INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Modelação e Simulação Baseada em Agentes para Cadeias de Abastecimento

José João Cruz Dias

Dissertação apresentada à

Escola Superior de Tecnologia e Gestão Instituto Politécnico de Bragança

para obtenção do grau de Mestre em

Engenharia Industrial – Ramo Engenharia Eletrotécnica

Trabalho realizado sob a orientação de

Prof. Dr. Paulo Leitão

Prof. Dr. José Barbosa

2015/2016

Agradecimentos

Um especial obrigado aos Professores Paulo Leitão e José Barbosa pela orientação, conhecimento e disponibilidade ao longo do ano.

A todos os amigos, pelo apoio e motivação, sem vocês tudo teria sido bem mais difícil.

Agradeço à minha família, a paciência e encorajamento que em muito contribuíram para que conseguisse finalizar mais uma etapa da minha vida.

Resumo

Nos últimos anos, considerando o vasto campo de aplicações, a modelação baseada em agentes tem mostrado o seu valor como uma ferramenta essencial para simular e analisar fenómenos sociais e económicos. Estas ferramentas permitem um fácil e rápido desenvolvimento de protótipos e provas de conceito, através da simulação de diferentes cenários.

Numa cadeia de abastecimento o stock das empresas é muito flexível, porque a quantidade de produtos encomendados variam diariamente. Sendo assim, os principais desafios são: gestão da produção, estratégias de transporte e política de stocks.

Neste trabalho é apresentado o desenvolvimento de um modelo de uma cadeia de abastecimento utilizando uma ferramenta de modelação baseada em agentes, NetLogo. A utilização deste modelo baseado em agentes para a cadeia de abastecimento, permite implementar mecanismos automáticos e dinâmicos de balanceamento do fluxo de fornecimento (relacionado com a gestão dos níveis de stocks dos vários intervenientes na cadeia de abastecimento), e simular diferentes estratégias para diferentes cenários.

Foi efetuado um estudo sobre a ferramenta NetLogo, que incidiu principalmente na análise das simulações da própria biblioteca e nos recursos disponibilizados na página web, nomeadamente guias de programação, dicionário NetLogo e tutoriais. Durante este estudo, foram também desenvolvidas várias simulações com o objetivo de explorar as várias funcionalidades disponíveis na ferramenta.

O modelo baseado em agentes para a simulação de uma cadeia de abastecimento é composto por fornecedores, fabricantes, distribuidores e clientes, através dos quais a matéria-prima é adquirida, transformada e entregue ao cliente. Estas entidades distribuídas são agentes inteligentes, autónomos e cooperativos, cada um responsável por uma ou mais atividades na cadeia de abastecimento. A interação entre os vários agentes permite a emergência do sistema global da cadeia de fornecimento.

Palavras-chave: Agentes, Sistemas Multiagentes, NetLogo, Cadeia de Abastecimento.

Abstract

In recent years, considering the wide range of applications, the agent-based modelling has shown its value as an essential tool to simulate and analyze social and economic phenomena. These tools allow an easy and rapid development of prototypes and proofs of concept through the simulation of different scenarios.

In a supply chain the stock of companies is very flexible, because the quantity of products ordered varies daily. Therefore, the main challenges are: production management, transport strategies and stock policy.

In this paper we present the development of a supply chain model using an agent-based modeling tool, NetLogo. The use of this agents based on model to the supply chain, allows for the implementation automatic and dynamic mechanisms of supply flow balancing (related to the management of the levels of stocks of the various players in the supply chain), and simulate different strategies for different scenarios

It was made a study about the NetLogo tool, which focused mainly on the analysis of its libraries, simulations and resources available on web page such as: program guides, NetLogo dictionary and tutorials. During this study, several simulations were also developed in order to explore the various features available in the tool.

The agent-based model for the simulation of a supply chain is composed of suppliers, manufacturers, distributors and customers, through which the raw materials are acquired, transformed and delivered to the client. These entities are intelligent agents, being each one responsible for one or more activities in the supply chain, by interacting with other agents and perform its tasks.

Keywords: Agent, Multiagent Systems, NetLogo, Supply Chain.

Conteúdo

Agradecimentos ..... **iii**

Resumo ..... **v**

Abstract ..... **vi**

Lista de Tabelas ..... **ix**

Lista de Figuras ..... **xi**

Lista de Abreviações ..... **xiii**

1 Introdução **1**

1.1 Motivação e Objetivos 2

1.2 Organização do trabalho 2

2 Sistemas Multiagentes e Simulação Baseada em Agentes **5**

2.1 Agentes 5

2.2 Sistemas Multiagentes 7

2.3 Simulação 9

2.3.1 Simulação de Eventos Discretos 10

2.3.2 Simulação Baseada em Agentes 10

2.4 Plataformas de Simulação Baseadas em Agentes 11

2.4.1 Swarm 11

2.4.2 Repast 12

2.4.3 NetLogo 12

3 Caso de Estudo **15**

3.1 Conceito 15

3.2 Caso de estudo 16

3.3 Trabalhos existentes 18

4 Desenvolvimento do Modelo de Simulação **21**

4.1 Desenvolvimento do Modelo Baseado em Agentes 21

4.1.1 Fornecedor 23

4.1.2 Fabricante 24

4.1.3 Distribuidor 25

4.1.4 Cliente 26

4.2 Ambiente de Simulação do Modelo 27

4.2.1 Introdução de Dados no Modelo 28
4.2.2 Setup do Modelo 28
4.2.3 Execução da Simulação 29
4.2.4 Apresentação dos Resultados 32

5 Análise dos resultados experimentais 33

6 Conclusão 39

Referências bibliográficas 41

A Código Fonte 1

Lista de Tabelas

Tabela 1- Agentes e respetivas variáveis 22
Tabela 2 - Dados de entrada do Caso I. 33
Tabela 3 - Dados de Saída do Caso I. 34
Tabela 4 - Dados de entrada do Caso II. 35
Tabela 5 - Dados de Saída do Caso II. 37

Lista de Figuras

Figura 1 – Esquema típico de um agente. [Reis,2003]..... 6
Figura 2 - Sistemas multiagente.[Reis,2003] 7
Figura 3 – Exemplo de modelo da plataforma Swarm. 11
Figura 4- Exemplo de simulação Repast. 12
Figura 5- Interface principal do NetLogo. 13
Figura 6- Biblioteca de modelos no NetLogo. 13
Figura 7 - Exemplo de programação no NetLogo. 14
Figura 8- Cadeia de abastecimento típica. 15
Figura 9- Visão geral da cadeia de abastecimento. 16
Figura 10 - Cadeia de abastecimento complexa. 17
Figura 11 – Relação entre agentes. 21
Figura 12- Interface principal do modelo criado no ambiente NetLogo. 27
Figura 13 – Diagrama de blocos do modelo. 27
Figura 14 - Fluxograma da simulação. 30
Figura 15 – Exemplo de um histograma obtido numa simulação. 32
Figura 16 - Resultados gráficos do Caso I. 34
Figura 17 - Rutura do stock nos distribuidores. 35
Figura 18- Resultados gráficos do Caso II. 36

Lista de Abreviações

IA	Inteligência Artificial
IAD	Inteligência Artificial Distribuída
MBA	Modelação Baseada em Agentes
SBA	Simulação Baseada em Agentes
SED	Simulação de Eventos Discretos
SMA	Sistemas Multiagente

Capítulo 1

1 Introdução

O trabalho realizado está relacionado com o estudo de técnicas de modelação e simulação baseadas em agentes, aplicadas ao estudo de cadeias de abastecimento num sistema logístico.

Uma cadeia de abastecimento está organizada em quatro áreas principais. Primeiro, temos os fornecedores, responsáveis pela entrega da matéria-prima nas fábricas, de seguida, os produtores, indústrias transformadoras que através da matéria-prima produzem bens para o consumidor final, e por último, os distribuidores, responsáveis pelo escoamento de bens produzidos nas fábricas e por fazer chegar esses bens ao cliente [Simchi-Levi *et al*., 2003]. De maneira a obter uma gestão eficiente de uma cadeia de abastecimento, é necessário que estas áreas funcionem de forma coordenada, com o objetivo de encontrar boas soluções para os problemas inerentes na cadeia de logística [Fox et al.,1993].

Para o estudo da gestão de uma cadeia de abastecimento, ao longo dos anos foram desenvolvidos vários modelos, divididos em três categorias: estratégico, tático e operacional. A nível estratégico, o objetivo deste tipo de modelos consiste em determinar o melhor custo para localização das instalações, estimar o fluxo de mercadorias, atribuir clientes aos distribuidores e planear o envio de mercadorias. Já o principal objetivo dos modelos a nível tático é coordenar a produção e distribuição, definir níveis de stock para cada produto, entrega de produtos dentro dos prazos e avaliar o nível de serviço. A nível operacional é aplicado à colocação dos trabalhadores, rotas e escalonamento de veículos [Fox et al.,1993].

A grande diversidade de problemas encontrados numa cadeia de logística torna-a numa rede complexa. A maneira que se encontrou para estudar e compreender o comportamento dinâmico foi recorrer à simulação baseada em agentes [Cavalieri *et al*.,2003], que utilizam o paradigma de sistemas multiagentes (SMA).

Os SMA provêm, da área da Inteligência Artificial Distribuída. Um SMA compreende um conjunto de agentes, cada um dos quais é autónomo e cooperativo, que interagem para atingir os seus objetivos de forma distribuída e descentralizada. Um agente pode ser definido como uma entidade virtual, está inserido num determinado ambiente, pode interagir de forma autónoma através de sensores e atuadores, e pode comunicar com outros agentes para partilhar informação e estabelecer mecanismos de cooperação e colaboração para atingir os seus objetivos. Assim sendo, os agentes tem um comportamento autónomo como consequência das suas observações, das suas interações com outros agentes e do conhecimento que adquiriu.

1.1 Motivação e Objetivos

O principal objetivo deste trabalho foi estudar a dinâmica de cadeias de abastecimento utilizando ferramentas de modelação baseada em agentes, que permitem simular e validar problemas complexos, implementando diversas soluções estratégicas para o balanceamento do fluxo na cadeia de abastecimento. Nesta simulação, os agentes tem como tarefa manter os níveis de stock dos materiais equilibrados, tendo para isso que adaptar o seu comportamento sempre que exista compra/venda de produtos, podendo optar sempre pelo melhor preço ou pelo menor tempo de entrega.

Após estudo de várias plataformas de modelação e simulação baseadas em agentes, foi escolhida a plataforma NetLogo para desenvolver o modelo de agentes para a cadeia de abastecimento, que posteriormente será simulada em diversos cenários. A grande vantagem desta plataforma reside na sua capacidade de modelar e simular um processo de uma forma fácil e compreensível, além da interface simples e amigável para a configuração de parâmetros e visualização dos resultados.

1.2 Organização do trabalho

A presente relatório encontra-se estruturado em seis capítulos, sendo o primeiro composto pela introdução do trabalho.

No segundo capítulo é apresentado o conceito de agente, sendo abordadas as características que um agente pode possuir, tipos de agentes e diferentes aplicações. Ainda

neste capítulo, é referido as motivações para o uso de SMA, bem como a relevância dos modelos baseados em agentes. No final deste capítulo, foram descritos algumas plataformas de modelação e simulação baseadas em agentes, entre os quais o Netlogo, utilizado neste trabalho para a modelação baseada em agentes de uma cadeia de abastecimento.

No terceiro capítulo é descrito o caso de estudo utilizado neste trabalho, nomeadamente conceito de cadeia de abastecimento e o processo logístico que será modelado. Também neste capítulo, são apresentados alguns trabalhos e simulações nesta área desenvolvidos por outros autores.

No quarto capítulo é apresentado o modelo baseado em agentes para a cadeia de abastecimento desenvolvido utilizando a ferramenta de simulação NetLogo. Nomeadamente é efetuada uma descrição de todos os tipos de agentes desenvolvidos, das variáveis presentes em cada um, e por fim é explicado o funcionamento do modelo.

No quinto capítulo são apresentados os resultados experimentais resultantes da simulação desenvolvida, e discutidos os resultados obtidos.

No sexto capítulo são apresentadas as conclusões do trabalho e são propostos alguns trabalhos futuros.

Capítulo 2

2 Sistemas Multiagentes e Simulação Baseada em Agentes

Neste capítulo é apresentado o conceito de agente e as motivações para o uso de SMA. Também neste capítulo, é descrito dois tipos de simulação, simulação de eventos discretos e simulação baseada em agentes. Por fim, são apresentadas algumas plataformas de simulação baseada em agentes, entre os quais o NetLogo, utilizado neste trabalho para o desenvolvimento do MBA de uma cadeia de abastecimento.

2.1 Agentes

O termo “agentes” tem sido aplicado indistintamente tanto nas áreas de inteligência artificial, como nas de computação. Não há propriamente um conceito único que o caracteriza depende muito do ponto de vista do autor. Posto isto, conforme ilustrado Figura 1, um agente é uma entidade que interage de forma autónoma num determinado ambiente através de sensores e atuadores e, que procura cumprir os objetivos para o qual foi projetado [Wooldridge, 2002]. Por fim, a palavra agente apresenta as seguintes definições:

- É uma entidade **real** ou **virtual**;
- Encontra-se inserido num **ambiente**;
- É capaz de **perceber** o seu ambiente;
- É capaz de **agir** no ambiente;
- É capaz de **comunicar** com outros agentes;
- Tem um comportamento **autónomo**.

Pela definição de Reis [Reis, 2002b], um agente é "*um sistema computacional, situado num dado ambiente, que tem a perceção desse ambiente através de sensores, tem capacidade de decisão, age de forma autónoma nesse ambiente através de atuadores, e possui capacidades de comunicação de alto-nível com outros agentes e/ou humanos, de forma a desempenhar uma dada função para a qual foi projetado.*"

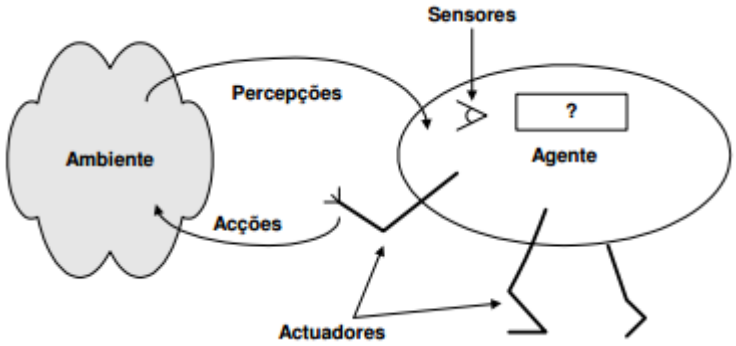

Figura 1 – Esquema típico de um agente. [Reis,2003]

No desenvolvimento de um agente há que especificar as suas necessidades, as suas restrições e as suas preferências. Sendo, indispensável conhecer as suas características, tais como, a comunicação, estatuto, autonomia e adaptabilidade. Deste modo, um determinado agente está dependente do tipo de características que lhe são atribuídas, ou seja, estas características definem como um agente interage num ambiente e com outros agentes [Coelho, 2007].

