Plataforma de Simulação em Repast Simphony para Análise da Divulgação de Novas Ideias em Mercado

Joaquim Margarido Passos de Sousa

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Arquiteturas, Sistemas e Redes

Orientadora: Doutora Isabel Praça

Coorientador: Doutor Pedro Campos

Júri:

Presidente:

Doutor José António Reis Tavares, DEI/ISEP

Vogais:

Doutora Maria Goreti Carvalho Marreiros, DEI/ISEP

Doutora Isabel Cecília Correia da Silva Praça Gomes Pereira, DEI/ISEP

Doutor Pedro José Ramos Moreira de Campos, Faculdade de Economia, Universidade do Porto

Porto, Novembro, 2013

Resumo

Modelação e simulação baseadas em agentes estão a ganhar cada vez mais importância e adeptos devido à sua flexibilidade e potencialidade em reproduzir comportamentos e estudar um sistema na perspetiva global ou das interações individuais.

Neste trabalho, criou-se um sistema baseado em agentes e desenvolvido em *Repast Simphony* com o objectivo de analisar a difusão de um novo produto ou serviço através de uma rede de potenciais clientes, tentando compreender, assim, como ocorre e quanto tempo demora esta passagem de informação (inovação) com diversas topologias de rede, no contato direto entre pessoas.

A simulação baseia-se no conceito da existencia de iniciadores, que são os primeiros consumidores a adotar um produto quando este chega ao mercado e os seguidores, que são os potenciais consumidores que, apesar de terem alguma predisposição para adotar um novo produto, normalmente só o fazem depois de terem sido sujeitos a algum tipo de influência.

Com a aplicação criada, simularam-se diversas situações com a finalidade de obter e observar os resultados gerados a partir de definições iniciais diferentes. Com os resultados gerados pelas simulações foram criados gráficos representativos dos diversos cenários.

A finalidade prática desta aplicação, poderá ser o seu uso em sala de aula para simulação de casos de estudo e utilização, em casos reais, como ferramenta de apoio à tomada de decisão, das empresas.

Palavras-chave: Simulação, sistemas multi-agente, redes colaborativas, divulgação de novas ideias, inovação.

Abstract

Modeling and agent-based simulation are increasingly gaining importance and supporters due to its flexibility and capability to reproduce behaviors and study a system in the global perspective of the individual interaction.

In this work, an agent-based system has been created and developed in Repast Simphony in order to analyze the diffusion of a new product or service through a network of potential customers while trying to understand how it happens and how long it takes to pass the information (innovation) among people using various network topologies, through direct contact among people.

The simulation is based on the concept of the existence of starters, which are the first consumers to adopt a product when it hits the market and the followers, who are potential consumers who, despite having some predisposition to adopt a new product, usually only do it after they have been subject to some sort of influence.

With the system, different situations were simulated in order to attain and observe the results generated from different initial settings. With the generated results through the simulations, graphics were created to express the results of the various scenarios.

The practical purposes of this application are its use in a classroom context for simulation of case studies, and its use in real cases as a tool to support decision making in companies.

Keywords: Simulation, multi-agent systems, collaborative networks, diffusion of new ideas, innovation.

Agradecimentos

Quero agradecer à Doutora Isabel Praça pela disponibilidade, pelo profissionalismo e pela paciência, porque não é fácil.

Também quero agradecer ao Doutor Pedro Campos pelo apoio e, acima de tudo, pelo incentivo de acreditar em mim.

Índice

Lista de código

Lista de tabelas

Lista de figuras

Acrónimos e Símbolos

Lista de Acrónimos

API	*Application Programming Interface*.
ABMS	*Agent-based modeling and simulation*.
XML	*Extensible Markup Language*
FIPA	*The Foundation for Intelligent Physical Agents*
Matlab	*Matrix Laboratory*
Repast	*Recursive Porous Agent Simulation Toolkit*
JADE	*Java Agent Development*
RS	*Repast Simphony*
AMES	*Wholesale Power Market Test Bed*
GIS	*Geographic Information System*

1 Introdução

Na área da Economia as simulações tornam-se de especial importância pelo facto de esta ser uma ciência social complexa e com a qual é difícil, às vezes mesmo impossível, realizar experiências em contexto real. Por exemplo, baixar o ordenado mínimo só para descobrir o que acontece, ou subir os juros dos empréstimos a longo prazo, para perceber se as consequências seriam positivas ou negativas, ou aumentar o desemprego, são experiências que não são viáveis.

O principal objetivo de uma empresa é a maximização de lucro [MICROECONOMIA II 1E108, URL]. O cumprimento deste objetivo determina o sucesso ou fracasso da empresa. Qualquer empresa para ter visibilidade no mercado tem que, habitualmente fazer um grande investimento em publicidade para ser conhecida e atrair novos clientes, o mesmo acontece para manter os clientes que já o são.

Quando uma empresa necessita de clientes, por norma tenta adquiri-los investindo em estratégias de *marketing*. Se a estratégia de *marketing* adotada, que pode ter um enorme peso nas suas finanças, não for a melhor, a empresa pode entrar em rutura financeira, até porque para financiar as suas estratégias de *marketing* a empresa precisa de faturar e isso só é possível se tiver clientes.

A ferramenta desenvolvida, no âmbito deste trabalho, poderá ser uma ajuda significativa na redução de gastos nestas estratégias porque, os resultados por ela gerados, permitem às empresas melhor compreender o efeito da passagem de informação através da comunicação direta entre pessoas, e assim, ajudá-las a decidir qual a melhor estratégia a adotar para a divulgação dos seus produtos.

Com esta ferramenta as empresas podem estudar quais as tendências de um determinado público-alvo e tentar compreender a melhor maneira de chegar a esse público, usufruindo, assim, de informação que poderá contribuir para a definição de uma estratégia de *marketing* mais incisiva e eficaz.

Esta ferramenta, baseada em agentes, permite simular a passagem de informação através da comunicação direta entre pessoas, simulando diversos tipos de redes sociais como a rede em malha e em estrela, mostrando, ao mesmo tempo e graficamente, a evolução da passagem dessa informação entre os agentes e mostrando, também, os agentes que vão aderindo ao novo produto. A simulação é baseada no princípio da existência de consumidores iniciais (chamados iniciadores) e potenciais novos consumidores (chamados seguidores), estes com alguma predisposição a aderir ao novo produto se forem influenciados, o que poderá ocorrer de diversas formas, como por exemplo, por recorrência ou por empatia com a pessoa que o influencia. No fim de cada simulação, poderá ser consultado um gráfico e um ficheiro com o

historial temporal da simulação, que ilustra quanto tempo cada seguidor demorou a aderir ao produto.

2 Ambientes de Simulação Multi-Agente

Neste capítulo vão ser apresentadas algumas das primeiras aplicações baseadas em sistemas multi-agente mais relevantes seguindo-se uma definição dos sistemas multi-agente e a descrição de algumas plataformas e ambientes mais conhecidos.

2.1 Evolução da simulação

As primeiras aplicações baseadas em Sistemas Multi-agente apareceram em meados dos anos 80 e foram, de uma forma crescente, cobrindo mais áreas como manufatura, controlo de processo, controlo de trafego aéreo e gestão de informação [Sycara, *1998*].

Uma das primeiras aplicações multi-agente é assim descrita por [Parunak, 1987]: uma empresa de manufatura é modelada como um conjunto de células de produção hierarquicamente organizadas. Coletivamente, estas células representam o complexo produtivo. Cada fábrica e cada componente produtivo são representados como um agente.

Outra aplicação para Sistemas Multi-agente destinava-se à monitorização distribuída de veículos [Durfee and Lesser, 1991] onde os agentes distribuídos geograficamente monitorizavam os veículos que passavam na sua área numa tentativa de interpretar a forma como o trafego se distribuía globalmente, e seguir os movimentos dos veículos.

Um sistema multi-agente muito conhecido é o ARCHON [Jennings, et al., 1995] que também pode ser usado como base para a criação de novas aplicações. O sistema foi usado em várias aplicações de controlo de processos, incluindo a gestão do transporte de eletricidade e controlo de acelerador de partículas. Outros sistemas de controlo de processos baseados em agentes foram criados para monitorizar e diagnosticar falhas em centrais nucleares [Wallace, et al., 2011].

