, pp. 2–23, Springer, 1994.

[7] L. F. Jacintho, A. F. Batista, T. L. Ruas, M. G. Marietto, and F. A. Silva, "An agent-based model for the spread of the dengue fever: a swarm platform simulation approach," in *Proceedings of the 2010 Spring Simulation Multiconference*, p. 2, Society for Computer Simulation International, 2010.

[8] J. Ferber, "Reactive distributed artificial intelligence: Principles and applications," *Foundations of distributed artificial intelligence*, pp. 287–314, 1996.

[9] N. Gilbert and K. Troitzsch, *Simulation for the social scientist*. Open university press, 2005.

[10] P. Klingstam and P. Gullander, "Overview of simulation tools for computer-aided production engineering," *Computers in Industry*, vol. 38, no. 2, pp. 173–186, 1999.

[11] H. R. Hoeger and D. W. Jones, "Integrating concurrent and conservative distributed discrete-event simulators," *Simulation*, vol. 67, no. 5, pp. 303–314, 1996.

[12] D. F. Geuder, "Object oriented modeling with simple++," in *Simulation Conference Proceedings, 1995. Winter*, pp. 534–540, IEEE, 1995.

[13] A. M. Law and M. G. McComas, "Simulation of manufacturing systems," in *Proceedings of the 30th conference on Winter simulation*, pp. 49–52, IEEE Computer Society Press, 1998.

[14] D. Krahl, "Extendsim advanced techology: discrete rate simulation," in *Simulation Conference (WSC), Proceedings of the 2009 Winter*, pp. 333–338, IEEE, 2009.

[15] K. Ueda, "A concept for bionic manufacturing systems based on dna-type information," in *Proceedings of the IFIP TC5/WG5. 3 Eight International PROLAMAT Conference on Human Aspects in Computer Integrated Manufacturing*, pp. 853–863, North-Holland Publishing Co., 1992.

[16] R. F. Babiceanu and F. F. Chen, "Development and applications of holonic manufacturing systems: a survey," *Journal of Intelligent Manufacturing*, vol. 17, no. 1, pp. 111–131, 2006.

[17] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy, and H. Van Brussel, "Reconfigurable manufacturing systems," *CIRP Annals-Manufacturing Technology*, vol. 48, no. 2, pp. 527–540, 1999.

[18] M. Onori, H. Alsterman, and J. Barata, "An architecture development approach for evolvable assembly systems," in *Assembly and Task Planning: From Nano to Macro Assembly and Manufacturing, 2005.(ISATP 2005). The 6th IEEE International Symposium on*, pp. 19–24, IEEE, 2005.

[19] L. Ribeiro, J. Barata, G. Cândido, and M. Onori, "Evolvable production systems: an integrated view on recent developments," in *Proceedings of the 6th CIRP-Sponsored International Conference on Digital Enterprise Technology*, pp. 841–854, Springer, 2010.

[20] M. Wooldridge, *An introduction to multiagent systems*. Wiley, 2002.

[21] M. Wooldridge and N. Jennings, "Agent theories, architectures, and languages: a survey," *Intelligent agents*, pp. 1–39, 1995.

[22] Y. Deng, J. Cao, and X. Wang, "A distributed multi-agent simulation platform," in *Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on*, pp. 64–71, IEEE, 2012.

[23] P. Vrba, "Mast: manufacturing agent simulation tool," in *Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA'03. IEEE Conference*, vol. 1, pp. 282–287, IEEE, 2003.

[24] G. Nadoli and J. Biegel, "Decision making agents in manufacturing systems simulation: examples," in *Systems, Man, and Cybernetics, 1991.'Decision Aiding for Complex Systems, Conference Proceedings., 1991 IEEE International Conference on*, pp. 411–416, IEEE, 1991.

[25] A. S. Tanenbaum, "Distributed systems principles and paradigms, pearson education, inc," 2008.

[26] R. Engelmore and T. Morgan, "Blackboard systems. 1988."

[27] L. Ribeiro, J. Barata, M. Onori, C. Hanisch, J. Hoos, and R. Rosa, "Self-organization in automation - the ideas pre-demonstrator," in *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*, pp. 2752 –2757, nov. 2011.

[28] "Ideas-consortium, instantly deployable evolvable assembly systems," in *http://www.ideas-project.eu/*, 2010.

[29] Ribeiro and J. Barata, "Iade - ideas agent development environment: Lessons learned and research directions," *CIRP Conference On Assembly Technologies And Systems (CATS 12), Ann Arbor*, 2012.