Wooldridge e Jennings [Wooldridge et al.,1995] definem quais os principais atributos que são essenciais para melhor caraterização de um agente, tais como:

- Autonomia – O agente deve ser capaz de executar as suas tarefas, tomando decisões sem interferência direta de agentes humanos ou de outros agentes computacionais, ou seja, ser capaz de agir independentemente no ambiente, possuindo controlo total sobre suas ações e estado interno;
- Habilidade social – Os agentes interagem entre si por força do meio em que estão. Sempre que não conseguem resolver certos problemas, tem a capacidade de comunicar, com outros agentes do sistema, com o objetivo de os solucionar;
- Reatividade – Os agentes percebem e reagem às diferentes alterações no ambiente em que estão inseridos;

- Pró-Atividade – Os agentes para além de atuar em resposta às alterações ocorridas no seu ambiente, apresentam um comportamento orientado a objetivos, sendo capazes de tomar iniciativas.

Um dos aspetos a ter em conta num modelo baseado em agentes é o tipo de raciocínio que é aplicado ao agente. O raciocínio permite ao agente interpretar e analisar o ambiente e os seus objetivos baseando-se em dados obtidos, em simulações anteriores, ou em regras pré-definidas na programação do modelo. Esse raciocínio pode ser:

- Baseado em regras – Os agentes avaliam o ambiente e executam as suas tarefas tendo em conta um conjunto de condições, pré-definidas na programação do modelo.
- Baseado em conhecimento – Os agentes avaliam o ambiente e executam as suas tarefas através de uma base de dados resultante de simulações anteriormente realizadas.

No geral, agentes são entidades autónomas que atuam num determinado ambiente de forma a interagir com este e com outros agentes. O agente ideal seria aquele que fosse capaz de aprender e tomar decisões sempre que surgisse uma situação nova para ele.

2.2 Sistemas Multiagentes

Os SMA compreendem o uso de um conjunto de vários agentes, cada um com as suas características, que se encontram num ambiente comum. Dentro destes sistemas, os agentes podem estar ou não agrupados em diferentes organizações, conforme Figura 2.

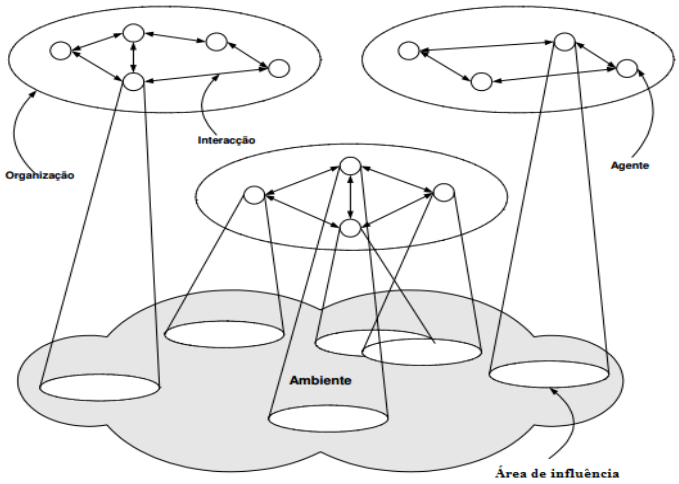

Figura 2 - Sistemas multiagente.[Reis,2003]

Com a finalidade dos agentes realizarem as tarefas a que são propostos, os agentes podem interagir uns com os outros, para isso acontecer, é-lhes conferido características sociais inspiradas nos humanos, tais como a partilha de informações, cooperação e coordenação na execução de tarefas [Reis,2003].

É possível distinguir duas abordagens na construção de SMA: sistemas cooperativos e sistemas competitivos [Reis, 2003]. Num SMA cooperativo, os agentes além de terem objetivos individuais, também colaboram uns com os outros de forma a resolver um problema comum. Desta forma, há uma maior preocupação com o desempenho global do sistema e não tanto com desempenho individual dos agentes. Já nos SMA competitivos, cada agente tem o seu próprio objetivo e motivação, sendo que não se preocupam com o desempenho global do sistema. Este tipo de sistemas é geralmente usado em cenários que envolvam aquisição de bens.

A principal motivação para a utilização de SMA encontra-se relacionada com o facto de a maior parte dos problemas a resolver serem inerentemente distribuídos de uma ou várias formas [Reis, 2003]. Em particular, as seguintes motivações são referidas por [Reis, 2003]:

- A dimensão do problema pode ser demasiado elevada para ser resolvido por um único agente;
- Possibilita a interconexão e interoperação de múltiplos sistemas legados (“legacy”);
- Providenciar uma solução natural para problemas geográfica e/ou funcionalmente distribuídos;
- Para a resolução de problemas deve-se arranjar soluções em que os peritos, os conhecimentos ou as informações necessárias se encontram distribuídos;
- Para que a relação homem-máquina ocorra naturalmente é necessário uma interface cooperativa em que ambos funcionam como agentes no sistema;
- Oferece ainda uma maior clareza e simplicidade conceptual de projeto.

Na generalidade dos casos, um Sistema Multiagente, implica que esses agentes possam trabalhar em conjunto para cumprir um objetivo comum. Desta forma, é indispensável a interação entre diferentes agentes de maneira a atingirem os seus objetivos. Sendo assim, existem várias razões para a utilização de um SMA [Stone et al., 1996; Reis, 2003]:

- Rapidez na resolução de problemas devido à distribuição das diferentes tarefas pelos diferentes agentes;
- Diminuição da comunicação devido ao processamento estar localizado junto à fonte de informação e a comunicação ser realizada a alto-nível;
- Fiabilidade do sistema, porque ao utilizar diferentes agentes não existe pontos de falha comuns;
- Permite o aumento do número de agentes num determinado sistema;
- A simplificação das tarefas individuais de programação, dividindo o problema global em vários subproblemas;
- O estudo da inteligência individual e do comportamento social, pois os SMA permitem a interoperacionalidade entre os agentes;
- A manutenção da privacidade da informação e conhecimentos de cada agente;
- Melhor capacidade de resposta devido aos sensores, sistemas de processamento e atuadores estarem localizados em conjunto, no interior dos agentes;
- Facilidade acrescida de desenvolvimento de sistemas devido à modularidade resultante da decomposição dos problemas e da decomposição dos sistemas em agentes semiautónomos.

O número de aplicações potenciais da tecnologia dos SMA a nível industrial e comercial é enorme, das quais se destacam os sistemas de produção, comércio eletrónico, telecomunicações, redes elétricas de energia, robótica.

2.3 Simulação

A simulação é atualmente uma das ferramentas mais poderosas disponível para os mais diversos sectores, sejam económicos ou áreas da ciência, aos quais permite prevenir possíveis situações ou testar alterações no funcionamento das suas tarefas [Babiceanu et al., 2005].

Shannon, define simulação como "*o processo de conceção de um modelo de um sistema real e a realização de experiências com este modelo para o propósito de compreender o comportamento do sistema e/ou avaliação de várias estratégias para o funcionamento do sistema*" [Shannon, 1998].

2.3.1 Simulação de Eventos Discretos

Na simulação de eventos discretos, procura-se representar um sistema real, através de um conjunto de entidades, com determinados atributos, sendo estas entidades responsáveis por reproduzir as ocorrências do sistema num determinado intervalo de tempo proporcional à duração do sistema real [Hemant et al., 2012].

Nos últimos anos houve um melhoramento e desenvolvimento de programas direcionados para o auxiliar a simulação de eventos discretos. Os *softwares* atuais são mais fáceis de utilizar e apresentam melhor desempenho quando comparados com os anteriores. Foram então desenvolvidos vários *softwares* de simulação orientada a objetos, como por exemplo, Arena, Automod, Simfactory, Simpy, Simevents e Promodel.

2.3.2 Simulação Baseada em Agentes

Nos dias que correm, o uso da simulação baseada em agentes ou modelação baseada em agentes facilita a validação de um conceito, visto que, o grau de complexidade dos processos é cada vez maior e na maioria dos casos, é praticamente impossível recorrer à construção de um sistema físico para caso de estudo.

Para auxiliar a implementação de SBA, nos últimos anos têm sido criadas diversas plataformas, existindo no entanto diferenças significativas entre elas. Cada uma destas plataformas apresenta diferentes métodos na resolução de problemas complexos, na sua análise e avaliação do seu desempenho. Estas ferramentas permitem desenvolver, representar e simular um caso de estudo.

Os agentes contemplados neste tipo de modelação agem de forma autónoma, tem como função o cumprimento de objetivos estabelecidos, podem interagir com o ambiente e estabelecer contacto com outros agentes.

Tendo em conta a cadeia de abastecimento, o recurso aos SBA é vantajoso, visto que há uma grande dinâmica entre os diferentes agentes. Cada um dos agentes vai adaptar o seu comportamento em resposta a estímulos por parte de outros agentes.

2.4 Plataformas de Simulação Baseadas em Agentes

Um dos principais fatores que permitiram o crescimento de MBA, foi a criação de diversas plataformas que nos últimos anos tornaram a modelação baseada em agentes, mais fácil e atrativa para os diferentes campos de investigação, no entanto existe diferenças significativas a nível de suporte que cada uma destas plataformas proporciona. A seguir, serão apresentados os principais simuladores, Swarm, Repast e NetLogo.

2.4.1 Swarm

Originalmente desenvolvido no Santa Fé Institute em 1994 por Chris Langton. Atualmente continua a ser desenvolvido pelo Swarm Development Group.

O Swarm é uma plataforma para modelação baseada em agentes, vem com uma framework conceitual para projetar e executar simulações em MBA, ver Figura 3, assim como diversas ferramentas essenciais para o desenvolvimento de novos modelos e inclui uma grande comunidade de utilizadores que compartilham as suas ideias e simulações.

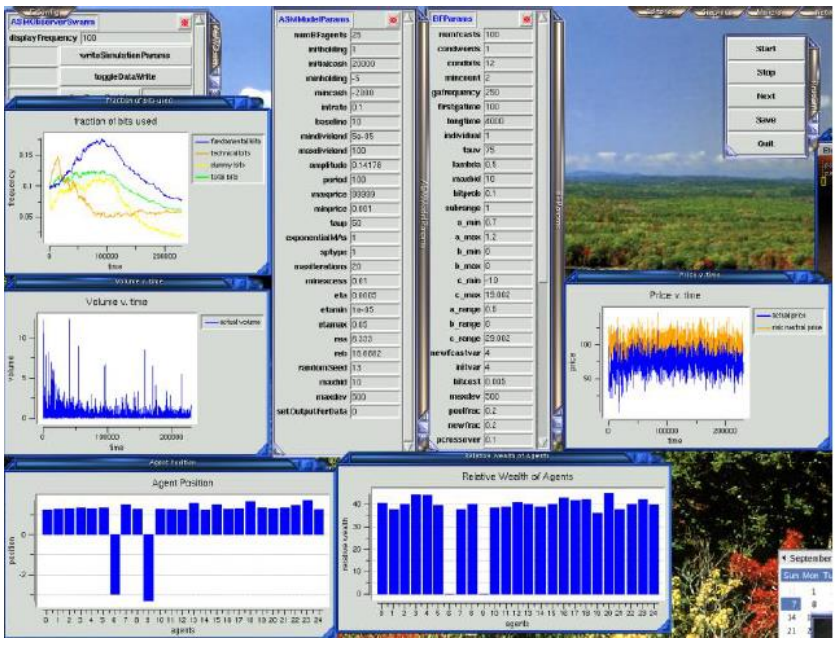

Figura 3 – Exemplo de modelo da plataforma Swarm.

A plataforma Swarm foi inicialmente desenvolvida na linguagem Objective-C, atualmente conta com uma interface Java para simplificar o uso desta plataforma. O Swarm fornece ainda uma grande variedade de bibliotecas para a construção e análise de modelos [Allan, 2009].

2.4.2 Repast

O Repast (Recursive Porous Agent Simulation Toolkit) foi desenvolvido na Universidade de Chicago, mas atualmente é mantido pela Repast Organization for Architecture and Development (ROAD), é um ambiente de simulação para modelação baseada em agentes, muito similar com o Swarm, assim como ele, fornece uma biblioteca com uma grande variedade de códigos e exemplos, para criar, executar, apresentar e recolher dados dos modelos desenvolvidos [Allan, 2009]. O Repast permite o desenvolvimento de um modelo em diversas linguagens como Java, C#, C++, Visual Basic.Net, Lisp, Prolog e Python.

Atualmente o Repast está integrado ao IDE Eclipse, Figura 4, onde é possível criar agentes e ambientes, assim como especificar as suas ações e características [Allan, 2009].

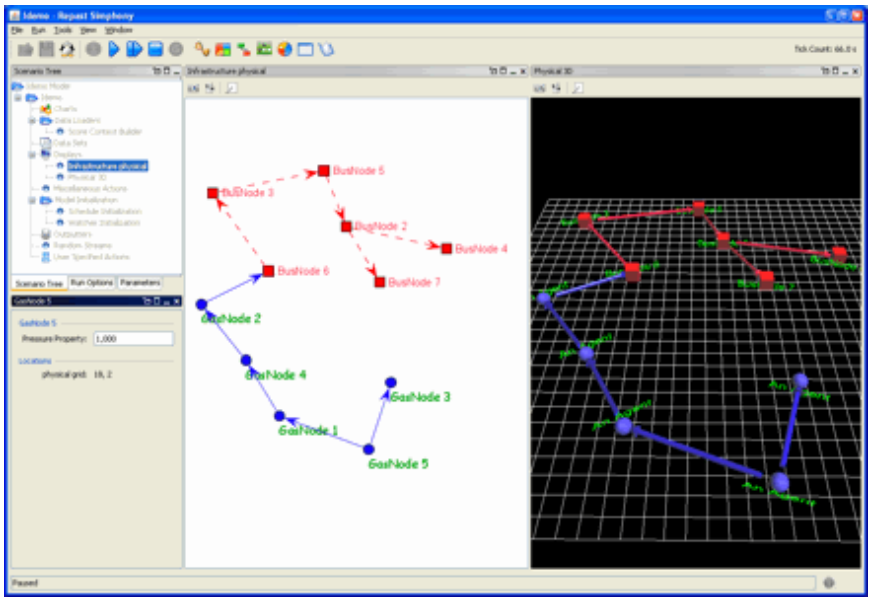

Figura 4- Exemplo de simulação Repast.