Um outro exemplo de aplicação é o sistema multi-agente para controlo de tráfego aéreo, conhecido por OASIS, descrito por [Ljunberg and Lucas, 1992]. Sempre que uma aeronave entre no espaço aéreo é-lhe atribuído um agente e inicializado com os objetivos do mundo real para essa aeronave. Os agentes de controlo de tráfego aéreo são responsáveis por gerir o sistema.

Estas são algumas das primeiras aplicações relevantes, baseadas em sistemas multi-agente, que se tornaram das mais conhecidas, de acordo com a literatura.

2.2 Sistemas multi-agente

Um sistema multi-agente é compreendido como um sistema que inclui as seguintes características [Ferber, 1999]:

- Um ambiente, o espaço, que normalmente tem um volume;
- Um conjunto de objetos posicionados no ambiente e que podem ser reconhecidos, criados, destruídos e modificados pelos agentes;
- Um conjunto de agentes, que são objetos específicos e representam as entidades cativas do sistema;
- Um conjunto de relações, que ligam os objetos (e consequentemente os agentes) entre si;
- Um conjunto de operações, que permite aos agentes reconhecer, produzir, consumir e manipular os objetos;
- Operadores, com a tarefa de representar estas operações e as reações do mundo a esta tentativa de modificação.

Agentes são todos os elementos que interagem e tomam decisões em qualquer situação, seja ela real ou simulada [North and Macal, 2007] ou, segundo [Wooldridge, 2002], agentes são simplesmente sistemas computacionais capazes de ações autónomas em determinados ambientes com o fim de atingirem os seus objetivos designados.

Na definição de agentes esperam-se determinadas características, entre elas [North and Macal, 1998]:

- Capacidade de adaptação: faz com que o agente se comporte de acordo com a informação que vai obtendo do meio envolvente;
- Capacidade de aprender e modificar o seu comportamento: capacidade de usar a informação que obtém do meio envolvente para afetar o seu comportamento;
- Autonomia: Capacidade de executar algumas decisões de forma autónoma;
- Heterogeneidade, resultando numa população de agentes com diversas características.

Os agentes podem ser complexos ou baseados em regras simples [North and Macal, 1998].

Os agentes baseados em regras simples têm algumas vantagens sobre os complexos, tais como rápido desenvolvimento de modelos permitindo uma maior concentração no problema principal, menor tempo de verificação e validação de regras, permitindo uma mais rápida produção de resultados assim como uma maior facilidade na reutilização do modelo, uma vez que não será muito difícil ajustar as regras para uma nova simulação.

Os agentes complexos recorrem a um conjunto mais sofisticado de regras e a técnicas de modelação mais avançadas assim como, a informação gerada dessas simulações tende a ser mais completa.

Segundo [Ferber, 1999], existem dois tipos de agentes que deram origem a duas escolas de pensamento distintas: os agentes cognitivos e os agentes reativos.

Nos modelos construídos baseados em agentes cognitivos cada agente tem uma base de conhecimento definida que lhe dá todos os conhecimentos e informação necessários para a sua interação com os outros agentes, dando-lhes, assim, a conotação de inteligentes.

Estes agentes também são chamados de intencionais, pois têm objetivos a atingir sendo levados a negociar com outros agentes, tal e qual como as pessoas ou empresas na vida real que são levadas a negociar com outras entidades para atingir os seus objetivos.

Em contraste a escola que defende os agentes reativos acha que não há necessidade de agentes inteligentes. Este resultado pode ser obtido através de mecanismos de reação a acontecimentos sem considerar qualquer plano para atingir os objetivos nem mecanismos de planeamento.
Estes tipos de agentes deram origem a alguns subtipos como por exemplo:

- O BDI (*Belief, desire and Intention* – Convicção, desejo e intenção) que aplica o modelo de inteligência humana como conhecimento, vontade, escolha, desejo e empenho em que:
    - Convicções são factos que representam o que um agente acredita sobre o mundo, ou seja, a representação, através de um agente, do estado do mundo. [Milind Tambe, URL].
    - Os desejos são objetivos ou alguns estados finais desejados. Um agente pode ter vários desejos, que podem, eventualmente, estar em conflito [Milind Tambe, URL].
    - Vontades referem-se tanto aos compromissos do agente em relação aos seus desejos (objetivos) como ao seu compromisso com os planos selecionados para atingir esses objetivos. As vontades não podem entrar em conflito umas com as outras - elas têm que ser consistentes. Assim, um agente pode ter intenções apenas para um subconjunto dos seus desejos [Milind Tambe, URL].
    - A chave para a abordagem BDI é que se concentra no raciocínio em agentes de recursos limitados [Milind Tambe, URL].
- O racional, que atua de maneira a obter o máximo de êxito na tarefa que lhe é designada [Maalal and Addou, 2011].
- O adaptativo, que tem a capacidade de alterar os seus objetivos quando a base de conhecimento muda [Maalal and Addou, 2011].

- O comunicativo, que comunica a sua informação a outros agentes [Maalal and Addou, 2011].

Alguns agentes podem reunir mais do que uma destas características em simultâneo.

A tomada de decisão entre agentes, tipicamente, implica a aprendizagem e adaptação desses agentes. As decisões têm que ser tomadas numa base de conhecimento e variam conforme as definições dos agentes. Essas definições chamam-se atributos. A tomada de decisão de um agente, normalmente influencia a tomada decisão de outro agente, criando assim uma dependência entre eles.

Os atributos de cada agente que condicionam as suas decisões podem variar em complexidade. Por exemplo, o sexo ou a cor dos olhos nunca alteram, mas o seu rendimento mensal ou a idade poderão sofrer alterações com o tempo. Por outro lado as preferências tendem a ser mais complexas apesar de algumas, praticamente, não sofrerem alterações ao longo da vida, tais como preferências clubistas ou até partidárias. Algumas dessas preferências podem ser muito mais complexas tais como a preferência por roupa, por determinado filme ou livro, ou ainda mais complexo, por determinada pessoa.

O comportamento dos agentes pode ser baseado nos seus atributos e de forma a refletir a diversidade da situação (cada caso implica um determinado tipo de comportamento).

Existem essencialmente dois níveis de regras para os agentes [Casti, 1998]: de primeiro e segundo nível. As de primeiro nível são regras-base e que especificam como o agente reage a acontecimentos de rotina. As regras de segundo nível permitem alterar as regras de primeiro nível [Casti, 1998]. Segundo [Casti, 1998], os agentes têm regras e regras para alterar as regras [Casti, 1998]. Estas regras determinam como os agentes comunicam entre si, como compreendem o ambiente e como decidem.

Exemplo prático: um cliente entra numa loja para comprar um determinado produto e dirige-se ao balcão para falar com o vendedor. Nesta situação temos dois agentes, o cliente e o vendedor. O vendedor irá tentar fazer uma boa venda e o comprador uma boa compra. Os atributos de um irão influenciar a atitude do outro tentando cada um deles atingir o seu objetivo.

Atributos relevantes para cada um dos agentes:

Tabela 1 - Atributos do vendedor e do cliente

Vendedor	**Cliente**
Apresentação	Apresentação
Simpatia	Simpatia
Capacidade de compreender as necessidades do cliente	Capacidade de explicar as suas necessidades
Capacidade de expressão	Orçamento disponível
Domínio do assunto	Informação Prévia (Preparação)

Com estes atributos, os dois agentes irão interagir, tentando dar o seu melhor para tirarem o melhor partido da situação, e vão ganhando conhecimento ao longo dessa interação aprendendo com a informação que o outro agente vai passando para o exterior.

Este exemplo, muito simples, diz respeito a apenas dois agentes, mas é possível ter situações mais complexas, com diversos agentes com características muito diversas, a interagirem entre si.

Os atributos quantitativos não põem grandes problemas quando influenciam a tomada de decisão (idade, rendimento, altura, etc.). Os qualitativos são, sem dúvida, os mais complexos de definir. Por exemplo, como definir o domínio do assunto do vendedor ou a sua simpatia. Para se saber se o domínio do assunto do vendedor é grande ou pequeno ter-se-á que criar uma escala, mas o número que lhe é atribuído nessa escala tem que se basear em alguma informação.

Para se decidir até que ponto o vendedor domina o assunto é necessário analisar os seus conhecimentos (o que não é muito diferente de uma situação real). Para isso, uma boa forma é ver a quantas questões o vendedor responde, o seu grau de segurança na resposta e o valor da própria resposta, isto é, se a resposta nos satisfaz como tal. Este caso já envolve uma definição mais complexa do agente. Neste caso, o valor de um atributo ter-se-á que basear nos valores de outros. Em resumo, os valores de alguns atributos apenas podem ser apurados durante a simulação e mediante análise de outros atributos, ao contrário de outros casos em que podem ser definidos os seus valores antes de iniciar a simulação.