7

Anexo 1 - Esquema da Habilidade

<xsd:complexType name="tMySkill»
<xsd:sequence>
<xsd:element name="myName"type="xsd:string"/>
<xsd:element name="mySpeed"type="xsd:double"/>
<xsd:element name="myStation"type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>

8

Anexo 2 - Esquema do Recurso

<xsd:complexType name="tMyResources»
<xsd:sequence>
<xsd:element name="myName"type="xsd:string"/>
<xsd:element name="myId"type="xsd:int"/>
<xsd:element name="mySkillListInfo"minOccurs="0"maxOccurs="unbounded"
type="tns:tMySkill"/>
<xsd:element name="myStation"type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>

9

Anexo 3 - Esquema da Estação

<xsd:complexType name="tMyStation»
<xsd:sequence>
<xsd:element name="myId"type="xsd:int"/>
<xsd:element name="myResourceListInfo"maxOccurs="unbounded"
type="tns:tMyResources»</xsd:element>
<xsd:element name="myCyclicUnit"type="xsd:int"/>
<xsd:element name="myLocation"type="xsd:double"/>
<xsd:element name="myEntryStopper"type="xsd:boolean"/>
<xsd:element name="myExitStopper"type="xsd:boolean"/>
<xsd:element name="myResourceIndex"type="xsd:int"/>
<xsd:element name="mySkillIndex"type="xsd:int"/>
</xsd:sequence>
</xsd:complexType>

10

Anexo 4 - Esquema do Produto

<xsd:complexType name="tMyBox»
<xsd:sequence>
<xsd:element name="myId"type="xsd:int"/>
<xsd:element name="myLocation"type="xsd:double"/>
<xsd:element name="myLength"type="xsd:double"/>
<xsd:element name="mySkillListInfo"maxOccurs="unbounded"type="tns:tMySkill"/>
<xsd:element name="myGlobalId"type="xsd:int"/>
<xsd:element name="myStationId"type="xsd:int"/>
<xsd:element name="myStationCase"type="xsd:int"/>
<xsd:element name="myEntryPointId"type="xsd:int"/>
</xsd:sequence>
</xsd:complexType>

11

Anexo 5 - Esquema do Sistema

<xsd:complexType name="tMyFactory»
<xsd:sequence>
<xsd:element name="myEntryListInfo"type="tns:tMyEntry"minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="myConveyorListInfo"maxOccurs="unbounded"minOccurs="0"
type="tns:tMyConveyor"/>
<xsd:element name="myExitListInfo"maxOccurs="unbounded"minOccurs="0"
type="tns:tMyExit"/>
<xsd:element name="myJunctionListInfo"maxOccurs="unbounded"minOccurs="0"
type="tns:tMyJunction"/>
<xsd:element name="myComponentConnectionsListInfo"maxOccurs="unbounded"
minOccurs="0"type="tns:tMyComponentConnections"/>
</xsd:sequence>
</xsd:complexType>

12

Anexo 6 - Esquema do Ponto de Entrada

```
<xsd:complexType name="tMyEntry»
<xsd:sequence>
<xsd:element name="myId"type="xsd:int"/>
<xsd:element name="mySpeed"type="xsd:double"/>
<xsd:element name="myBoxListInfo"type="tns:tMyBox"maxOccurs="unbounded"/>
<xsd:element name="myNextBox"type="xsd:int"/>
<xsd:element name="myCyclicUnit"type="xsd:int"/>
<xsd:element name="myCyclic"type="xsd:boolean"/>
<xsd:element name="myKill"type="xsd:boolean"/>
<xsd:element name="myCyclicValue"type="xsd:double"/>
<xsd:element name="myGlobalIdBox"type="xsd:int"/>
</xsd:sequence>
</xsd:complexType>
```

13

Anexo 7 - Esquema do Ponto de Saída

<xsd:complexType name="tMyExit»
<xsd:sequence>
<xsd:element name="myId"type="xsd:int"/>
<xsd:element name="myBoxListInfo"type="tns:tMyBox"maxOccurs="unbounded"
minOccurs="0»</xsd:element>
<xsd:element name="myCyclicUnit"type="xsd:int"/>
<xsd:element name="myKill"type="xsd:boolean"/>
</xsd:sequence>
</xsd:complexType>

14

Anexo 8 - Esquema do Conveyor

<xsd:complexType name="tMyConveyor»
<xsd:sequence>
<xsd:element name="myId"type="xsd:int"/>
<xsd:element name="myLength"type="xsd:double"/>
<xsd:element name="mySpeed"type="xsd:double"/>
<xsd:element name="myBoxListInfo"type="tns:tMyBox"minOccurs="0"
maxOccurs="unbounded»</xsd:element>
<xsd:element name="myCyclicUnit"type="xsd:int"/>
<xsd:element name="myHorinzontal"type="xsd:boolean"/>
<xsd:element name="myDirection"type="xsd:boolean"/>
<xsd:element name="myStations"type="tns:tMyStationList"/>
<xsd:element name="myCyclicValue"type="xsd:double"/>
<xsd:element name="myKill"type="xsd:boolean"/>
</xsd:sequence>
</xsd:complexType>

15

Anexo 9 - Esquema da Intersecção

<xsd:complexType name="tMyJunction»
<xsd:sequence>
<xsd:element name="myId"type="xsd:int"/>
<xsd:element name="myTime"type="xsd:double"/>
<xsd:element name="myBoxListInfo"type="tns:tMyBox"/>
<xsd:element name="myCyclicUnit"type="xsd:int"/>
<xsd:element name="myType"type="xsd:string"/>
<xsd:element name="myRatio"type="xsd:boolean"/>
<xsd:element name="myRatios"minOccurs="0"maxOccurs="3"type="tns:tMyBoxRatios"/>
<xsd:element name="myFilters"minOccurs="0"maxOccurs="unbounded"
type="tns:tMyBoxFilters"/>
<xsd:element name="myCyclicValue"type="xsd:double"/>
<xsd:element name="myKill"type="xsd:boolean"/>
</xsd:sequence>
</xsd:complexType>