2.4.3 NetLogo

O *software* NetLogo foi criado Uri Wilensky, baseado na linguagem de programação Logo, desenvolvida por Seymour Papert em 1967. É um sistema de modelação baseada em agentes que permite aos utilizadores projetar e criar simulações de fenômenos naturais e sociais e avaliar o seu comportamento nesses cenários sob várias condições. Esta ferramenta é particularmente adequada para modelar sistemas complexos que se desenvolvem ao longo do tempo. O programa é livre e está disponível no seu site web. É utilizado em larga escala pelo simples facto de se tratar de uma ferramenta versátil, simples e acessível o suficiente para

professores e alunos de diferentes níveis de escolaridade e ainda uma ferramenta poderosa ao nível de investigação dos mais variados campos da ciência. O Netlogo é executado na máquina virtual Java, por isso funciona nas principais plataformas (Mac, Windows, Linux) [Wilensky, 1999].

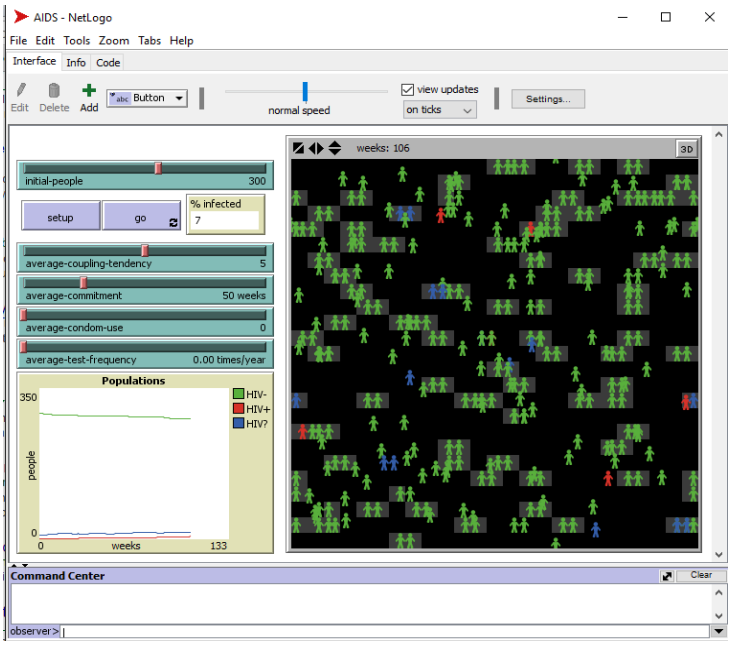

Figura 5- Interface principal do NetLogo.

O NetLogo é composto por três tipos de agentes: observer, turtles e patches. O observer é um agente único responsável pela interligação de todo um mundo virtual, tem como função executar o programa através dos botões que se encontram na interface, conforme a Figura 5, botões estes, que estão associados a linhas de código. Os patches são agentes estacionários organizados numa matriz de quadrados, formando assim o mundo virtual, através do qual os agentes turtles se movem, interagindo entre si e com os patches.

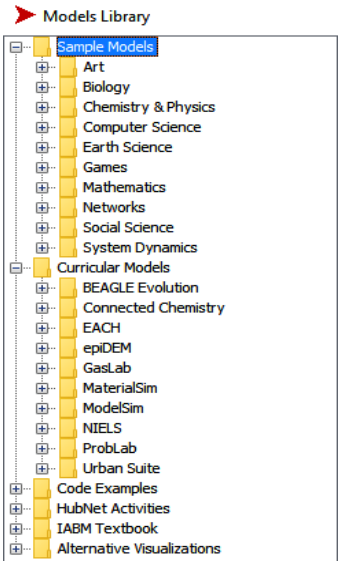

Figura 6- Biblioteca de modelos no NetLogo.

O NetLogo vem acompanhado com uma grande biblioteca de simulações, Figura 6, das mais diversas áreas da ciência, além disso possui uma extensa documentação e tutoriais. O programa permite utilizar e modificar os modelos existentes, tal como o da Figura 7.

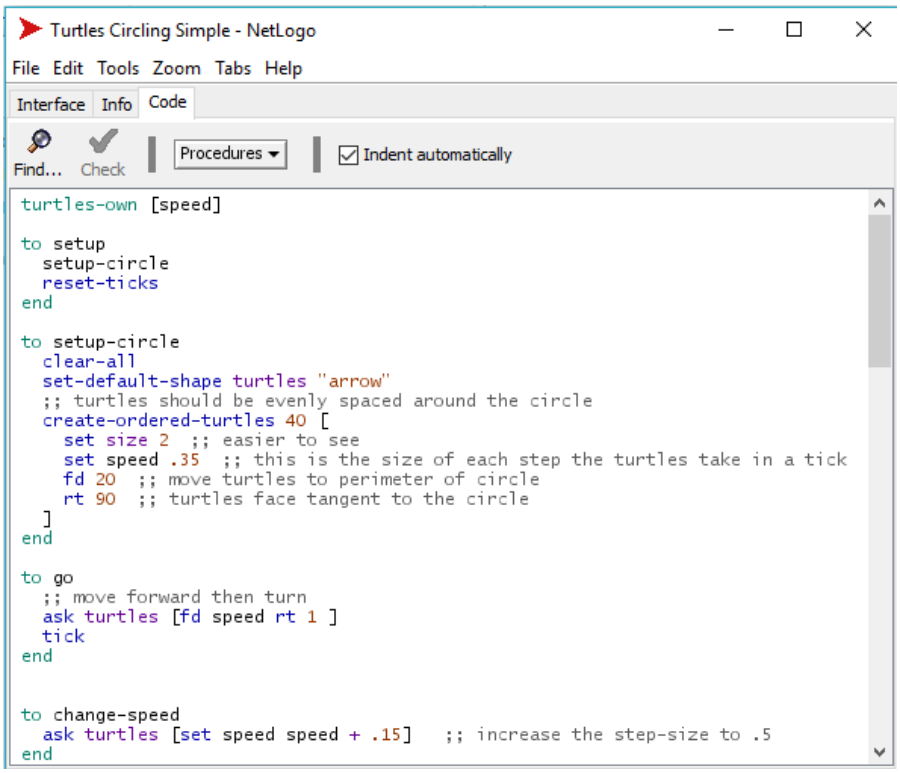

Figura 7 - Exemplo de programação no NetLogo.

Uma das grandes vantagens desta ferramenta, é a sua capacidade de modelar e simular um SMA com milhares de agentes de forma fácil e compreensível, por possuir uma interface de configuração dos parâmetros simples e intuitiva, e por permitir a visualização dos resultados de forma clara.

Capítulo 3

3 Caso de Estudo

Neste capítulo é realizada uma descrição do que é uma cadeia de abastecimento e apresentado o caso de estudo. São descritos aspetos relativos aos fornecedores, fabricantes, distribuidores, clientes, produtos e matéria-prima. Por último, são apresentados alguns trabalhos que têm como caso de estudo a cadeia de abastecimento.

3.1 Conceito

É possível definir cadeia de abastecimento como um conjunto de empresas, responsáveis por transformar a matéria-prima num produto final e por o fazer chegar ao consumidor final [Souza et al., 2006]. Estas empresas são essencialmente constituídas por fornecedores, produtores e distribuidores que se encontram constantemente a interagir, quer a trocar informações quer a trocar produtos.

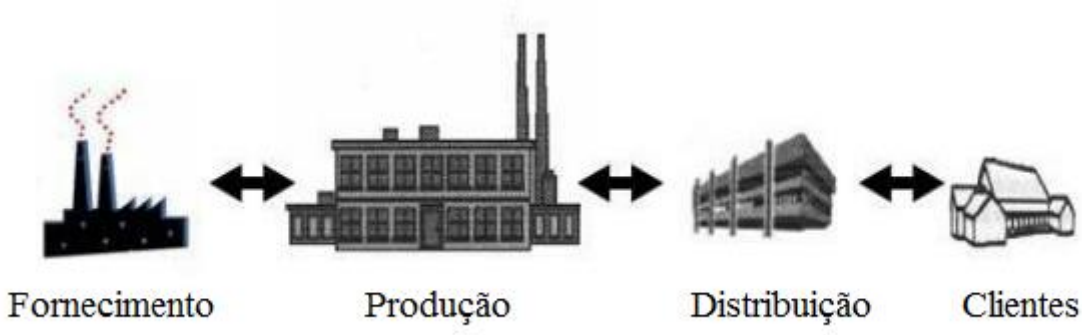

Figura 8- Cadeia de abastecimento típica.

Nesta estrutura da cadeia abastecimento mais tradicional, ilustrada na Figura 8, é importante realçar os seguintes pontos [Beamon, 1999]:

- Produção/Distribuição – planear todas as atividades direta ou indiretamente relacionadas com a produção e/ou distribuição;
- Níveis de Stock – determinar a quantidade armazenada de matérias-primas e produtos acabados;
- Número de níveis da cadeia logística – que engloba decisões a nível da integração vertical/horizontal;
- Número de centros de distribuição – em conjunto com o número de centros de produção;
- Relação Comprador/Vendedor – avaliar todos os aspetos críticos da relação entre o comprador e o vendedor;
- Especificação e diferenciação do produto – que resulta de um conjunto de iniciativas, usualmente provenientes do departamento de vendas e de marketing, definindo as especificações desse mesmo produto.

No geral, os materiais e a informação fluem ao longo dos diferentes níveis, com o objetivo de satisfazer as necessidades em cada um dos pontos da cadeia de abastecimento.

3.2 Caso de estudo

O caso de estudo considerado neste trabalho é baseado numa cadeia de abastecimento tradicional, constituída por quatros entidades, conforme ilustrado na Figura 9: fornecedor, fabricante, distribuidor e cliente. De uma maneira geral, os aspetos mais importantes deste sistema são a produção, o armazenamento e o transporte de produtos.

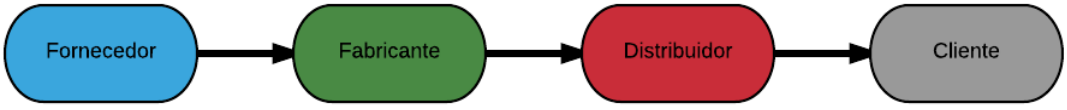

Figura 9- Visão geral da cadeia de abastecimento.

Neste caso de estudo, existem dois tipos de produtos disponíveis no mercado para os clientes, “Produto A” e “Produto B”. Os clientes tem acesso aos produtos através dos distribuidores, os quais tem capacidade de decidir que pretendem comercializar os dois produtos ou apenas um. No início do processo, os fabricantes também escolhem qual o

produto que pretendem produzir, podendo escolher um ou os dois. Para a produção apenas existe um tipo de matéria-prima.

Para o produto chegar à entidade cliente, há negociação desde o fornecedor ao distribuidor. Para o distribuidor armazenar e ter produtos disponíveis para vender a qualquer momento necessita encomendar novos produtos, sendo assim, vai ocorrer uma negociação entre o distribuidor e os diversos fabricantes no mercado. Para esta negociação é necessário o tipo de produto, o preço e o tempo de entrega. A outra negociação ocorre entre o fabricante e os fornecedores, com a finalidade de reabastecer os níveis de stock de matéria-prima das indústrias. Nesta negociação é necessário o preço e o tempo de entrega da matéria-prima. Nestas negociações as entidades compradoras têm duas opções. Caso estas entidades não necessitem urgentemente dos produtos ou da matéria-prima optam sempre pelo menor preço a pagar, caso contrário optam pelo tempo de entrega mais rápido.

O sistema começa com o fornecedor, este tem como função a exploração e armazenamento da matéria-prima. O fornecedor interage apenas com a entidade produtora, com o objetivo de suprimir as necessidades das indústrias.

O fabricante tem como função a produção e armazenamento de novos produtos. Tem uma relação compra/venda com os fornecedores e tem como objetivo fazer chegar esses produtos aos centros de distribuição.

O distribuidor tem como função armazenar e vender produtos. Tem uma relação compra/venda com os fabricantes e tem como objetivo o escoamento do produto das fábricas para posterior venda.

O cliente tem como função o consumo dos produtos do mercado.

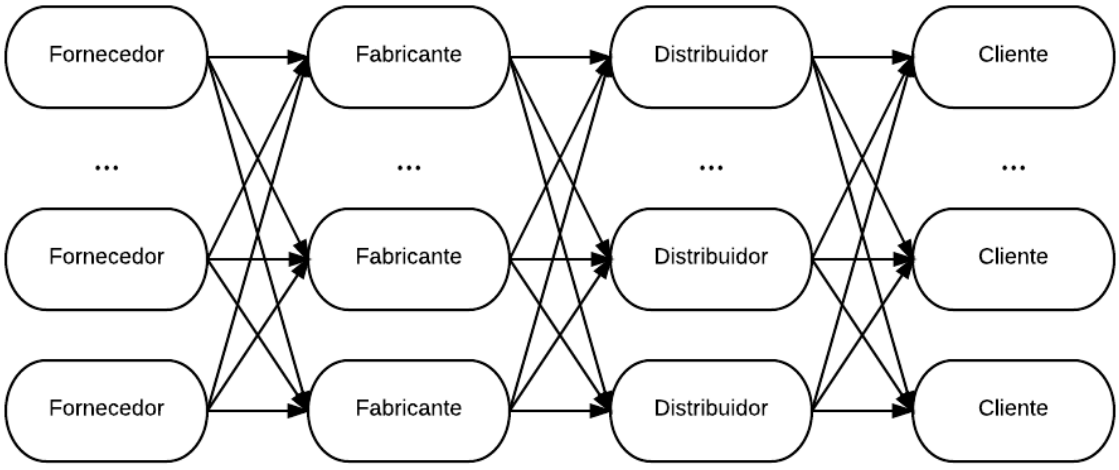

Figura 10 - Cadeia de abastecimento complexa.

Uma cadeia de abastecimento pode ser uma rede simples, apenas com um fornecedor, um fabricante e um distribuidor, como ilustrado na Figura 9, ou pode ser uma rede complexa,

como ilustrado na Figura 10. Numa rede complexa, há maior variedade de empresas para cada tipo de entidade, maior concorrência entre as entidades criadoras de valor e uma maior oferta de produtos. Contudo, estas entidades dentro das suas responsabilidades, tais como, comprar, vender e entregar produtos, procuram desenvolver relacionamentos inter-empresas de modo a contemplar o objetivo comum de fazer chegar o produto ao consumidor final, sem esquecer claro os objetivos individuais de cada entidade.

Neste trabalho pretende-se que o modelo suporte de uma forma fácil e flexíveis diferentes números de fabricantes, fornecedores e distribuidores, que serão definidos pelo utilizador através da interface gráfica. Do mesmo modo, a parametrização da taxa de produção de cada produto, assim como o volume de encomendas, também será definida pelo utilizador através da interface gráfica.

3.3 Trabalhos existentes

Nesta seção, serão apresentados trabalhos sobre modelos baseados em agentes aplicados à cadeia de abastecimento.

Eymann e outros [Eymann et al., 1998] desenvolveram um modelo baseado em agentes que permite definir os níveis da cadeia com suas políticas de preço. Então, cada nível da cadeia propõe um preço de venda que contém uma margem de negociação para ser utilizada com os agentes da cadeia. O objetivo deste modelo é analisar o preço final médio com que o produto chega ao consumidor.