Quando os agentes comunicam entre si de uma forma organizada podem ser um potente e sofisticado sistema de simulação e apoio à decisão.

As tomadas de decisão, em certos casos, podem ser de tal complexidade devido ao número de variáveis a tomar em consideração que apenas recorrendo a ferramentas informáticas é que podemos ter um resultado fiável.

Alguns exemplos da utilização de sistemas multi-agente para tomadas de decisão são usados no nosso dia-a-dia, como por exemplo [Moreno, 2010]:

- O controlo de semáforos. O sistema central de controlo de semáforos pode perceber que o trânsito numa determinada via intensificou ao ponto de haver necessidade de ter o semáforo mais tempo aberto e consequentemente os das vias adjacentes mais tempo fechados;

- Controlo de velocidade de cruzeiro;

- Piloto automático de um avião;

- Automóvel sem condutor que a Google vem já há algum tempo desenvolvendo e aperfeiçoando [Forbes, URL].

Para serem considerados úteis, os agentes devem fornecer duas coisas [Jennings and Wooldrige, 1995]:

- Capacidade de resolver problemas que estão para além da automação ou porque a tecnologia existente é incapaz de resolver o problema ou por ser demasiado cara (difícil, demorada ou arriscada);

- Capacidade de resolver problemas para os quais já existe uma solução mas de forma significativamente melhor (mais barata, mais natural, mais fácil, mais eficiente ou mais rápida).

O número de exemplos em que são usados sistemas multi-agente cresce de dia para dia e estão presentes num enorme número de áreas de atividade que requerem tomadas de decisão rápidas e baseadas em múltiplos fatores complexos que tornam o trabalho difícil, senão impossível, para o ser humano. Alguns exemplos são:

- **Missões de exploração espacial**. A NASA está a investigar uma forma de tornar as missões de exploração espacial mais autónomas, tornando-as mais autossuficientes na tomada de decisões [Seah, et al., 2011].

- **Sociologia**. Os sistemas multi-agente são uma ferramenta relevante na simulação de sociedades [Davidsson, 2000].

- **Economia experimental**. Os sistemas multi-agente podem simular fenómenos micro e macroeconómicos [Heckbert, 2009].

- ***E-medicine, e-diagnosis***. Pode ser usado para simular atividades médicas específicas tais como diagnósticos [Hussain and Wood, 2009].

- **Internet**. É usado como leitor de noticias, com *browsers (google),* leitores de *email* [Dawid, 1999].

- **Viagens**. Planeamento de percursos [Wood, 1990].

Estas são apenas algumas áreas onde as simulações multi-agente se estão a tornar cada vez mais populares até porque existe uma enorme quantidade de *software* de simulação baseado em agentes, algum *Open Source* e outro proprietário que permite o desenvolvimento deste tipo de aplicações.

2.3 Plataformas para simulação

As ferramentas para simulação são muitas e de diversos níveis de complexidade [North and Macal, 1998].

As mais simples são:

- Sistemas matemáticos computacionais
- Ambientes dedicados a prototipagem

2.3.1 Sistemas matemáticos computacionais

Sistemas matemáticos computacionais *(Computational Mathematics Systems)* são alternativas a folhas de cálculo e podem ser usados para modelação de sistemas. Uma das suas principais caracteristicas é o facto de serem especializadas em modelos matemáticos, podendo lidar com simulações baseadas em matemática bastante mais complexa do que os sistemas baseados em folhas de cálculo.

2.3.1.1 MATLAB

O MATLAB (nome que vem da nomenclatura *Matrix Laboratory*) é um ambiente de programação para desenvolvimento de algoritmos, análise de informação, visualização e computação numérica [Matlab, URL]. Este software é poderoso na sua diversidade de capacidades mas com o inconveniente de ter uma linha de aprendizagem pouco acentuada, havendo necessidade de o seu utilizador, à partida, ter conhecimentos de matemática que podem ir do razoável até ao especializado para além de ter que aprender a sua linguagem de programação, o que poderá não ser tarefa fácil. Uma das vertentes positivas prende-se com o facto de ser fácilmente escalonável e na possibilidade de produzir informação em diversos formatos que pode, mais tarde, ser analisada por ferramentas especializadas.

Podemos ainda integrar *add-ons* com a aplicação, especializando ainda mais as suas funcionalidades, tornando-a numa poderosíssima ferramenta de prototipagem de modelos baseados em agentes, de grande rapidez no cálculo de funções matemáticas complexas. Apesar de toda a sua complexidade não deixa de ser muito mais fácil de criar um modelo recorrendo às suas funcionalidades do que recorrendo a linguagens de programação complexas como o C++ ou o Fortran. A tudo isto juntamos um resultado gráfico bastante apelativo e diversificado.

2.3.1.2 Mathematica

O Mathematica [Mathematica, URL] foi idealizado por Stephen Wolfram e é o único sistema computacional de cálculo completamente integrado no mundo. Além das características do MATLAB o *Mathematica* inclui também computação simbólica.

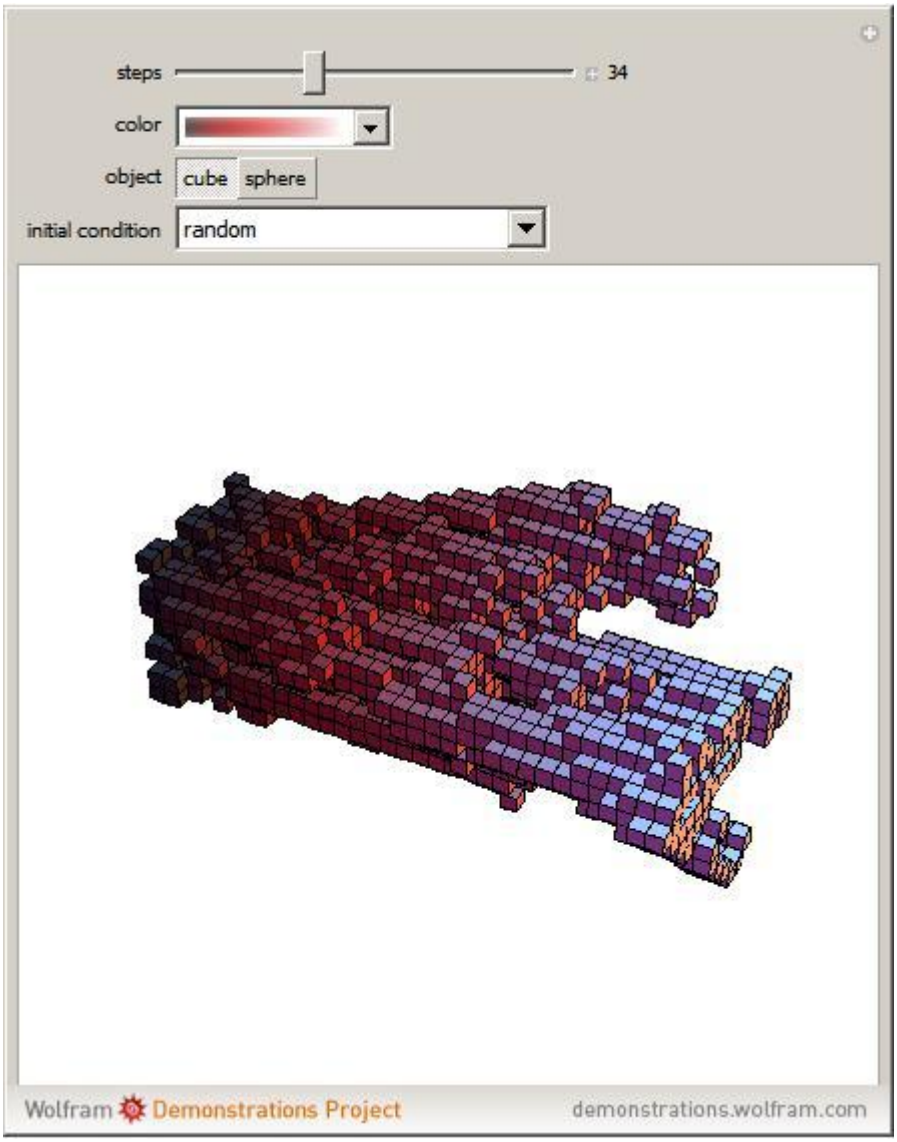


Figura 1 - Game of life in 3D layers [Mathematica, URL]

2.3.2 Ambientes dedicados a prototipagem

Muitos são os ambientes dedicados a prototipagem *(Dedicated ABMS Prototyping Environments)*. Não se pretende fazer uma análise exaustiva dos ambientes existentes mas apenas dar alguma relevancia a alguns dos mais utilizados na prática. Na secção 2.3.3 será dado especial destaque ao *Repast Simphony*, por ter sido a plataforma utilizada no desenvolvimento da aplicação de simulação de redes colaborativas proposta nesta Tese.