Sarker e outros [Sarker et al., 2005] apresentam uma cadeia de abastecimento constituída por fornecedores, produtores, distribuidores, retalhistas e clientes. Este trabalho mostra que a falta de coordenação na troca de informações entre os níveis da cadeia vai provocar um acumular de materiais nos níveis da cadeia que se encontram mais distantes do consumidor final.

Outro modelo semelhante é o de Fox [Fox et al., 2000] onde também se modela cada nível da cadeia como um agente, adicionando interações entre os níveis da cadeia que se relacionam. A análise ao comportamento dos agentes é em termos do stock de matéria-prima, do stock de produto e de prazos de entrega.

Já Carvalho em [Carvalho et al., 2005] apresenta uma simulação de um sistema multiagente onde é possível escolher diferentes estratégias e táticas de decisão, e diferentes

critérios de avaliação. Na simulação usa modelos matemáticos para determinar os stocks de materiais.

O objetivo deste estudo é estudar a resposta da cadeia de fornecimento a diferentes estruturas desta (isto é diferentes números de fabricantes e distribuidores, assim como volumes de encomendas), e considerar mecanismos de balanceamento dinâmico e automático do fluxo de produtos.

Capítulo 4

4 Desenvolvimento do Modelo de Simulação

No presente capítulo irá ser apresentado o desenvolvimento do modelo de simulação de uma cadeia de abastecimento.

A implementação deste modelo foi realizada com o auxílio da plataforma Netlogo, dadas as suas características e vantagens no estudo de MBA como anteriormente discutido no capítulo 2.

4.1 Desenvolvimento do Modelo Baseado em Agentes

Com o recurso à ferramenta Netlogo, foi desenvolvido um modelo baseado em agentes para uma cadeia de abastecimento, que simula a produção de materiais, o transporte de diferentes materiais e a comunicação entre agentes. O modelo é composto por 4 tipos de diferentes agentes, definidos de acordo com a cadeia de Abastecimento, sendo eles "fornecedor", "fabricante", "distribuidor" e "cliente".

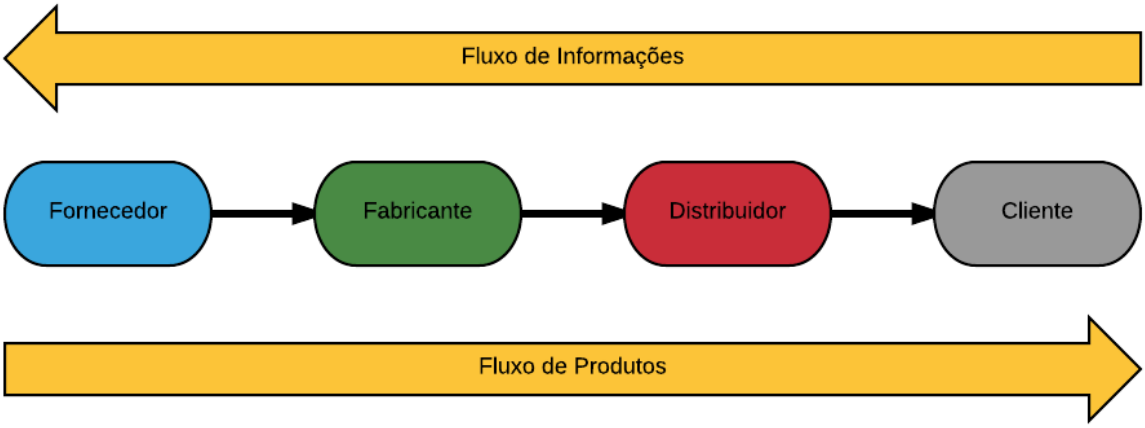

Figura 11 – Relação entre agentes.

A parte principal deste modelo de simulação consiste na troca de informação e de produtos entre os diferentes níveis da cadeia, conforme a Figura 11. Estas interações entre agentes seguem este sentido linear:

- O agente “Fornecedor” interage com o agente “Fabricante” para entregar matéria-prima;
- O agente “Fabricante” interage com o agente “Fornecedor” para fazer encomendas de matéria-prima e interage com o agente “Distribuidor” para entregar produtos;
- O agente “Distribuidor” interage com o agente “Fabricante” para fazer encomendas de produtos e interage com o agente “Cliente” para entregar produtos;
- O agente “Cliente” interage com o agente “Distribuidor” para comprar produtos.

Escolhidos os agentes do modelo, de acordo com o sub-capítulo 2.1, é necessário conferir-lhes um comportamento autónomo, assim sendo, foram atribuídos especificamente a cada um dos agentes certas características e atributos que vão influenciar o modo como percecionam e interagem entre eles.

Tabela 1- Agentes e respetivas variáveis

Variáveis	Agentes			
	Fornecedor	Fabricante	Distribuidor	Cliente
ID	X	X	X	
Preço	X	X		
Stock Produtos		X	X	
Stock Matéria-Prima	X	X		
Capacidade Produção		X		
Tipo de Produto		X	X	
Tempo de Entrega	X	X		
Encomendas	X	X	X	
Vendas	X	X	X	
Compras				X

No ponto de vista da simulação foi necessário dividir algumas destas variáveis em três categorias diferentes, sendo elas: fixas, aleatórias e autónomas.

Antes de se iniciar qualquer simulação, é dada a possibilidade ao utilizador de alterar algumas variáveis em que vai decorrer a simulação, ver secção 4.2.1. Sendo assim designadas por variáveis fixas, porque ao longo da simulação não se alteram.

As variáveis aleatórias começam por ser definidas no início de cada simulação e podem ser alteradas no decorrer da mesma. As variáveis que cumprem esta restrição são as seguintes:

- Preços;
- Tipo de Produto;
- Tempo de entrega.

Estas variáveis cumprem um papel muito importante na simulação contribuindo para as tomadas de decisão, permitindo aos diferentes agentes escolher a melhor solução.

São consideradas como variáveis autónomas todas aquelas que no decorrer da simulação se vão alterando, tais como:

- Encomendas;
- Vendas;
- Capacidade de produção;
- Stocks;
- Compra.

Estas variáveis são peças fundamentais na comunicação entre agentes.

4.1.1 Fornecedor

O agente "Fornecedor" tem como objetivo suprimir as necessidades de matéria-prima do agente "Fabricante". Cada um dos fornecedores é identificado pelo "ID", que corresponde a um número e tipo de agente. É um agente não produtivo, trata da procura de matéria-prima para vender ao agente "Fabricante", o excesso armazena em stocks com um limite máximo, define um preço e o tempo de entrega, no entanto, só pode fazer uma encomenda por cada tick ou dia com um limite máximo de unidades.

O seguinte pseudo-código é responsável por correr a negociação entre os agentes fabricantes e fornecedores.

```
Para negotiation_factory_industry
    Para cada agente Fabricante
       Encomenda
       Enquanto( encomenda != 0)
        Se tempo de entrega = 0
              Encomenda não Urgente
              Melhor Preço
              Para cada agente Fornecedor
               Se Stock > Encomenda
                   Venda = Encomenda
                   Tempo de entrega
                   Encomenda = 0
                Se não
                   Venda = Stock
                   Tempo de entrega
                   Encomenda = Encomenda - Stock
         Se não
              Encomenda Urgente
              Menor Tempo de Entrega
```

```
                Para cada agente Fornecedor
                 Se Stock > Encomenda
                    Venda = Encomenda
                    Tempo de entrega
                    Encomenda = 0
                 Se não
                    Venda = Stock
                    Tempo de entrega
                    Encomenda = Encomenda - Stock
Fim
```

Para começar ele seleciona a encomenda do fabricante, se não tiver caracter urgente ele opta pelo fornecedor que pratica o melhor preço, caso contrário ele opta pelo que entrega a encomenda mais rápido. Depois, negoceia com o fornecedor até a encomenda ser zero, passando ao próximo fabricante.

4.1.2 Fabricante

O agente "Fabricante" tem como objetivo transformar a matéria-prima adquirida ao agente "Fornecedor" num produto final, para ser comprada pelo agente "Distribuidor", que posteriormente será revendida ao agente "Cliente". Cada um dos fabricantes é identificado pelo "ID", que corresponde a um número e tipo de agente. É um agente produtivo, encomenda matéria-prima ao agente "Fornecedor", no entanto, só pode fazer uma encomenda por cada tick ou dia com um limite máximo de unidades, pode produzir dois tipos de produtos ou apenas um deles, designados "Produto A" ou "Produto B", define um preço de venda e um prazo de entrega do produto, em stock tem a matéria-prima com um limite máximo e um mínimo para garantir um dia de produção normal na fábrica e também tem um stock para armazenamento de produtos.

O seguinte pseudo-código é responsável pela produção nos agentes fabricantes.

```
Para production_A
    Para cada agente Fabricante
       Se Stock A < 500
          Se Stock MP >= 0
             Se stock MP < Capacidade de Produção * 2
                Stock A = Stock A + (Stock MP / 2)
                Stock MP = 0
             Se não
                Stock A = Stock A + Capacidade de Produção
                Stock MP = Stock MP – (Capacidade de Produção * 2)
Fim
```

Para começar, verifica se o stock do produto não é superior a 500 e se há stock de matéria-prima para iniciar a produção. Depois, caso o stock de matéria-prima seja inferior à

quantidade necessária para atingir o máximo de produção da fábrica (1 Produto = 2 Matéria-Prima) a produção será igual à metade do stock de matéria-prima e o stock será igual a zero. Caso contrário, o fabricante produz à capacidade máxima.

4.1.3 Distribuidor

O agente "Distribuidor" tem como objetivo o escoamento do produto do agente "Fabricante", para depois vender ao agente "Cliente". Cada um dos distribuidores é identificado pelo "ID", que corresponde a um número e tipo de agente. Cada centro de distribuição pode vender um ou dois tipos de produtos, que são encomendados ao agente "Fabricante", no entanto, cada agente só pode fazer uma encomenda por cada tick ou dia com um limite máximo de unidades, também, define um preço de revenda e armazena produtos.

O seguinte pseudo-código é responsável por correr a negociação de produtos entre os agentes distribuidores e fabricantes. No programa desenvolvido, há um código destes para cada produto.

```
Para negotiation_client_factory_A
    Para cada agente Distribuidor
       Encomenda
       Enquanto( encomenda != 0)
       Se tempo de entrega = 0
            Encomenda não Urgente
            Melhor Preço
            Para cada agente Fabricante
               Se Stock > Encomenda
                  Venda = Encomenda
                  Tempo de entrega
                  Encomenda = 0
               Se não
                  Venda = Stock
                  Tempo de entrega
                  Encomenda = Encomenda - Stock
       Se não
            Encomenda Urgente
            Menor Tempo de Entrega
            Para cada agente Fabricante
               Se Stock > Encomenda
                  Venda = Encomenda
                  Tempo de entrega
                  Encomenda = 0
               Se não
                  Venda = Stock
                  Tempo de entrega
                  Encomenda = Encomenda - Stock
```

Para começar ele seleciona a encomenda do “Distribuidor”, se não tiver caracter urgente ele opta pelo fabricante que pratica o melhor preço, caso contrário ele opta pelo que entrega a encomenda mais rápido. Depois, negoceia com o fabricante até a encomenda ser zero, passando ao próximo distribuidor.

4.1.4 Cliente

O agente “Cliente” tem como objetivo a compra de produtos no agente “Distribuidor”. É um agente abstrato na simulação, mas não menos importante que os outros, a partir dele é que começa o desenrolar da cadeia de logística. A única variável que possui, “compra” vai afetar os níveis de stock dos centros de distribuição, que por sua vez vai influenciar as encomendas de produto às fábricas, estas que por fim, necessitam encomendar matéria-prima para repor o stock.

O seguinte pseudo-código é responsável por originar as encomendas, tendo em conta as vendas dos distribuidores aos clientes.

```
Para encomends_client_A
     Para cada agente Distribuidor
       Se Vende Produto A
         Vendas = 30 + (random 60)
          Stock = Stock – Vendas
       Se Vendas <= 70
          Encomendas = Vendas
        Se não
          Encomendas = 70
        Se Stock > 600
          Encomenda = 0
        Se Stock <= 200
          Encomenda Urgente
          Tempo de entrega = 1
          Encomenda = 70
        Se não
          Encomenda não Urgente
Fim
```

As vendas nos agentes distribuidores por dia variam entre 30 e 90. Se forem igual ou inferior a 70, a encomenda é igual ao número vendas, caso contrário é igual ao limite máximo por encomenda 70. Se o stock for superior a 600 não haverá encomenda. Ainda, se o stock for inferior a 200 a encomenda será considerada urgente, caso contrario não é urgente.

4.2 Ambiente de Simulação do Modelo

O modelo baseado em agentes descrito atrás, foi implementado na plataforma NetLogo, sendo a interface principal ilustrada na Figura 12. O mundo virtual em que se encontram é estático, não existe nenhum tipo de interação entre agentes que seja percetível durante a simulação, só a nível estatístico.

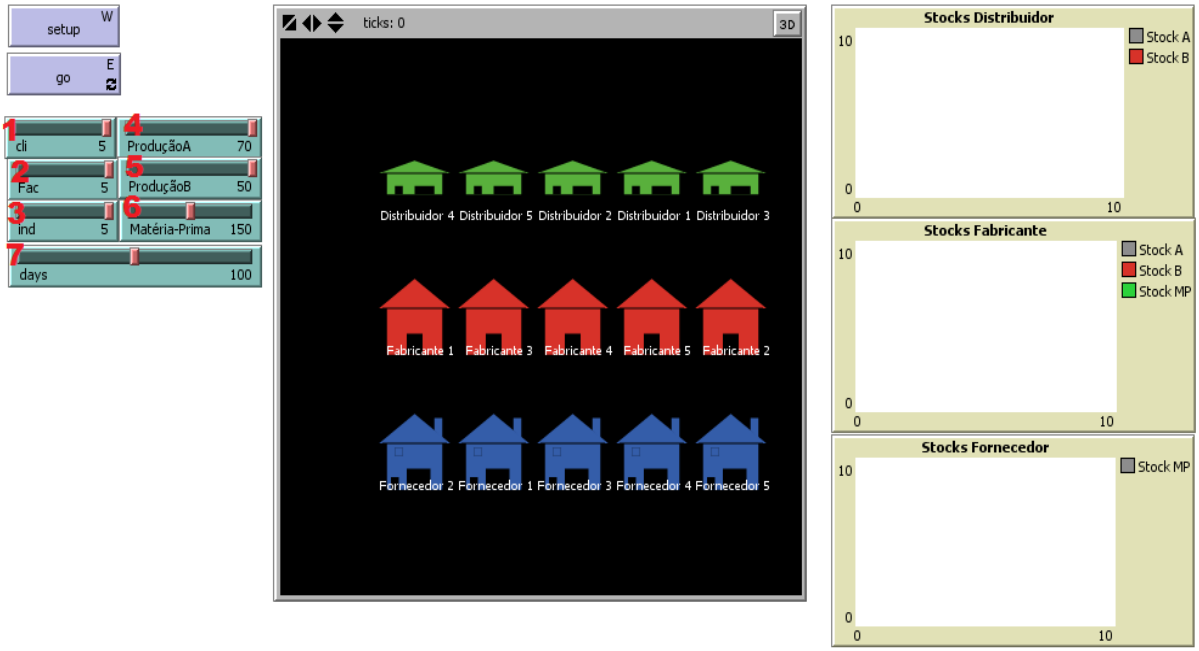

Figura 12- Interface principal do modelo criado no ambiente NetLogo.

A interface principal está estruturada de acordo com o diagrama de blocos da Figura 13. Nos dados de entrada é possível definir os parâmetros em que vai correr a simulação. A simulação é representada pelo mundo virtual na interface e está relacionada com o processamento de uma sequência de etapas programadas (anexo A). Os dados de saída estão relacionados com amostragem dos resultados, neste modelo, apresentados graficamente.

Figura 13 – Diagrama de blocos do modelo.

4.2.1 Introdução de Dados no Modelo

Os dados de entrada do modelo são definidos nos ***sliders*** da interface principal, conforme ilustrado na Figura 13, que posteriormente serão introduzidos no modelo quando for pressionado o botão ***setup,*** sendo eles:

1) *Slider* que define o número de distribuidores;
2) *Slider* que define o número de fabricantes;
3) *Slider* que define o número de fornecedores;
4) *Slider* que define a capacidade máxima de produção do produto A;
5) *Slider* que define a capacidade máxima de produção do produto B;
6) *Slider* que define a capacidade máxima de repor os níveis de stock de matéria-prima pelos fornecedores;
7) *Slider* que define a duração total da simulação;

O seguinte código, é um exemplo de como são introduzidos os dados dos ***sliders*** aos respetivos agentes. Neste caso, é demostrado como é colocada a capacidade máxima de produção do produto A.

```
set capacity_pro_A ProduçãoA
```

4.2.2 Setup do Modelo

Para configurar os dados de entrada da simulação é pressionado o botão ***Setup***. Este é criado na interface principal e posteriormente programado. Quando este botão é pressionado, cria o mundo, cria os agentes tendo em conta os valores definidos nos ***sliders*** da interface principal, define o tipo de produto que os distribuidores vendem, o tipo de produto que os fabricantes produzem, preços, tempos de entrega e os valores iniciais de algumas variáveis, por exemplo stocks. O botão ***setup*** executa o seguinte pseudo-código.

```
Para setup
       Cria Distribuidor
       Cria Fabricante
       Cria Fornecedor
Fim
```

O botão ***Setup***, cria cada um dos agentes com as respetivas variáveis, definidas na Tabela 1 na secção 4.1 Desenvolvimento do Modelo.

4.2.3 Execução da Simulação

O botão ***go*** inicia a simulação entrando num ciclo de tempo, que apenas termina quando o número de ***ticks*** for igual ao número de dias de simulação definido aquando da criação do mundo com o botão ***setup***. De referir que a escala de tempo utilizada foi o dia, que na ferramenta Netlogo corresponde a um tick.

Cada ciclo corre um conjunto de sub-programas. Da mesma forma, o botão ***go*** executa o seguinte pseudo-código.

```
Para go
        Se tick = dias [stop]
        Encomendas Distribuidor
        Negociação Distribuidor/Fabricante
        Transporte Fabricante/Distribuidor
        Produção Fabricante
        Encomendas Fabricante
        Negociação Fabricante/Fornecedor
        Transporte Fornecedor/Fabricante
        Matéria-Prima Fornecedor
        Histogramas
        tick
Fim
```

A simulação começa com a formalização das encomendas dos distribuidores, após as vendas de produto aos clientes. No próximo passo ocorre a negociação entre os distribuidores e fabricantes. No passo anterior o distribuidor gerou uma encomenda, com tipo de produto, carácter de urgência e quantidade de produto. Agora, o distribuidor vai escolher o fabricante que melhor se adequa a esses parâmetros. Após a negociação, os produtos dão saída dos stocks dos fabricantes e fica guardado numa base de dados quando é que o produto tem de ser entregue. Sendo que, se o tempo de entrega for 1, a encomenda é entregue assim que o ciclo termine, caso seja 2, é entregue após terminar o ciclo do dia seguinte. Assim que é dada a baixa dos produtos nos fabricantes, é iniciado o processo de produção tendo em conta os níveis de stock de produto. A produção de novos produtos, vai afetar o stock de matéria-prima. Posto isto, é feita uma encomenda aos fornecedores com a quantidade e o carácter de urgência.

A seguir, começa a negociação entre os fabricantes e os fornecedores, com os parâmetros definidos anteriormente, escolhendo o fornecedor que cumpre com os requisitos. A entrega de produtos do fornecedor para os fabricantes ocorre da mesma forma que a entrega de produtos do fabricante para os distribuidores. Após a saída de matéria-prima, dá-se inicio à reposição

dos stocks de matéria-prima, com um limite máximo por dia. Por último, a cada ***tick*** são apresentados os níveis de stock de todos os agentes envolvidos na simulação.

A imagem seguinte é um fluxograma do modelo de simulação desenvolvido. A simulação é executada da direita para a esquerda tendo em conta a cadeia de abastecimento. Cada simulação obedece a uma série de etapas e compreende um dia típico de trabalho.

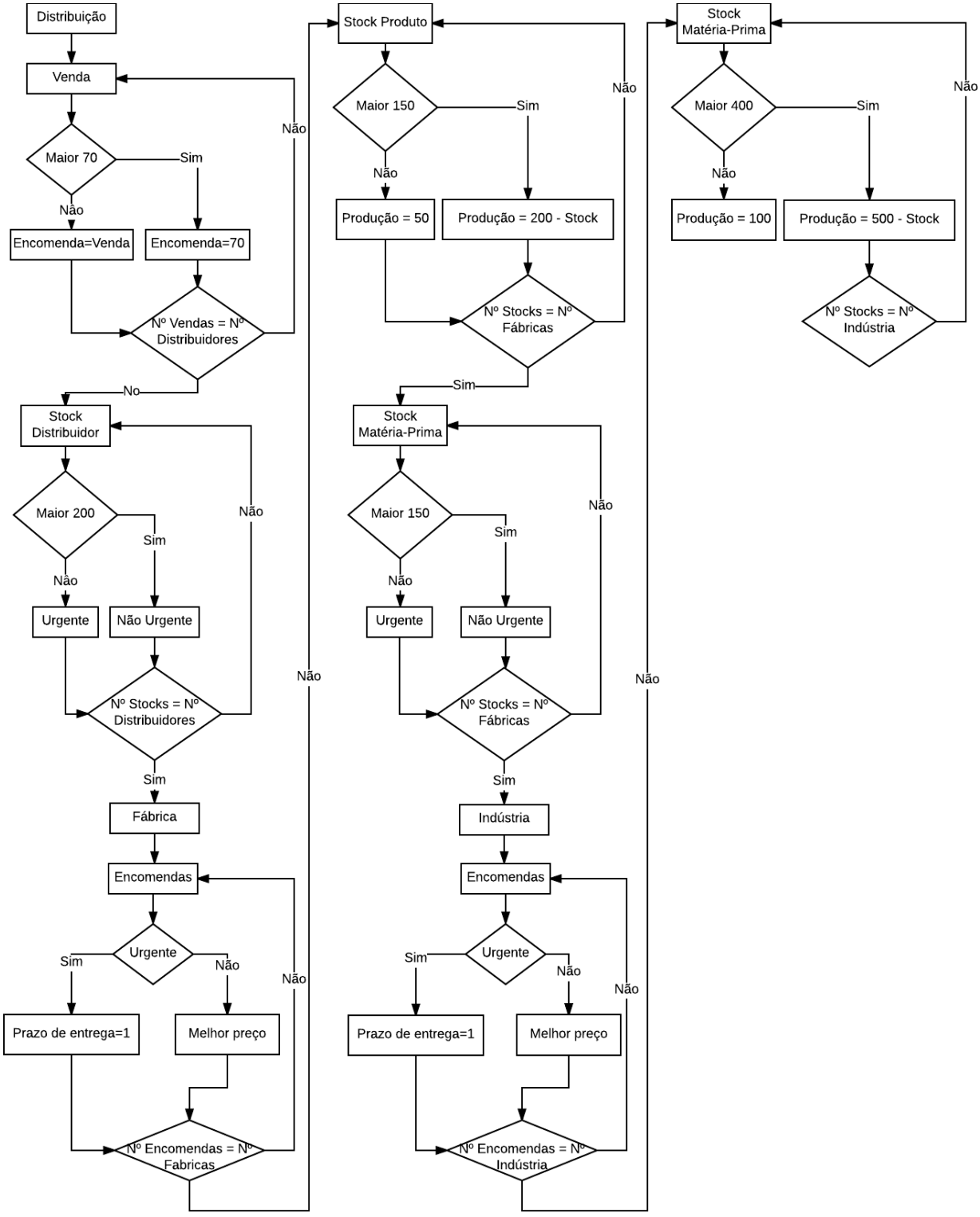

Figura 14 - Fluxograma da simulação.

A simulação começa com a venda dos produtos em stock por parte dos agentes "Distribuidor" aos agentes "Cliente". De seguida, o agente "Distribuidor" vai ter de fazer uma encomenda para repor o stock. Essa encomenda é composta por: tipo de produto, quantidade e urgência. Estes três parâmetros são indispensáveis para a escolha da fábrica que dará resposta à encomenda. O tipo de produto a encomendar é igual ao que é vendido. Lembrando que existe um limite máximo de produtos que é possível encomendar num dia, a quantidade a encomendar está diretamente relacionado com a quantidade vendida. Assim sendo, se o número de vendas for superior ao limite máximo, a encomenda será igual ao valor máximo que se pode encomendar, senão, a encomenda será igual ao número de vendas. Para finalizar, é necessário definir o carácter de urgência da encomenda. Então, se o nível de stock estiver abaixo da metade do limite máximo de stock, a encomenda é urgente e o produto tem de ser entregue no prazo de 1 dia. Caso contrário, a encomenda não é considerada urgente, não ficando obrigada a qualquer prazo de entrega, optando sempre pelo melhor preço do mercado.

A próxima fase é selecionar o agente "Fabricante" que dará resposta à encomenda. Tendo este que corresponder aos parâmetros definidos anteriormente. De seguida, o agente "Fabricante" inicia o processo de produção para repor o stock de produtos. O que vai ditar a quantidade a ser produzida é o nível de stock. Se o nível de stock estiver abaixo do considerado ideal, a capacidade de produção é máxima, caso contrário, é igual ao número de produtos vendidos. Por consequência, o de stock de matéria-prima é gasto, sendo necessário fazer uma encomenda aos agentes "Fornecedores". Essa encomenda é composta por: quantidade e urgência. Estes dois parâmetros são indispensáveis para a escolha do fornecedor que dará resposta à encomenda. Lembrando que existe um limite máximo de matéria-prima que é possível encomendar num dia, a quantidade a encomendar está diretamente relacionado com a quantidade vendida. Assim sendo, se o número de vendas for superior ao limite máximo, a encomenda será igual ao valor máximo que se pode encomendar, senão, a encomenda será igual ao número de vendas. Para finalizar, é necessário definir o carácter de urgência da encomenda. Então, se o nível de stock estiver abaixo da metade do limite máximo de stock, a encomenda é urgente e o produto tem de ser entregue no prazo de 1 dia. Caso contrário, a encomenda não é considerada urgente, não ficando obrigada a qualquer prazo de entrega, optando sempre pelo melhor preço do mercado.

A próxima fase é selecionar o agente "Fornecedor" que dará resposta à encomenda. Tendo este que corresponder aos parâmetros definidos anteriormente. De seguida, o agente "Fornecedor" vai ter de fazer uma encomenda para repor o stock de matéria-prima. O que vai ditar a quantidade a ser encomendada é o nível de stock. Se o nível de stock estiver abaixo do

considerado ideal, é encomendado o máximo possível, caso contrário, é igual ao número de matéria-prima vendida.

4.2.4 Apresentação dos Resultados

A apresentação dos resultados é feita com recurso a três histogramas, representando os níveis de stock dos agentes fornecedor, fabricante e distribuidor. Na interface do Netlogo, são criados os três gráficos através da ferramenta ***plot***. Depois de criados, dá-se um título a cada gráfico e escolhe-se a escala e o tipo de gráfico, entre histograma, linha ou por pontos. Para que seja possível apresentar os resultados da simulação, é necessário criar e atribuir um nome às variáveis, chamadas ***pen***, assim como, o tipo de cor.

O seguinte pseudo-código, apresenta os resultados da simulação.

```
Para histo
 set-current-plot "Stocks Fornecedor"
 Para cada agente Fornecedor
      set-current-plot-pen "Stock MP" x=Posição y=stock_MP
 set-current-plot "Stocks Fabricante"
 Para cada agente Fabricante
      set-current-plot-pen "Stock A" x=Posição y=stock_A
      set-current-plot-pen "Stock B" x=Posição y=stock_B
      set-current-plot-pen "Stock MP" x=Posição y=stock_MP
 set-current-plot "Stocks Distribuidor"
  Para cada agente Distribuidor
      set-current-plot-pen "Stock A" x=Posição y=stock_A
      set-current-plot-pen "Stock B" x=Posição y=stock_B
Fim
```

Para apresentar os resultados gráficos, basta indicar o nome do gráfico e atribuir à ***pen*** a variável que se pretende apresentar e em que posição. Na Figura 15, é possível observar um exemplo de apresentação de resultados.

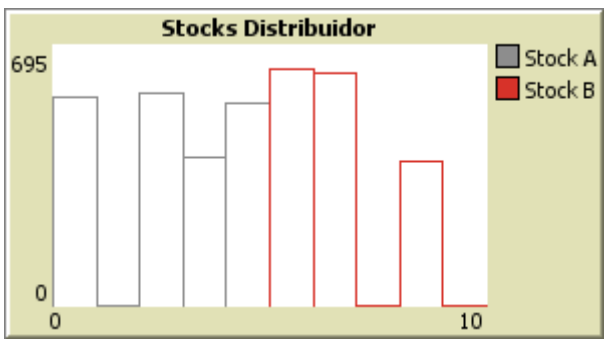

Figura 15 – Exemplo de um histograma obtido numa simulação.

Capítulo 5

5 Análise dos resultados experimentais

Concluído o modelo de simulação baseado em agentes para cadeias de abastecimento é chegada a altura de simular diferentes cenários e avaliar o comportamento dos diferentes agentes.

Nesta simulação existe um interesse especial na coordenação dentro das diferentes atividades e da coordenação entre as mesmas, que permitem uma estabilização dos níveis de stock.