2.3.2.1 NetLogo

NetLogo [NetLogo, URL] é um ambiente para simulações multi-agente e foi desenvolvido em Java, é completamente grátis e conta com milhares de utilizadores. O facto de correr sobre o *Java Virtual Machine* em muito contribui para a sua popularidade, uma vez que pode ser executado sobre os principais sistemas operativos, como por exemplo *Windows*, *Mac* ou *Linux*. É utilizado por professores e alunos das mais diversas áreas científicas como matemática, biologia, economia, ciências sociais, medicina, física e química, isto para referir algumas. Inclui uma ferramenta chamada *HubNet* para simulação participativa em que uma simulação pode ter participantes em tempo de execução real e cada um controla um agente, recorrendo a uma rede de computadores ou mesmo através de calculadoras gráficas *Texas Instruments*. NetLogo é completamente programável e vem com inúmeros modelos de exemplo o que o torna atrativo para quem quer aprender a usar a aplicação, tendo uma curva de aprendizagem acentuada. A sua versatilidade abre possibilidades para simulações simples ou complexas, visualização de modelos em 2D ou 3D. A acrescentar a estas características está um grande conjunto de documentação disponível.

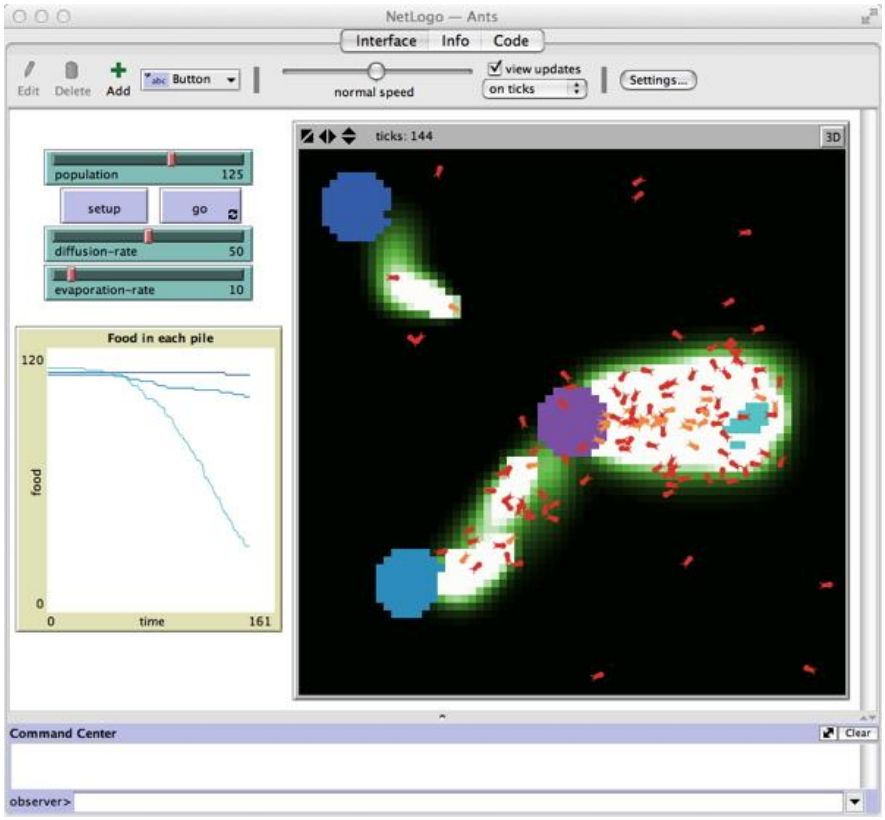


Figura 2 - Simulação com o NetLogo [NICO, URL]

2.3.2.2 StarLogo

É uma ferramenta poderosa para simulação de ambientes descentralizados, isto é, ambientes que surgem da interacção de diversos objectos. A criação de ambientes simulados foi simplificada ao ponto de não ser necessário, ao utilizador, ter conhecimentos profundos de

matemática teórica nem conhecimentos avançados de programação, fornecendo, assim, a possibilidade de criação de simulações de ambientes sofisticados sem exigir grande esforço ao utilizador. Neste momento a última versão de StarLogo chama-se *StarLogo TNG (The Next Generation) versão 1.0* [StarLogo, URL]e foi lançado em Julho de 2008 [StarLogo, URL]. Esta nova versão, além de manter todas as caracteristicas do StarLogo original, acrescenta uma programação gráfica e um mundo 3D. A versão actual está escrita em C e Java e corre em qualquer sistema. O StarLogo foi desenvolvido para fins educativos pelo *MIT – Massachusetts Institute of Technology* e é grátis.

O StarLogo também pode ser usado para criação de jogos educativos, sem necessitar de grande esforço da parte do utilizador, pelo facto de ter um *interface* gráfico baseado no sistema *drag and drop.*

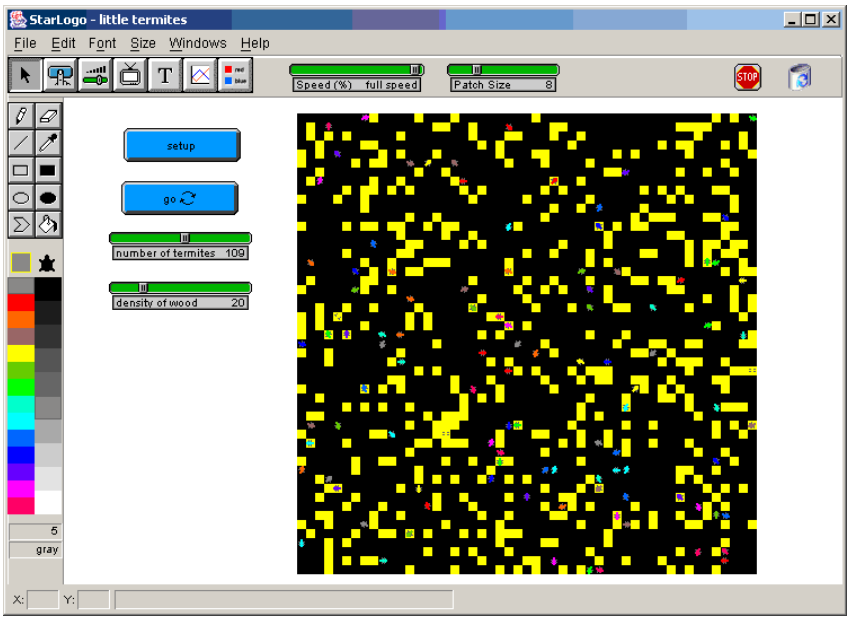


Figura 3 - Little termites [StarLogo, URL]

2.3.2.3 Swarm

O *Swarm* é um conjunto de bibliotecas que definem uma estrutura para a criação de sistemas de simulação multi-agente que se traduz numa *framework* dentro da qual os modelos podem ser construídos [Swarm, URL].

O *Swarm* insere neste tipo de *software* o conceito de enxame (*swarm*), um conjunto de agentes com uma definição de atividades e comportamentos.

O *enxame* é o elemento estrutural básico de um modelo. Numa simulação podemos criar vários *enxames* podendo desta forma criar simulações mais complexas e de multinível sendo possível, desta forma, definir agentes que criam o seu próprio mundo.

As bibliotecas do *Swarm* são criadas em "Objective-C" e as suas principais características são:

- Orientadas a objetos;
- Hierarquia, em que os agentes podem ser compostos por grupos de outros agentes pertencentes a outra estrutura;
- Disponibilização de ferramentas que facilitam a criação de simulações na forma de código.

O *Swarm* é uma ferramenta cujo número de adeptos tem vindo a aumentar significativamente.