Na simulação, a etapa de tempo considerada é o dia. No decorrer da simulação, cada agente vai registando alguns dados e resultados do dia-a-dia para posterior análise. Para se aceder a estes dados o NetLogo permite inspecionar individualmente qualquer agente, enquanto os resultados são apresentados graficamente. Os dados que vão sendo registados pelos agentes são os preços, produtos, níveis de stock, encomendas, tempos para entrega e vendas. Os resultados apresentados graficamente são os níveis de stock dos agentes.

Para demonstrar a aplicação do sistema de simulação desenvolvido nos diferentes processos de uma cadeia de abastecimento, foram testados dois casos.

No Caso I, a simulação compreende 100 dias e considera dois distribuidores, dois fabricantes e um fornecedor.

Tabela 2 - Dados de entrada do Caso I.

Distribuidor	Dados de Entrada
	Produto
1	A; B
2	A; B

Fornecedor	Dados de Entrada		
	Preço	Tempo	Matéria-Prima
1	13	1	150

Fabricante	Dados de Entrada						
	Produto	Preço A	Tempo A	Preço B	Tempo B	ProduçãoA	ProduçãoB
1	A	41	1	-	-	50	-
2	A; B	43	1	60	2	50	30

É possível observar na Tabela 2 os atributos de cada tipo de agente da simulação do Caso I. Os dois agentes distribuidores vendem os dois tipos de produto. No caso dos agentes fabricantes não acontece o mesmo, sendo que o "fabricante 1" produz apenas o produto A e demora 1 dia para o entregar, já o "fabricante 2" produz dois tipos de produto, sendo que demora 1 dia para entregar o produto A e 2 dias para entregar o produto B. O agente fornecedor demora 1 dia para entregar a matéria-prima. Os atributos "ProduçãoA", "ProduçãoB" e "Matéria-Prima" são definidos no começo das simulações, conforme explicado no sub-capítulo 4.1.

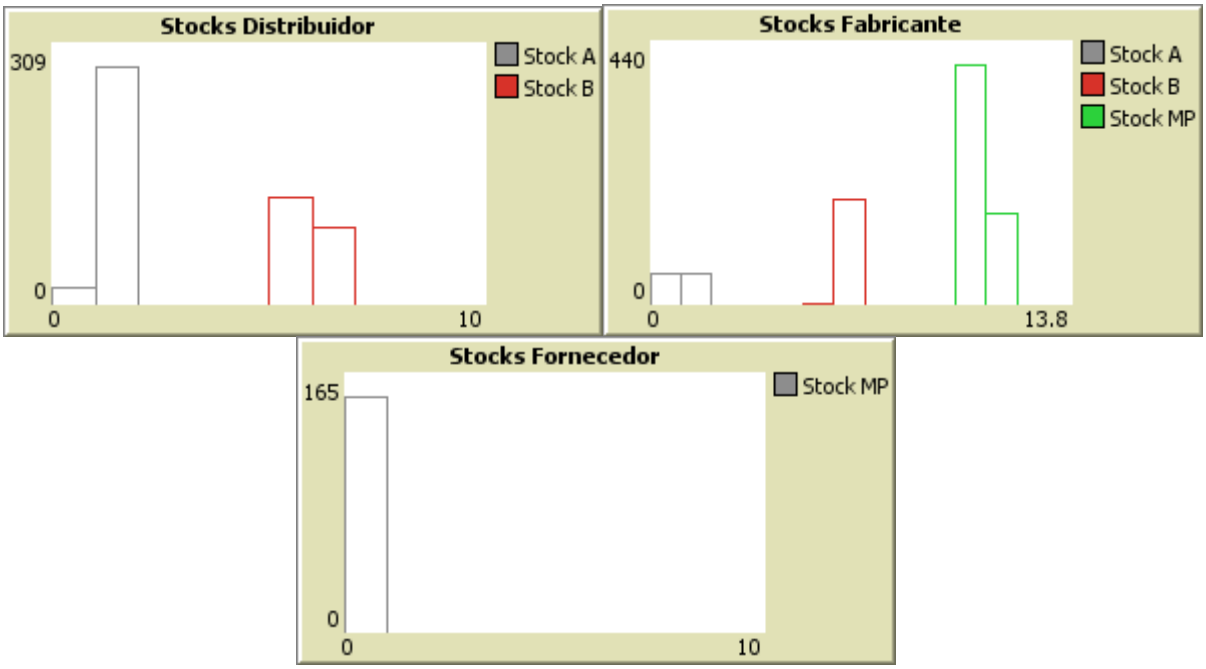

Figura 16 - Resultados gráficos do Caso I.

Analisando os resultados da Figura 16 e da Tabela 3, é possível considerar os níveis de stock nas diversas entidades como aceitáveis. Começando pelo agente fornecedor, tudo o que consegue armazenar de matéria-prima num dia é imediatamente vendido no dia seguinte. O mesmo acontece com os agentes fabricantes, tudo o que produzem do produto A é vendido no dia seguinte, ao contrário do produto B que mantém um stock razoável, assim como, o stock de matéria-prima de cada um deles. No caso dos agentes distribuidores, vão tendo um stock de produtos positivo e sem escassez.

Tabela 3 - Dados de Saída do Caso I.

Agentes	Dados de Saída		
	Stock A	Stock B	Stock MP
Distribuidor 1	20	126	-
Distribuidor 2	281	90	-
Fabricante 1	50	-	400
Fabricante 2	50	175	150
Fornecedor 1	-	-	150

No caso de num intervalo de tempo o número de encomendas de produto por dia ultrapassar a capacidade máxima de produção dos fabricantes, daria origem a uma rutura do stock de matéria-prima porque o fornecedor não teria capacidade suportar e abastecer esses gastos de matéria-prima, o que provocaria uma escassez de produtos no distribuidor.

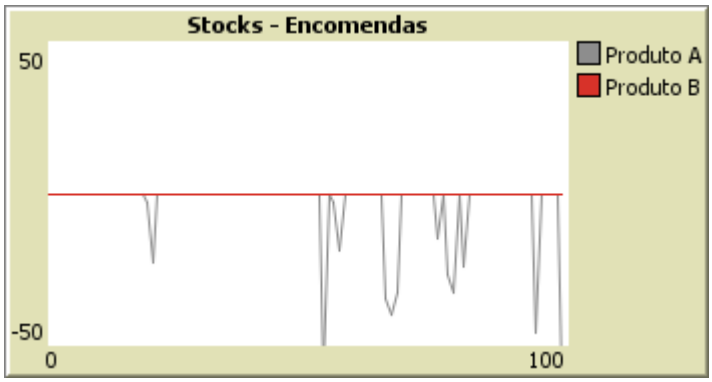

Figura 17 - Rutura do stock nos distribuidores.

Para os mesmos intervenientes do Caso I, foram testadas novas simulações, com a diminuição da capacidade máxima de produção dos fabricantes. O gráfico apresentado na Figura 17, representa a diferença entre o stock dos distribuidores e o número de encomendas ao longo da simulação.

No Caso II, a simulação compreende 100 dias e considera cinco distribuidores, cinco fabricantes e cinco fornecedores com os seguintes dados de entrada.

Tabela 4 - Dados de entrada do Caso II.

Distribuidor	Dados de Entrada
	Produto
1	B
2	B
3	A
4	A; B
5	B

Fornecedor	Dados de Entrada		
	Preço	Tempo	Matéria-Prima
1	8	2	140
2	11	2	140
3	8	2	140
4	8	2	140
5	11	1	140

Fabricante	Dados de Entrada						
	Produto	Preço A	Tempo A	Preço B	Tempo B	ProduçãoA	ProduçãoB
1	A	40	2	-	-	70	-
2	A	38	2	-	-	70	-
3	B	-	-	58	2	-	50
4	B	-	-	73	1	-	50
5	A; B	35	2	58	2	70	50

É possível observar na Tabela 4 os atributos de cada tipo de agente da simulação do Caso II. Nesta simulação há três agentes distribuidores que vendem apenas o produto B, um que vende apenas o produto A e um que vende os dois produtos. Com os agentes fabricantes, há

dois que produzem apenas o produto A e demoram 2 dias para entregar, dois que produzem apenas o produto B sendo que um demora 1 dia e o outro 2 dias para entregar e um que produz os dois produtos e que demora 2 dias para entregar. No caso dos fornecedores, há quatro agentes que demoram 2 dias a entregar a matéria-prima e um agente que demora apenas 1 dia. No Caso II em relação ao Caso I, aumentou-se a capacidade de produção das fábricas e diminui-se a capacidade máxima de matéria-prima que os fornecedores conseguem repor por dia.

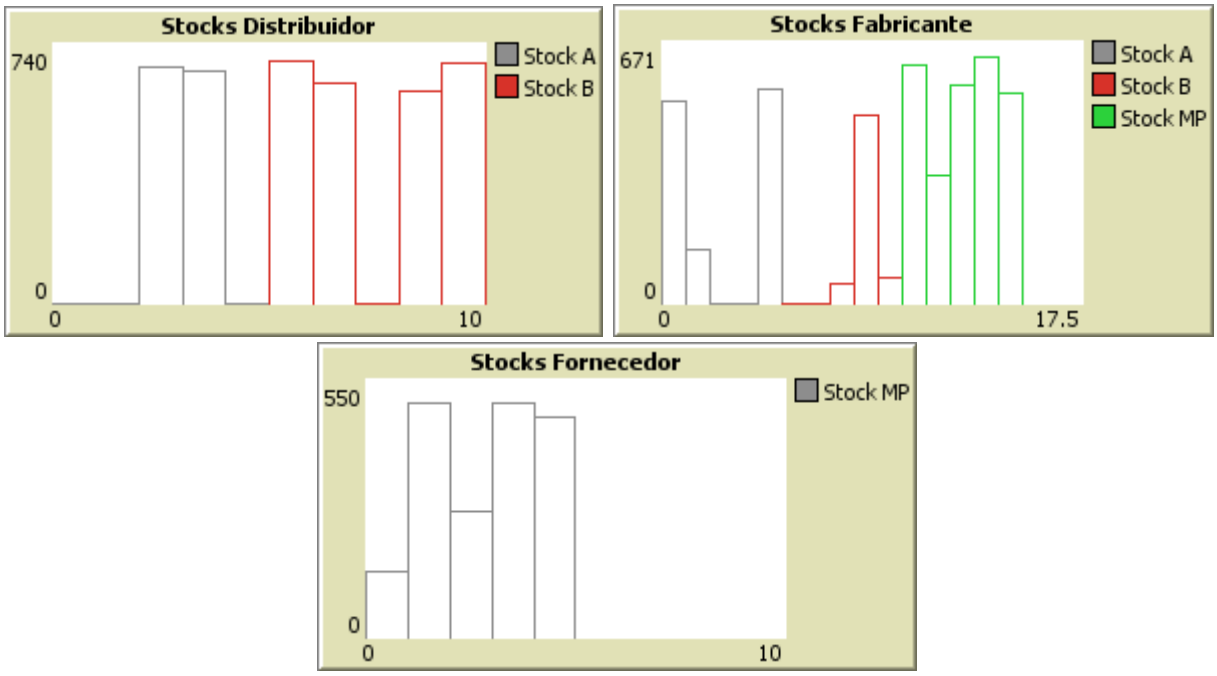

Figura 18- Resultados gráficos do Caso II.

Para o Caso II, analisando os resultados da Figura 18 e da Tabela 5, os distribuidores apresentam um stock positivo e sem escassez de produto. Por outro lado, apesar dos fabricantes apresentarem um stock positivo, comparativamente há fabricantes com stocks de produtos mais elevados e outros muito mais baixos, isto deve-se às diferenças de preços praticados por cada um dos agentes que também se faz notar no stock de matéria-prima, os que vendem mais precisam de mais matéria-prima para voltar a produzir. Entre os fornecedores, destaca-se o "fornecedor 1" por conseguir vender tudo o que tem em stock, isto porque, analisando este na Tabela 5, percebe-se que o tem em stock ao final do dia é equivalente à capacidade máxima de repor o stock de matéria-prima, sendo que as fabricas só recorrem a outros agentes depois do stock deste chegar a zero. Isto é justificado por este fabricante apresentar um menor preço nos produtos que produz.

Tabela 5 - Dados de Saída do Caso II.

Agentes	Dados de Saída		
	Stock A	Stock B	Stock MP
Distribuidor 1	-	686	-
Distribuidor 2	-	625	-
Distribuidor 3	673	-	-
Distribuidor 4	663	601	-
Distribuidor 5	-	681	-
Fabricante 1	516	-	610
Fabricante 2	140	-	330
Fabricante 3	-	50	560
Fabricante 4	-	480	630
Fabricante 5	550	67	540
Fornecedor 1	-	-	140
Fornecedor 2	-	-	500
Fornecedor 3	-	-	270
Fornecedor 4	-	-	500
Fornecedor 5	-	-	470

Utilizando o mesmo modelo, seria possível simular diferentes cenários e identificar e parametrizar a melhor configuração da cadeia de fornecimento tendo em vista os requisitos dos clientes.

A parte experimental do trabalho permitiu concluir que a utilização de plataformas de modelação baseada em agentes permite simular diferentes cenários, testar possíveis soluções e contribui para uma validação rápida de um caso de estudo, através da definição das melhores estratégias de controlo e gestão.

Capítulo 6

6 Conclusão

Este trabalho apresenta a simulação de um modelo baseado em agentes para o funcionamento de uma cadeia de abastecimento com múltiplos sectores, tendo por objetivo o estudo da aplicabilidade de mecanismos automáticos, dinâmicos e adaptativos na resolução de problemas complexos. Nesta simulação, os agentes tem como objetivo manter os níveis de stock dos materiais equilibrados, tendo para isso que adaptar o seu comportamento sempre que exista compra/venda de produtos, podendo optar sempre pelo melhor preço ou pelo menor tempo de entrega.

A plataforma NetLogo foi selecionada para desenvolver o modelo baseado em agentes para a cadeia de abastecimento, permitindo criar o modelo de uma forma modular, flexível e reconfigurável, assim como simular diferentes cenários de execução para a cadeia de fornecimento e verificar a forma como esta se comporta. O que diferencia este tipo de simulação das plataformas tradicionais é a sua capacidade de modelar e simular um SMA de forma fácil e compreensível, quer ao nível da programação simples e intuitiva, quer por permitir uma visualização clara dos resultados obtidos.

Analisando os resultados da simulação do modelo baseado em agentes, com o fluxo de informação é possível fazer um balanceamento do fluxo de produtos ao longo da cadeia, sendo possível determinar o que cada entidade deve fornecer ao longo do tempo. Ou seja, o sistema para uma determinada encomenda, de maneira a ter stock de produtos nos distribuidores, vai funcionar de forma a que os fornecedores satisfaçam as necessidades de matéria-prima dos fabricantes, para que estes sejam capazes de produzir e abastecer os distribuidores.

Como trabalho futuro seria interessante introduzir fenómenos económicos no modelo baseado em agentes, sendo que cada agente teria capital financeiro e na relação compra/venda não só efetuavam trocas de produto, como efetuariam transações monetárias, neste caso, se algum agente ficasse com o seu capital reduzido a zero seria excluído da simulação.

Referências bibliográficas

[Allan, 2009] Allan, R. J. (2009). Survey of agent based modelling and simulation tools.

[Babiceanu et al., 2005] Babiceanu, R.F.; Chen, F.F.(2005). Performance evaluation of agent-based material handling systems using simulation techniques. *In: Winter Simulation Conference. Florida.*

[Beamon, 1999] Beamon, B. (1999). Measuring Supply Chain Performance. *International Journal of Operational and Production Management, v. 19, n. 3, p:276.*

[Carvalho et al., 2005] Carvalho, R.; Luís Custódio, L. (2005). A Multiagent Systems Approach for Managing Supply-Chain Problems: new tools and results. *Inteligência Artificial V. 9, No 25.*

[Cavalieri et al.,2003] Cavalieri, S.; Cesarotti, V.; Introna, V. (2003). A Multiagent Model for Coordinated Distribution Chain Planning. *Jornal of Organizational Computing and Eletronic Commerce, v. 13(3-4), p. 267-287.*

[Coelho, 2007] Coelho, Hélder (2007). Modelação Computacional Baseada em Agentes: Enfrentar a Complexidade.

[Daniels, 2011] Daniels, M (2011). The swarm simulation system - a tool for studying complex systems.

[Eymann et al., 1998] Eymann, T.; Padovan, B.; Schoder, D. (1998). Simulating value chain coordination with artificial life agents. *Proceedings of the 3rd International Conference on Multi Agent Systems. Washington, DC, EUA: IEEE Computer Society. p. 423–424.*

[Fox et al., 1993] Fox, Mark S.; Chionglo, John F.; Barbuceanu, Mihai (1993). The Integrated Supply Chain Management System

[Fox et al., 2000] Fox Mark S.; Barbuceanu, Mihai; Teigen, R. (2000). Agent-oriented supply chain management. *The International Journal of Flexible Manufacturing Systems. [S.l.]: Kluwer Academic Publishers. v. 12, p. 165–188.*

[Hemant et al., 2012] Hemant B. Patil, P. A. P., Nitin G.Bagul (2012). Paradigms of simulation - A review. *World Journal of Science and Technology 2012.*

[Reis, 2002b] Reis, L. P. (2002). Agentes Autónomos. *Research Report: LIACC (NIAD&R).*

[Reis, 2002d] Reis, L. P. (2002). Coordenação de Agentes Competitivos: Negociação e Resolução de conflitos. *Research Report: LIACC (NIAD&R).*

[Reis, 2002e] Reis, L. P. (2002). Coordenação de Agentes Cooperativos e Trabalho de Equipa. *Research Report: LIACC (NIAD&R).*

[Reis, 2003] Reis, L. P. (2003). Coordenação em Sistemas Multi-Agente: Aplicações na Gestão Universitária e Futebol Robótico. *(PhD thesis), Faculdade de Engenharia da Universidade do Porto, Porto, Portugal.*

[Sarker et al., 2005] Sarker, R.; Kara, S.; Freeman, G.; Kayis, B.; Ray, T.; Abbass, H. (2005). A multi-agent simulation study for supply chain operation. *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce. Washington, DC, EUA: IEEE Computer Society. v. 1, p. 728–733.*

[Shannon, 1998] Shannon, Robert E. (1998). Introduction to the art and science of simulation. *Winter Simulation Conference.*

[Simchi-Levi et al., 2003] Simchi-Levi, David; Kaminsky, Philip (2003). Edith – Designing and managing the supply chain: concepts, strategies, and case studies. *2ª ed. Nova Iorque: McGraw-Hill/Irwin.*

[Souza et al., 2006] Souza, G.; Carvalho, M.; Liboreiro, M. (2006). Gestão da Cadeia de Suprimentos Integrada à Tecnologia da Informação. *Revista de Administração Pública vol. 40 no 4. Rio de Janeiro.*

[Stone et al., 1998] Stone, P.; Veloso, M. (1998). Towards collaborative and adversarial learning: a case study in robotic soccer. *Int. J. Hum.-Comput. Stud., 48(1), 83-104.*

[Wilensky, 1999] Wilensky, U. (1999). NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

[Wooldridge et al., 1995] Wooldridge, Michael; Jennings, Nicholas R. (1995). Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review, v.10, n.2, p.115–152.*

[Wooldridge, 2002] Wooldridge, M. (2002). An Introduction to MultiAgent Systems: John Wiley & Sons Australia, Limited.

Anexo A

A Código Fonte

```
turtles-own[id-number produtos]


breed [factory]
breed [industry]
breed [client]


client-own[encomendas_c_A  stock_c_A  vendas_c_A  time_c_A  vendas_t_c_A
encomendas_c_B stock_c_B vendas_c_B time_c_B vendas_t_c_B encomenda_ped1-A
encomenda_ped2-A  turno1-A  turno2-A  encomenda_ped1-B  encomenda_ped2-B
turno1-B turno2-B]


factory-own[capacity_pro_A  capacity_pro_B  price_A  price_B  time_A  time_B
capacity_pro_f  encomendas_f  stock_f_A  stock_f_B  stock_f_i  vendas_f_A
vendas_f_B turno_1 turno_2 encomenda_ped_1 encomenda_ped_2 time_fi]


industry-own[price time capacity_pro_i stock_i vendas_i]


globals[
  list_price_A
  list_price_B
  list_time_A
  list_time_B
  list_en_A
  list_en_B
  list_encomendas_A
  list_encomendas_B
  list_time_c_A
  list_time_c_B
  pedidos
```