2.3.2.4 JADE

Java Agent DEvelopment Framework (JADE) [JADE, URL] é um *framework* implementado em Java para a implementação de simulações baseadas em multi-agentes conforme as normas da FIPA (*The Foundation for Intelligent Physical Agents*).

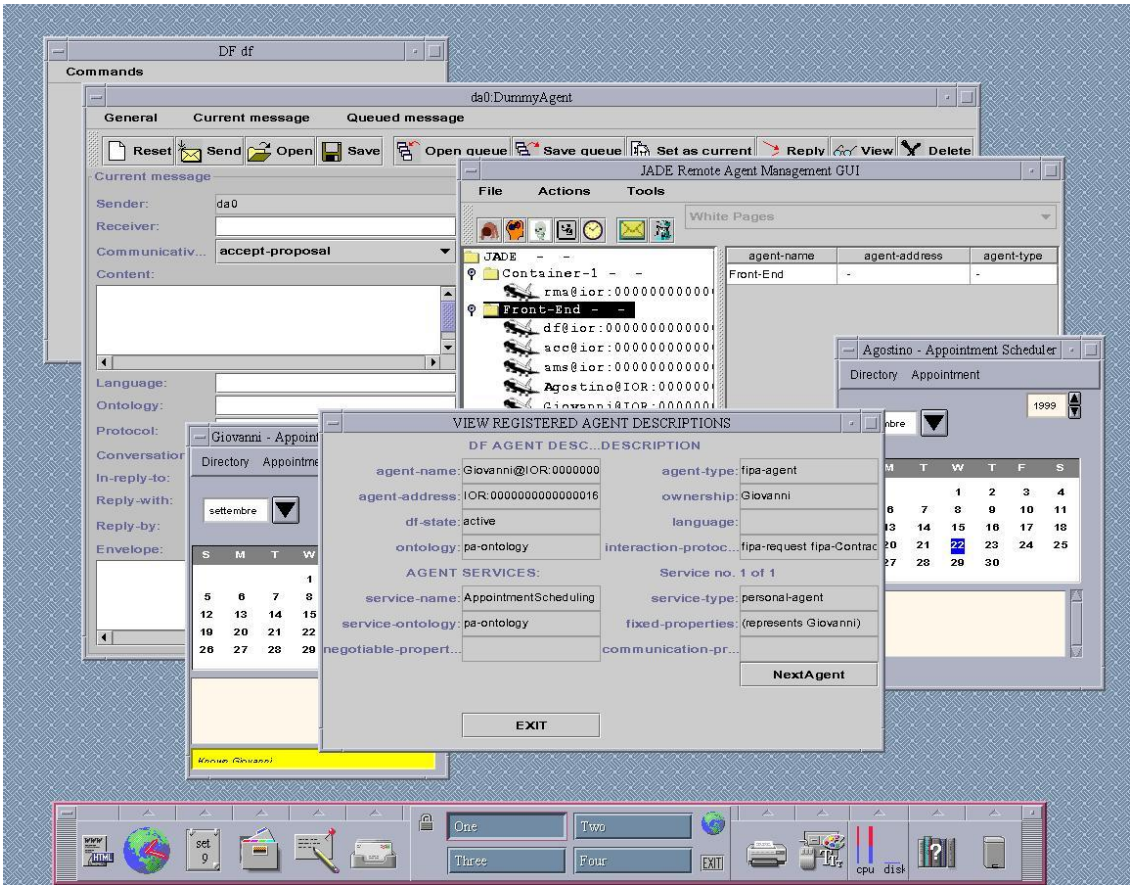


Figura 4 - O JADE (Java Agent Development) [JADE, URL]

O JADE é desenvolvido em JAVA e consiste em vários pacotes de bibliotecas. JADE é grátis, é distribuído pela *Telecom Italia* e é *open source* segundo os termos da LGPL (*Lesser General Public License Version 2*). A última versão é *JADE 4.3.0***,** lançada em 29/03/2013.

A plataforma pode ter vários *containers* onde residem os agentes que podem não estar, necessariamente, na mesma máquina. Cada máquina tem um JVM (*Java Virtual Machine*) e cada agente tem uma *thread.*

2.3.3 Repast Simphony

Vai dar-se especial destaque a este ambiente dado ser o ambiente escolhido para o desenvolvimento da aplicação apresentada neste trabalho. O *"Recursive Porous Agent Simulation Toolkit"* (*Repast*) [Repast Simphony, URL] é um ambiente de simulação para modelação baseada em agentes que comporta uma série de bibliotecas especialmente escritas para este efeito. A sua utilização, cada vez maior, em diversas áreas científicas e tecnológicas, é consequência da sua enorme flexibilidade e do facto de ser *open-souce*. Começou por ser um *front-end* para o *Swarm* mas quando a equipa que desenvolve o *Swarm* lançou o seu próprio API a equipa do *Repast Simphony* resolveu criar um API independente e a partir daí enveredar por um caminho independente do *Swarm.*

O *Repast Simphony* foi originalmente desenvolvido por David Sallach, Nick Collier, Tom Howe, Michael North, entre outros, da Universidade de Chicago. Esta ferramenta, nas suas diversas versões, corre em ambiente *Microsoft Windows, Linux e Apple Mac OS X* e existe em várias linguagens de programação.

Inicialmente o *Repast Simphony* era implementado apenas em JAVA (*Repast J*) sendo mais tarde adaptado para Python (*Repast Py*) e para compiladores baseados no Microsoft .NET Framework (*Repast .NET*) sendo por isso possível satisfazer uma diversa preferência por linguagens de programação.

O *Repast Simphony* proporciona a possibilidade de integrar na simulação algoritmos genéticos e redes neurais, através da utilização das bibliotecas *Java Genetic Algorithms Package (JGAP)* e o Java *Object Oriented Neural Engine (Joone).*

Não se vão descrever todas as possibilidades oferecidas pelo Repast *Simphony,* que são inúmeras, mas vão-se descrever algumas das suas características mais interessantes e que tornam esta ferramenta uma ferramenta de eleição para muitos profissionais das mais diversas áreas.

O Repast *Simphony* vem equipado com um Editor Visual de Agentes que possibilita a construção de um modelo recorrendo apenas a ferramentas visuais tal como na Figura 5. Para a criação de modelos mais complexos esta técnica é limitada havendo, para isso, necessidade de recorrer à escrita de código.

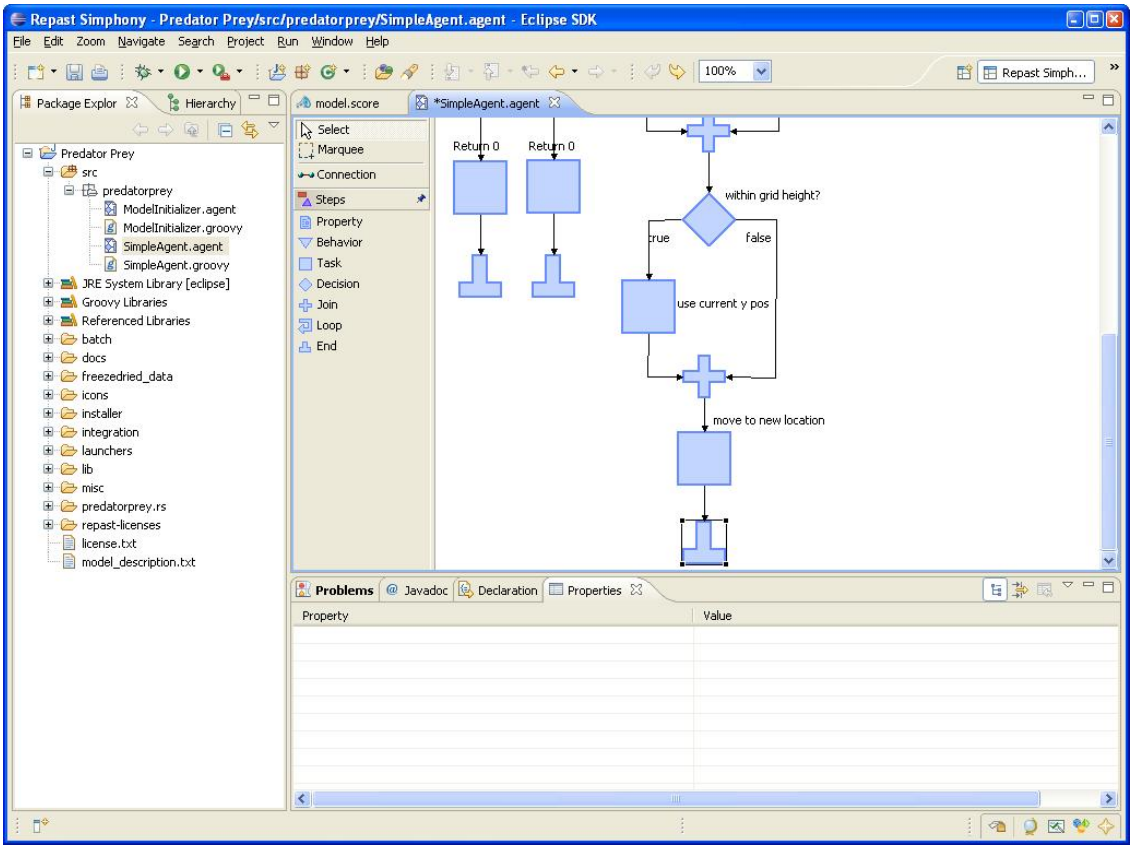


Figura 5 - Projeto recorrendo ao Editor Visual de Agentes [Repast Simphony, URL]

O *Repast Simphony* baseia-se numa tecnologia de *Contextos e Projeções.* Um contexto é basicamente um *recipiente* que contém agentes em que estes não têm qualquer relacionamento nem conceito de espaço. Os *Contextos* têm hierarquias e por isso podem conter *subcontextos*. Com as *Projeções* é possível atribuir aos agentes contidos num *Contexto,* um espaço e definir relacionamentos entre eles.

A versatilidade do *Repast Simphony* e a sua enorme, e sempre crescente comunidade de utilizadores, a possibilidade de poder ser executado em virtualmente todos os ambientes, a sua disponibilização em várias linguagens de programação, os seus algoritmos adaptativos, a escalabilidade, a facilidade de utilização, a sua interoperabilidade com algumas aplicações estatísticas e o facto de ser *open-source* fazem desta ferramenta uma opção muito interessante.

A criação de modelos é feita através da criação de objetos. O criador do modelo passa a informação dos objetos criados ao motor do *Repast Simphony* de forma declarativa e da mesma forma define a interação entre os vários componentes que constituem um modelo.

O *runtime system* do *Repast Simphony* define possibilidades como configuração de alguns parâmetros do modelo através de *point-and-click* (apontar e clicar), ligação a fontes de dados e ligação a ferramentas externas para análise estatística de informação e visualização de dados, assim como um sistema de informação geográfico integrado (GIS) em duas e três dimensões.

O *Repast Simphony* usa uma novidade do JAVA 5, as anotações. As anotações são usadas para declarativamente marcar código para utilização posterior e para notificar os agentes quando se verificarem alterações na simulação. Anotações são etiquetas de metadata que são compilados e afetam o código a que estão destinados.

Um exemplo de anotação mais usada é *ScheduledMethod*. Podemos ver no excerto de Código 1, um exemplo do uso dessa anotação.

Como se pode ver a anotação tem vários parâmetros:

- *start* – define quando queremos começar a fazer a monitorização deste método;
- *interval* – define com que intervalo essa monitorização deve ser feita;
- *priority* – define qual a prioridade que queremos atribuir a esta monitorização relativamente a outras monitorizações que estejamos a fazer sendo a que tem o maior valor a que também tem a maior prioridade.

Neste exemplo, a cada unidade de tempo da execução do modelo é lida a lista de consumidores.

```
@ScheduledMethod(start = 1, interval = 1, priority = 0)
 public void AgentesActivos() {
        final ArrayList<Consumidor> ListaConsumidores =
getConsumidorList();
```

Código 1 - Exemplo de uma anotação

O verdadeiro valor do *Repast Simphony* é notado quando precisamos de simulações que necessitem de muitos pormenores e quando há necessidade de controlar todos os aspetos da simulação em detalhe. A versatilidade do *Repast Simphony* permite-nos ter este controlo minucioso sobre qualquer simulação. Evidentemente que esta vantagem implica um maior esforço e mais tempo, assim como maiores e melhores conhecimentos de programação. Criar um modelo em *Repast Simphony* requer:

- mais tempo;
- maiores conhecimentos de programação;
- maior cuidado e atenção na escrita do código;
- depuração do código mais frequente e por vezes mais difícil.

Apesar deste maior esforço obtêm-se modelos controlados ao pormenor.

As bibliotecas do *Repast Simphony* permitem-nos criar todo o tipo de simulações para qualquer área científica seja ela social, económica ou em áreas mais experimentais como a física, a química a medicina, a informática, etc.

Estas bibliotecas estão definidas num conjunto de *packs* dos quais se podem destacar alguns dos mais importantes:

- ***Adaptation*** – Cria os *wrappers* para a utilização dos algoritmos genéticos, redes neurais e regressão.

- ***Agents*** – Usado para criação de agentes no *Editor Visual de Agentes.*

- ***Chart2*** – Classes para a criação de gráficos.

- ***Context*** – Classes para criação e manipulação de contextos de simulação. Um contexto contém agentes.

- ***DataLoader*** – Classe que define a estrutura de arranque da simulação, nomeadamente cria o contexto.

- ***Data2*** – Classe para manipular a informação gerada pelos agentes durante a simulação e que pode ser usada para geração de gráficos.

- ***GIS*** – Estas classes baseiam-se na projeção baseada em geografia. Este tipo de projeção é essencialmente um espaço em que os agentes são associados com uma geometria espacial (polígonos, pontos, etc).

- ***Parameter*** – Classes para criação de parâmetros usados na inicialização da simulação.

- ***Query*** – Extração de informação referente ao ambiente em que decorre a simulação (contexto, projeção, espaço, etc.).

- ***Relogo*** – Classes usadas nas simulações criadas com a linguagem *Groovy.*

- ***Space*** – Classes para criação do tipo de espaço em que os agentes se movimentam (grelha, geográfico, etc.).

- ***Visualization*** – Criação de um ambiente visual (*display*) para a simulação, ou seja, antes de podermos ver a simulação temos que dizer o que queremos ver.

3 Redes Colaborativas de Consumidores

Neste capítulo vão-se apresentar diversas topologias de redes colaborativas e alguns exemplos de uso. Também se vai falar sobre as razões que levam as pessoas a adotarem determinados produtos.

3.1 Caracterização

As redes colaborativas existem, praticamente, desde sempre e tendem a popularizar-se cada vez mais com a finalidade de criação de valor [Camarinha-Matos and Afsarmanesh, 2009].

Uma rede colaborativa é uma rede que consiste numa variedade de entidades (organizações ou pessoas) com uma grande autonomia, geograficamente distribuídas e heterogéneas em termos do seu ambiente, cultura, capital social e objetivos mas que colaboram para melhor atingir objetivos comuns ou compatíveis partilhando, também, os riscos [Camarinha-Matos and Afsarmanesh, 2009].

Segundo [Camarinha-Matos and Afsarmanesh, 2012] uma ligação em rede é um processo que envolve comunicação e troca de informação entre participantes para benefício mútuo.

Nestas redes colaborativas devemos notar a existência de dois tipos de colaboração, embora esta distinção não seja tratada ao longo deste trabalho:

- Cooperação, exige que haja troca de informação entre as partes de forma a atingirem os mesmos objetivos. Por exemplo quando várias partes contribuem para o fabrico de um artigo como acontece na indústria automóvel.
- Colaboração, quando as entidades intervenientes partilham informação, recursos e responsabilidade. Sendo mais exigente a nível de integração entre participantes acontece, por exemplo, quando técnicos de uma determinada área e pertencentes a entidades distintas, se juntam para criar uma solução conjunta.

A evolução e generalização da Internet deu lugar a um crescente número de negócios *online* e àquilo a que se chama *e-marketplaces.*

Provavelmente será nestes negócios *online* que melhor se poderá observar uma rede colaborativa.

Tomemos como exemplo alguns dos *sites* de comércio eletrónico (*e-commerce)* mais conhecidos, como a *Amazon* [Amazon, URL].

Esta, que começou por ser uma livraria *online* e vendeu o seu primeiro livro em 1995 a partir de uma garagem [Howstuffworks, URL] é agora um negócio que além de vender muitos outros produtos que não livros, conta com clientes em todo o mundo.