encomenda_rec
token
stock_fi
stock_cc
vendas
wait_time
list_time_f_i
list_encomendas_f
ex_prod_i
ex_prod_f
comparator
k
list_time_f
list_time_i
list_time_c
list_new
max_price_f
p_max_price_f
m
resto
profit_f1
encomenda_mp
min_price_f
min_price_f2
min_price_i
min_price_i2
encomenda
list_price_f
list_price_i
p_min_price_f
p_min_price_i
list_encomendas
list_capacity_f
list_capacity_i
p_encomenda cap_f
cap_i
list_stock
l
list_vendas_f
list_en

```
  list_produtos
  ]

to setup
  set comparator 0
  create
end

to go
  if ticks = days [stop]
  encomends_client_A
  encomends_client_B
  prices
  negotiation_client_factory_A
  transport_factory_client_A
  negotiation_client_factory_B
  transport_factory_client_B
  production_A
  production_B
  encomendas_factory_industry
  negotiation_factory_industry
  transport_industry_client
  production_industry
  histo
  set comparator comparator + 1
  tick
end

to create
  ca
  reset-ticks
  set-default-shape client "shop"
  ask patches with [pycor = 8 and (member? pxcor x-locations cli)] [
    sprout-client 1 [
      set heading 0
      set color green
      set size 5
      set produtos random 3
      set vendas_c_A 0
      set encomendas_c_A 0           ; client ----> factory
```

```
      set time_c_A 0
      set vendas_t_c_A 0
      set vendas_c_B 0
      set encomendas_c_B 0           ; client ----> factory
      set time_c_B 0
      set vendas_t_c_B 0
      if produtos = 0 [
        set stock_c_A 400
      ]
      if produtos = 1[
        set stock_c_B 300
      ]
      if produtos = 2[
        set stock_c_A 400
        set stock_c_B 400
      ]
    ]
  ]

  set-default-shape factory "house"
  ask patches with [pycor = 0 and (member? pxcor x-locations fac)] [
    sprout-factory 1 [
      set list_price_A []
      set list_time_A []
      set list_price_B []
      set list_time_B []
      set heading 0
      set color red
      set size 5
      set produtos random 3
      set encomendas_f 0          ; factory ----> industry
      if produtos = 0 [
        set capacity_pro_A ProduçãoA
        set stock_f_A 200
        set time_A (1 + random 2)
        ;show time_A
        ifelse time_A = 1 [set price_A (40 + random 15) set list_price_A
lput price_A list_price_A set list_time_A lput time_A list_time_A]
        [set price_A (35 + random 6) set list_price_A lput price_A
list_price_A set list_time_A lput time_A list_time_A]
        set price_B 100
```

```
        set list_price_B lput price_B list_price_B set list_time_B lput 0
list_time_B
        set stock_f_i 400
        set vendas_f_A 0              ; factory ----> client
      ]
      if produtos = 1[
        set capacity_pro_B ProduçãoB
        set stock_f_B 200
        set time_B (1 + random 2)
        ;show time_B
        ifelse time_B = 1 [set price_B (60 + random 15) set list_price_B
lput price_B list_price_B set list_time_B lput time_B list_time_B]
        [set price_B (55 + random 6) set list_price_B lput price_B
list_price_B set list_time_B lput time_B list_time_B]
        set price_A 100
        set list_price_A lput price_A list_price_A set list_time_A lput 0
list_time_A
        set stock_f_i 400
        set vendas_f_B 0              ; factory ----> client
      ]
      if produtos = 2[
        set stock_f_A 200
        set stock_f_B 200
        set capacity_pro_A ProduçãoA
        set capacity_pro_B ProduçãoB
        set time_A (1 + random 2)
        ;show time_A
        ifelse time_A = 1 [set price_A (40 + random 15) set list_price_A
lput price_A list_price_A set list_time_A lput time_A list_time_A][set
price_A (35 + random 6) set list_price_A lput price_A list_price_A set
list_time_A lput time_A list_time_A]
        set time_B (1 + random 2)
        ;show time_B
        ifelse time_B = 1 [set price_B (60 + random 15) set list_price_B
lput price_B list_price_B set list_time_B lput time_B list_time_B][set
price_B (55 + random 6) set list_price_B lput price_B list_price_B set
list_time_B lput time_B list_time_B]
        set stock_f_i 700
        set vendas_f_A 0              ; factory ----> client
        set vendas_f_B 0              ; factory ----> client
      ]
      ;show price_A
    ]
  ]
```

```
  set-default-shape industry "industry"
  ask patches with [pycor = -8 and (member? pxcor x-locations ind)] [
    sprout-industry 1 [
      set list_time_i []
      set list_price_i []
      set heading 0
      set color blue
      set size 5
      set stock_i 500
      set list_time_i lput (1 + random 2) list_time_i
      ifelse last list_time_i = 2 [set list_price_i lput (7 + random 5)
list_price_i][set list_price_i lput (11 + random 5) list_price_i]
      set price last list_price_i
      set time last list_time_i
      set capacity_pro_i Matéria-Prima      ; produção
      set vendas_i 0              ; industry ----> factory
    ]
  ]

  let c_num 1
  foreach sort client[
    ask ? [
      set label word "Distribuidor " c_num
      set id-number c_num
      set c_num c_num + 1
    ]
  ]

  let f_num 1
  foreach sort factory[
    ask ? [
      set label word "Fabricante " f_num
      set id-number f_num
      set f_num f_num + 1
    ]
  ]

  let i_num 1
  foreach sort industry [
    ask ? [
```

```
      set label word "Fornecedor " i_num
      set id-number i_num
      set i_num i_num + 1
    ]
  ]
end

to-report x-locations [number]
  let x-list []
  let interval round ((2 * (max-pxcor - 3) + 1) / (number + 1))
  let current-pos (- max-pxcor + 3 + interval)
  repeat number [
    set x-list lput current-pos x-list
    set current-pos current-pos + interval
  ]
  report x-list
end

to encomends_client_A
  set list_en_A []
  set list_encomendas_A []
  set list_time_c_A []
  foreach sort client [
    ask ? [
      if (produtos = 0) or (produtos = 2) [
        set vendas_c_A (30 + random 60)
        if vendas_c_A >= stock_c_A [set vendas_c_A stock_c_A]
        set vendas_t_c_A vendas_t_c_A + vendas_c_A
        set list_en_A lput vendas_c_A list_en_A
        set stock_c_A stock_c_A - vendas_c_A
        ifelse vendas_c_A <= 70 [set encomendas_c_A vendas_c_A][set
encomendas_c_A 70]
        if stock_c_A > 600 [set encomendas_c_A 0]
        ifelse stock_c_A <= 200 [set time_c_A 1 set encomendas_c_A 70][set
time_c_A 0]
        set list_time_c_A lput time_c_A list_time_c_A
        set list_encomendas_A lput encomendas_c_A list_encomendas_A
      ]
    ]
  ]
  ;show list_en_A
```

```
  ;show list_time_c
  ;show list_encomendas
end



to encomends_client_B
  set list_en_B []
  set list_encomendas_B []
  set list_time_c_B []
  foreach sort client [
    ask ? [
      if (produtos = 1) or (produtos = 2) [
        set vendas_c_B (30 + random 40)
        if vendas_c_B >= stock_c_B [set vendas_c_B stock_c_B]
        set vendas_t_c_B vendas_t_c_B + vendas_c_B
        set list_en_B lput vendas_c_B list_en_B
        set stock_c_B stock_c_B - vendas_c_B
        ifelse vendas_c_B <= 50 [set encomendas_c_B vendas_c_B][set
encomendas_c_B 50]
        if stock_c_B > 600 [set encomendas_c_B 0]
        ifelse stock_c_B <= 100 [set time_c_B 1 set encomendas_c_B 50][set
time_c_B 0]
        set list_time_c_B lput time_c_B list_time_c_B
        set list_encomendas_B lput encomendas_c_B list_encomendas_B
      ]
    ]
  ]
  ;show list_en_B
  ;show list_time_c
  ;show list_encomendas
end


to production_A
  foreach sort factory [
    ask ? [
      if stock_f_A < 500 [
        if stock_f_i >= 0 [
        ifelse stock_f_i < capacity_pro_A * 2 [set stock_f_A stock_f_A +
round(stock_f_i / 2) set stock_f_i 0]
        [set stock_f_A stock_f_A + capacity_pro_A set stock_f_i stock_f_i -
(capacity_pro_A * 2)]
        ]
```

```
      ]
    ]
  ]
end


to production_B
  foreach sort factory [
    ask ? [
      if stock_f_B < 450 [
        if stock_f_i >= 0 [
          ifelse stock_f_i < capacity_pro_B * 2 [set stock_f_B stock_f_B +
round (stock_f_i / 2) set stock_f_i 0]
          [set stock_f_B stock_f_B + capacity_pro_B set stock_f_i stock_f_i
- (capacity_pro_B * 2)]
        ]
      ]
    ]
  ]
end


to encomendas_factory_industry
  set list_time_f_i []
  set list_encomendas_f []
  foreach sort factory [
    ask ? [
      if stock_f_i < 500 [
        ifelse produtos = 2 [
          ifelse stock_f_i < 300 [set encomendas_f 250 set list_time_f_i
lput 1 list_time_f_i set time_fi 1]
          [set encomendas_f 150 set list_time_f_i lput 0 list_time_f_i set
time_fi 0]]
        [ifelse stock_f_i < 200 [set encomendas_f 150 set list_time_f_i
lput 1 list_time_f_i set time_fi 1]
          [set encomendas_f 100 set list_time_f_i lput 0 list_time_f_i set
time_fi 0]]
        set list_encomendas_f lput encomendas_f list_encomendas_f
      ]
    ]
  ]
end
to production_industry
  foreach sort industry [
```

```
    ask ? [
      if stock_i + capacity_pro_i >= 500 [set ex_prod_i 500 - (stock_i + capacity_pro_i) ]   ; o que não foi produzido
      ifelse stock_i <= 400 [set stock_i stock_i + capacity_pro_i][set stock_i 500]
    ]
  ]
end


to prices
  set list_time_A []
  set list_price_A []
  foreach sort factory [
    ask ? [set list_time_A lput time_A list_time_A set list_price_A lput price_A list_price_A]
  ]


  set list_time_B []
  set list_price_B []
  foreach sort factory [
    ask ? [set list_time_B lput time_B list_time_B set list_price_B lput price_B list_price_B]
  ]
  set list_time_i []
  set list_price_i []
  foreach sort industry [
    ask ? [set list_time_i lput time list_time_i set list_price_i lput price list_price_i]
  ]
end


to negotiation_client_factory_A


  foreach sort client [
    ask ? [
      ;show stock_c_A
      set encomenda encomendas_c_A
      let c 3
      while [c != 0][
        ifelse time_c_A = 0 [
          if encomenda > 0 [
```

```
            set min_price_f min(list_price_A)                                   ;
caso o tempo de encomenda seja 0, escolhe o menor preço de produção

            set p_min_price_f position min_price_f list_price_A         ; ve
qual é a fabrica

            ask factory with [id-number = p_min_price_f + 1][           ; a
posição nas listas começa em 0, enquanto as fa

              ifelse encomenda <= stock_f_A[                                    ;
caso o stock da fabrica responder a todas as encomendas

                set       stock_f_A       stock_f_A       -       encomenda
; actualiza o stock das fabricas

                set      vendas_f_A      vendas_f_A      +       encomenda
; actualiza o total de vendas

                set vendas encomenda

                ifelse time_A = 2 [set encomenda_rec vendas][set stock_cc
vendas]

                set encomenda 0

              ][

              set resto encomenda - stock_f_A                                   ;
caso alguma fabrica nao tenha

              set stock_f_A 0                                                   ;
actualiza o stock das fabricas

              set   vendas_f_A   vendas_f_A   +   (encomenda   -   resto)
; actualiza o total de vendas

              set vendas encomenda - resto

              ifelse time_A = 2 [set encomenda_rec vendas][set stock_cc
vendas]

              set encomenda resto

              ]

              if stock_f_A = 0 [set list_price_A replace-item p_min_price_f
list_price_A 100]         ;caso o stock seja zero essa fabrica retira-se da
negociação

            ]

            if encomenda_rec > 0 [ifelse encomenda_ped1-A = 0[set
encomenda_ped1-A encomenda_rec set turno1-A (comparator + 1) set
encomenda_rec 0]

              [set encomenda_ped2-A encomenda_rec set turno2-A (comparator
+ 1) set encomenda_rec 0]

            ]

            set stock_c_A stock_c_A + stock_cc

            set encomendas_c_A encomenda

          ]][

        ask factory with[time_A = 1 and stock_f_A > 0][

          ifelse encomenda <= stock_f_A [                                ; caso
o stock da fabrica responder a todas as encomendas

            set stock_f_A stock_f_A - encomenda                                 ;
actualiza o stock das fabricas
```

```
            set vendas_f_A vendas_f_A + encomenda                          ;
actualiza o total de vendas
            set vendas encomenda
            ifelse  time_A  =  2  [set  encomenda_rec  vendas][set  stock_cc
vendas]
            set encomenda 0
          ][
          set resto encomenda - stock_f_A                                  ; caso
alguma fabrica nao tenha
          set stock_f_A 0                                                  ;
actualiza o stock das fabricas
          set vendas_f_A vendas_f_A + (encomenda - resto)                  ;
actualiza o total de vendas
          set vendas encomenda - resto
          ifelse time_A = 2 [set encomenda_rec vendas][set stock_cc vendas]
          set encomenda resto
          ]
          if  stock_f_A  =  0  [set  list_price_A  replace-item  p_min_price_f
list_price_A 100]
        ]
        if   encomenda_rec   >   0   [ifelse   encomenda_ped1-A   =   0[set
encomenda_ped1-A  encomenda_rec  set  turno1-A  (comparator  +  1)  set
encomenda_rec 0]
          [set encomenda_ped2-A encomenda_rec set turno2-A (comparator + 1)
set encomenda_rec 0]
        ]
        set stock_c_A stock_c_A + stock_cc
        set encomendas_c_A encomenda
          ]
        set c c - 1
      ]
    ]
  ]
end


to transport_factory_client_A
  foreach sort client [
    ask ? [
      if turno1-A = comparator [
        set stock_c_A stock_c_A + encomenda_ped1-A
        set encomenda_ped1-A 0
      ]
      if turno2-A = comparator [
```

```
        set stock_c_A stock_c_A + encomenda_ped2-A
        set encomenda_ped2-A 0
      ]
    ]
  ]
end

to negotiation_client_factory_B
foreach sort client [
    ask ? [
      ;show stock_c_B
      set encomenda encomendas_c_B
      let c 3
      while [c != 0][
        ifelse time_c_B = 0 [
          if encomenda > 0 [
            set min_price_f min(list_price_B)                              ;
caso o tempo de encomenda seja 0, escolhe o menor preço de produção
            set p_min_price_f position min_price_f list_price_B        ; ve
qual é a fabrica
            ask factory with [id-number = p_min_price_f + 1][           ; a
posição nas listas começa em 0, enquanto as fa
              ifelse encomenda <= stock_f_B[                               ;
caso o stock da fabrica responder a todas as encomendas
                set       stock_f_B       stock_f_B       -       encomenda
; actualiza o stock das fabricas
                set       vendas_f_B      vendas_f_B      +       encomenda
; actualiza o total de vendas
                set vendas encomenda
                ifelse time_B = 2 [set encomenda_rec vendas][set stock_cc
vendas]
                set encomenda 0
              ][
              set resto encomenda - stock_f_B                              ;
caso alguma fabrica nao tenha
              set stock_f_B 0                                              ;
actualiza o stock das fabricas
              set   vendas_f_B   vendas_f_B   +   (encomenda   -   resto)
; actualiza o total de vendas
              set vendas encomenda - resto
              ifelse time_B = 2 [set encomenda_rec vendas][set stock_cc
vendas]
              set encomenda resto
              ]
```

```
            if stock_f_B = 0 [set list_price_B replace-item p_min_price_f
list_price_B 100]         ;caso o stock seja zero essa fabrica retira-se da
negociação

          ]

          if encomenda_rec > 0 [ifelse encomenda_ped1-B = 0[set
encomenda_ped1-B encomenda_rec set turno1-B (comparator + 1) set
encomenda_rec 0]

            [set encomenda_ped2-B encomenda_rec set turno2-B (comparator
+ 1) set encomenda_rec 0]

          ]

          set stock_c_B stock_c_B + stock_cc

          set encomendas_c_B encomenda

        ]][

      ask factory with[time_B = 1 and stock_f_B > 0][

        ifelse encomenda <= stock_f_B [                              ; caso
o stock da fabrica responder a todas as encomendas

          set stock_f_B stock_f_B - encomenda                        ;
actualiza o stock das fabricas

          set vendas_f_B vendas_f_B + encomenda                      ;
actualiza o total de vendas

          set vendas encomenda

          ifelse time_B = 2 [set encomenda_rec vendas][set stock_cc
vendas]

          set encomenda 0

        ][

        set resto encomenda - stock_f_B                              ; caso
alguma fabrica nao tenha

        set stock_f_B 0                                              ;
actualiza o stock das fabricas

        set vendas_f_B vendas_f_B + (encomenda - resto)               ;
actualiza o total de vendas

        set vendas encomenda - resto

        ifelse time_B = 2 [set encomenda_rec vendas][set stock_cc vendas]

        set encomenda resto

        ]

        if stock_f_B = 0 [set list_price_B replace-item p_min_price_f
list_price_B 100]

      ]

      if encomenda_rec > 0 [ifelse encomenda_ped1-B = 0[set
encomenda_ped1-B encomenda_rec set turno1-B (comparator + 1) set
encomenda_rec 0]

        [set encomenda_ped2-B encomenda_rec set turno2-B (comparator + 1)
set encomenda_rec 0]

      ]

      set stock_c_B stock_c_B + stock_cc

      set encomendas_c_B encomenda
```

```
          ]
        set c c - 1
      ]
    ]
  ]
end

to transport_factory_client_B
  foreach sort client [
    ask ? [
      if turno1-B = comparator [
        set stock_c_B stock_c_B + encomenda_ped1-B
        set encomenda_ped1-B 0
      ]
      if turno2-B = comparator [
        set stock_c_B stock_c_B + encomenda_ped2-B
        set encomenda_ped2-B 0
      ]
    ]
  ]
end

to negotiation_factory_industry
  foreach sort factory [
    ask ? [
      set pedidos encomendas_f
      let c 3
      while [c != 0][
        ifelse time_fi = 0 [
          if pedidos > 0 [
            show pedidos
            set min_price_i min(list_price_i)                                    ;
caso o tempo de encomenda seja 0, escolhe o menor preço de produção
            set p_min_price_i position min_price_i list_price_i        ; ve
qual é a fabrica
            ask industry with [id-number = p_min_price_i + 1][          ; a
posição nas listas começa em 0, enquanto as fa
              ifelse encomenda <= stock_i [                              ;
caso o stock da fabrica responder a todas as encomendas
                set stock_i stock_i - pedidos                            ;
actualiza o stock das fabricas
```

```
                set vendas_i vendas_i + pedidos                          ;
actualiza o total de vendas
                set vendas pedidos
                ifelse time = 2 [set encomenda_rec vendas][set stock_fi
vendas]
                set pedidos 0
              ][
              set resto pedidos - stock_i                                ; caso
alguma fabrica nao tenha
              set stock_i 0                                              ;
actualiza o stock das fabricas
              set vendas_i vendas_i + (pedidos - resto)                  ;
actualiza o total de vendas
              set vendas pedidos - resto
              ifelse time = 2 [set encomenda_rec vendas][set stock_fi
vendas]
              set pedidos resto
              ]
              if stock_i = 0 [set list_price_i replace-item p_min_price_i
list_price_i 100] ;caso o stock seja zero essa fabrica retira-se da
negociação
            ]
            if encomenda_rec > 0 [ifelse encomenda_ped_1 = 0[set
encomenda_ped_1 encomenda_rec set turno_1 (comparator + 1) set
encomenda_rec 0]
              [set encomenda_ped_2 encomenda_rec set turno_2 (comparator +
1) set encomenda_rec 0]
            ]
            set stock_f_i stock_f_i + stock_fi
            set encomendas_f pedidos
          ]][
        ask industry with[time = 1 and stock_i > 0][
          ifelse pedidos <= stock_i [                                ; caso o
stock da fabrica responder a todas as encomendas
            set stock_i stock_i - pedidos                            ;
actualiza o stock das fabricas
            show stock_i
            set vendas_i vendas_i + pedidos                          ;
actualiza o total de vendas
            set vendas pedidos
            ifelse time = 2 [set encomenda_rec vendas][set stock_fi vendas]
            set pedidos 0
          ][
          set resto pedidos - stock_i                                ; caso
alguma fabrica nao tenha
```

```
          set stock_i 0                                                    ;
actualiza o stock das fabricas
          set vendas_i vendas_i + (pedidos - resto)                        ;
actualiza o total de vendas
          set vendas pedidos - resto
          ifelse time = 2 [set encomenda_rec vendas][set stock_fi vendas]
          set pedidos resto
          ]
          if stock_i = 0 [set list_price_i replace-item p_min_price_i
list_price_i 100]
        ]
        if encomenda_rec > 0 [ifelse encomenda_ped_1 = 0[set
encomenda_ped_1 encomenda_rec set turno_1 (comparator + 1) set
encomenda_rec 0]
          [set encomenda_ped_2 encomenda_rec set turno_2 (comparator + 1)
set encomenda_rec 0]
        ]
        set stock_f_i stock_f_i + stock_fi
        set encomendas_f pedidos
          ]
        set c c - 1
      ]
    ]
  ]
end

to transport_industry_client
  foreach sort factory [
    ask ? [
      if turno_1 = comparator [
        set stock_f_i stock_f_i + encomenda_ped_1
        set encomenda_ped_1 0
      ]
      if turno_2 = comparator [
        set stock_f_i stock_f_i + encomenda_ped_2
        set encomenda_ped_2 0
      ]
    ]
  ]
end
;////////////////////******Factory******\\\\\\\\\\\\\\\\\\\\
;------------------a fabrica k tiver menos vendas vai baixar o preço do
produto-------------------
```

```
;to new_prices
;  if novo_preço = True [
;     if (comparator = 50) or (comparator = 100) or (comparator = 150) or
(comparator = 200) [
;      let list_vendas []
;      foreach sort factory [
;              ask  ?[  set  list_vendas_f  lput  vendas_f_A  list_vendas_f]]
; cria uma lista com todas as vendas das fabricas
;      let p_min_vendas_f position min(list_vendas_f) list_vendas_f
;         set list_price_f replace-item p_min_vendas_f list_price_f ((item
p_min_vendas_f list_price_f) + 1)
;    ]
;  ]
;;-------------------------------------------------------------------------
------------------------
;
;end
to histo
  set-current-plot "Stocks Fornecedor"
  clear-plot
  set-current-plot "Stocks Fornecedor"
  foreach sort industry[
    ask ?[set-current-plot-pen "Stock MP" plotxy (id-number - 1) stock_i]]
  set-current-plot "Stocks Fabricante"
  clear-plot
  set-current-plot "Stocks Fabricante"
  foreach sort factory[
    ask ?[set-current-plot-pen "Stock A" plotxy (id-number - 1) stock_f_A]]
  foreach sort factory[
    ask ?[set-current-plot-pen "Stock B" plotxy 4 + id-number stock_f_B]]
  foreach sort factory[
    ask ?[set-current-plot-pen "Stock MP" plotxy 9 + id-number stock_f_i]]
  set-current-plot "Stocks Distribuidor"
  clear-plot
  set-current-plot "Stocks Distribuidor"
  foreach sort client[
    ask ?[set-current-plot-pen "Stock A" plotxy (id-number - 1) stock_c_A]]
  foreach sort client[
    ask ?[set-current-plot-pen "Stock B" plotxy 4 + id-number stock_c_B]]
end
```