Qualquer pessoa pode vender os seus produtos na *Amazon*, aliás a *Amazon* tornou esta possibilidade tão fácil que a levou ao ponto de criar um API para que qualquer pessoa ou empresa possa criar uma loja *online* baseada na sua tecnologia [Amazon, URL]. As pessoas até lhes podem enviar os artigos que têm para venda que a *Amazon* faz todo o trabalho de distribuição e apoio ao cliente.

A *Rakuten* (também chamada de *Amazon* japonesa) é uma espécie de centro comercial *online* e reúne artigos de inúmeros retalhistas. Quando o cliente faz uma encomenda, essa encomenda é direcionada à empresa que possui o artigo de uma forma transparente ao utilizador. O ponto forte da *Rakuten* é precisamente o seu enorme número de retalhistas tornando assim possível fornecer produtos que normalmente são difíceis de encontrar [Pcworld, URL].

A maior arma publicitária das empresas que gerem estes negócios *online* são os próprios clientes que podem dar opiniões sobre os produtos incutindo, assim, maior confiança a quem os queira comprar.

Existem outros tipos de negócios *online,* que funcionam exclusivamente com produtos dos potenciais clientes, ou seja, onde cada pessoa pode ser cliente, vendedor ou simultaneamente cliente/vendedor [Ebay, URL]. Um dos maiores exemplos internacionais deste modelo de negócio é o *Ebay* [Ebay, URL]. Esta colaboração entre vendedores e compradores e a existência de um *stock* distribuído geograficamente ou descentralizado veio abrir a possibilidade de criação de negócios sem necessidade de grande investimento, uma vez que não há necessidade de haver um armazém nem *stock*, abrindo assim possibilidade para a prática de preços mais competitivos e a oferta de um leque muito mais alargado e variado de produtos. O *Ebay* desenvolveu um API para que qualquer pessoa ou empresa possa comunicar com as suas bases de dados através da linguagem XML.

3.2 Adesão a novos produtos

A publicidade, ou promoção de produtos, através da comunicação direta entre pessoas é a mais barata e das mais eficazes que se pode ter [Entrepreneur, URL], além de ser um fenómeno interessante, por isso, é importante compreender como funciona.

As interações deste tipo podem ser modeladas através de agentes, de forma a serem mais facilmente compreendidas.

A ideia principal é que admitimos que as pessoas, aqui representadas por agentes, se deixam influenciar por conhecidos, mais do que por qualquer outra forma de publicidade, mesmo na TV. A força dessa influência, a que todos estão sujeitos, varia entre outras coisas com a ligação que temos com as pessoas que pertencem à nossa rede de conhecimentos. Podemos ter uma ligação forte quando a comunicação é feita entre os elementos do nosso grupo ou fraca quando a comunicação é feita com os elementos com quem interagimos mas não pertencem ao nosso grupo [Goldenberg, et al., 2001].

Algumas condições são importantes no processo de decisão para a adesão a um produto [Kotler and Armstrong, 2012].

Este processo começa pelo reconhecimento de uma necessidade, que pode ser despoletado por um anúncio ou por uma conversa com um amigo que o pode levar a interessar-se por um produto.

Depois da necessidade identificada pode dar-se início ao processo de procura de informação. A procura de informação pode ocorrer conforme o grau de necessidade que o consumidor tenha pelo produto. Se a necessidade for grande e ele estiver satisfeito com a informação que já possui e que lhe foi passada por um amigo ou anúncio e se o produto tiver um preço razoável para as suas possibilidades financeiras, é natural que ele não procure muito mais informação.

Se, por outro lado, o produto for caro, como por exemplo um novo automóvel ou até um computador ou um sistema de som de alta-fidelidade, é muito provável que o consumidor vá procurar mais informação através de pessoas conhecidas, procura na internet ou conversas com vendedores ou técnicos especializados.

A avaliação de alternativas será o próximo passo. Esta, varia de pessoa para pessoa. Algumas poderão fazê-lo ponderando com cuidado, outras poderão comprar baseadas num impulso ou intuição, outros poderão olhar à sua volta e ver que mais pessoas estão a usar, se for caso disso, outros poderão pedir conselhos a amigos e vendedores ou técnicos especializados.

O próximo passo é a decisão de aquisição. Esta decisão pode ser influenciada por dois fatores [Kotler and Armstrong, 2012]:

- As atitudes dos outros – O consumidor pode decidir baseado em escolhas ou opiniões de pessoas importantes para ele;
- Fatores situacionais inesperados – o panorama económico piorar, a concorrência baixar o preço ou um amigo dizer que afinal está insatisfeito com o produto.

Todos estes passos, em geral, são tomados pelos consumidores desde a ideia inicial de comprar um produto até à concretização da compra.

Quando o consumidor adota um novo produto existem várias etapas nesse processo de adoção [Kotler and Armstrong, 2012]:

- O conhecimento – O consumidor toma conhecimento de um novo produto mas não tem informação suficiente sobre ele e poderá ser aqui, nesta situação, que nasce um inovador, um consumidor que está na disposição de adotar um produto quando ainda há pouca informação sobre ele;
- O interesse – reconhecendo potenciais ao novo produto, o consumidor procura mais informação sobre ele;
- Avaliação – O consumidor considera se deve ou não experimentar o produto;

- Teste – O consumidor experimenta o produto para tentar descobrir o seu valor e até que ponto pode ser satisfazer as suas necessidades;
- Adoção – O consumidor decide tornar-se um utilizador regular do produto.

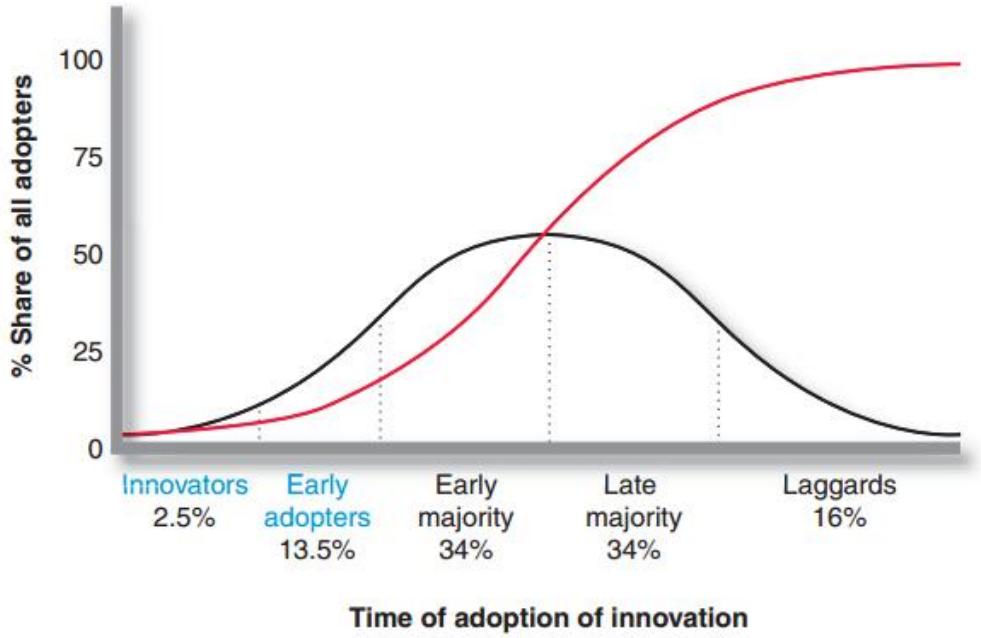


Figura 6 - Tempo de adoção de uma inovação [Kotler and Armstrong, 2012]

Conforme se vê pela curva a preto na Figura 6, depois de um início lento, um crescente número de pessoas adota o produto. O número de pessoas que adota atinge o pico e depois desce numa altura em que relativamente poucas pessoas ainda não adotaram. Conforme sucessivos grupos de consumidores adotam o novo produto (a curva a vermelho) o produto acaba por atingir o nível de saturação. Os inovadores estão definidos como os primeiros 2,5% de compradores a adotar uma nova ideia [Kotler and Armstrong, 2012].

A difusão de um novo produto ou inovação numa rede social tem uma evolução denominada Cascata [Alkemade and Castaldi, 2005], isto é, existindo um pequeno grupo (normalmente denominados inovadores) que adota o produto pela primeira vez, os consumidores em contacto com eles adotam, depois, o produto, os consumidores em contacto com esses consumidores adotam também e por aí fora. A quantidade de consumidores que adotam é crucial para considerar o novo produto um sucesso ou fracasso.
Em todo este processo de divulgação tem que se considerar a vontade que o agente tem de divulgar o produto. Nem toda a gente gosta de divulgar as suas preferências porque gosta de se sentir especial e não gosta de ser imitada. Por outro lado, algumas pessoas adotam determinado produto por sem ter uma razão especial (pois se não for esse, podem escolher outro produto) logo, normalmente, não têm grande empenho em o divulgar. É evidente que

depois também temos os entusiastas que não podem deixar de mostrar a grande novidade a quem os rodeia.

Nesta forma de divulgação de novos produtos existem dois tipos essenciais de consumidores: os inovadores, que gostam de estar um passo à frente, adoram novidades e de serem os primeiros a usá-las, e os seguidores, que só adotam um novo produto depois de verem um determinado número de pessoas que já o tenham adotado.

Segundo [Banerjee, 1992] este movimento de informação pode levar ao "comportamento em manada" em que os agentes decidem única e exclusivamente baseados nas decisões das pessoas que os rodeiam.

É de vital importância que seja compreendida a forma como essa informação é passada de uns para os outros, o que motiva a passagem dessa informação e as razões que podem levar outros consumidores a aderirem a um produto baseados nessa informação. Finalmente é importante saber quanto tempo irá levar até haver consumidores suficientes para que o novo produto seja sustentável para a empresa que o lança no mercado.

Para criar campanhas de marketing eficazes que atuem nas redes sociais é necessário compreender como funcionam essas redes, como os seus consumidores comunicam e as suas características.

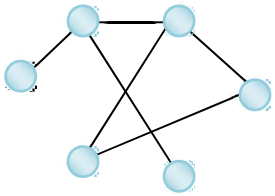


Figura 7 - Rede em malha

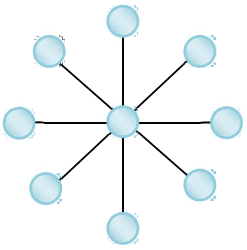


Figura 8 - Rede em estrela

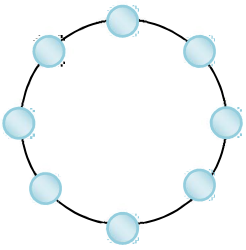


Figura 9 - Rede regular

A forma como os consumidores se ligam pode ter várias topologias, especialmente se o produto ou serviço a ser divulgado for destinado a um grupo específico de consumidores.

A mais vulgar é a topologia em malha (Figura 7) em que os consumidores comunicam entre si, mesmo não se conhecendo todos uns aos outros. Estas ligações existem de uma forma aleatória e facilmente um produto ou serviço é divulgado neste tipo de rede, porque normalmente há alguém que conhece alguém e muitas vezes o que acontece é cada consumidor, ou potencial consumidor, receber a mesma informação de diversas fontes diferentes. Este tipo de rede será provavelmente o melhor e mais eficaz para a disseminação de informação porque, dadas as suas características, essa informação é divulgada muito mais rapidamente e chegará a uma maior número de pessoas. Em contrapartida é necessário despender de maiores esforços para chegar ao maior número de pessoas possível numa fase inicial.

A empresa interessada em introduzir um novo produto ou serviço neste tipo de rede necessitará de definir uma estratégia de marketing que abranja um maior número de pessoas. Quanto mais pessoas forem influenciadas maior poderá ser a abrangência da rede. Para produtos com maior concorrência poderá haver necessidade de chegar a mais pessoas, havendo, assim, necessidade de um maior prolongamento no tempo, logo, um maior investimento.

Uma topologia em que será mais fácil para uma empresa penetrar é a rede em estrela (Figura 8). Neste tipo de rede, tipicamente, só o consumidor que está no centro da estrela ou membro dominante [Camarinha-Matos and Afsarmanesh 2012], é que divulga, logo a empresa apenas tem que afetar esse consumidor, precisando para isso de uma técnica de marketing mais localizada e menos extensa no tempo. Este tipo de rede existe em casos como um professor numa sala de aula em que poderá sugerir aos seus alunos um determinado produto (ex.: livros ou algum tipo de material escolar) ou então um médico que pode sugerir aos seus pacientes um determinado tipo de serviço ou produto.

Existe ainda outro tipo de rede chamada regular (Figura 9). Neste tipo de rede cada consumidor está a ligado a um número determinado de outros consumidores, normalmente os seus vizinhos [Alkemade and Castaldi, 2005]. Cada nó, nesta rede, representa um número exato de vizinhos. Esta rede define uma ligação em que o comportamento de cada consumidor depende do comportamento dos seus vizinhos mais próximos. Esta rede, normalmente serve para caracterizar ambientes como escolas ou pequenas comunidades.

3.3 Exemplos de uso

Hoje em dia para as empresas sobreviverem no mercado têm que criar parcerias com outras empresas que, assim, acrescentam valor às suas atividades. Na era da informação, os consumidores, já não compram produtos (com a exceção de casos pontuais) apenas pela embalagem mas sim pelo valor que esses produtos podem acrescentar às suas vidas. Toda esta colaboração faz com que não seja fácil saber quem nos poderá trazer valor. É muito fácil uma empresa criar protocolos de colaboração com outra para mais tarde descobrir que essa colaboração, afinal, não lhe traz qualquer benefício porque não é completamente compatível com as suas necessidades. Também, pode acontecer a um profissional liberal cair no mesmo erro porque não percebeu bem quais as necessidades do mercado em que atua. Em último caso é sempre possível tomar uma decisão por se basear numa opinião da fonte errada.

Muitos exemplos, modelos e estudos existem respeitantes a redes colaborativas. Um bom exemplo é o estudo de [Lee, et al., 2013] para maximizar a possibilidade de sucesso de um novo produto e minimizar o risco e custo de oportunidade do falhanço de um produto.

Os autores observaram que, os consumidores de *notebooks* Coreanos, tendem a comprar um produto sem hesitação e que a sua rede social é bastante regular mas pouco densa, e

conduziram análises de sensibilidade no que respeita ao relacionamento entre a rede social e a distribuição do tempo de aquisição.

Outro exemplo é o de [Zhang, et al., 2009] em que os autores recorrem a sistemas multi-agente colaborativos para monitorização e planeamento em tempo real de locais de construção. Um sistema multi-agente é discutido para suportar operadores de equipamento de construção através do uso de agentes, comunicações sem fios e tecnologia para captura de informação do local de construção. A informação recolhida, através de sensores ligados ao equipamento em conjunto com um modelo 3D actualizado do local de construção, é processada, pelo sistema multi-agente, para detectar qualquer colisão ou outros tipos de conflitos referentes à operação de equipamentos e para gerar um novo plano em tempo real.

A gestão de uma obra, mesmo pequena, apresenta um grande número de desafios, daí uma gestão de recursos assim como uma sintonização do *timing* entre as várias equipas que depende do tempo que cada uma leva a executar as suas tarefas, conduz a uma necessidade crescente de uma forma de controlo do projeto que seja eficiente, especialmente se tomarmos em conta que os tempos de duração das tarefas raramente são cumpridos.

Um dos objetivos da utilização da simulação baseada em agentes em Economia é a previsão. Na Economia, é frequente o desenvolvimento de modelos que reproduzem a dinâmica de alguns comportamentos dos agentes económicos, tais como bancos ou consumidores. Através da simulação podemos estudar a passagem do tempo e, assim, usar o modelo para prever o futuro. Um dos projetos que utiliza simulação de forma pedagógica e científica como técnica de previsão, entre outras, é o Eurace [Eurace, URL]. Este projeto, apoiado pela União Europeia, aborda o complexo problema de simular fenómenos macroeconómicos, propondo uma abordagem inovadora para a modelação macroeconómica e design de política económica no âmbito da economia computacional baseada em agentes. Os objetivos do projeto são caracterizados por âmbitos científico, tecnológico e social.

[Carley and Hill, 2001] criaram o *Construct-O*, um modelo multi-agente de mudança social e individual resultante da difusão de informações entre os indivíduos adaptáveis e comunicativos. Neste modelo, cada agente tem uma certa posição na rede social, em conjunto com um modelo de conhecimento. Quando os indivíduos interagem, eles comunicam uma parte do seu conhecimento para que a informação se difunda.

Um outro exemplo do uso deste tipo de técnicas é o projeto AMES - *Wholesale Power Market Test Bed*. AMES, é um laboratório computacional baseada em agentes de código aberto projetado para o estudo sistemático dos mercados de energia restruturados [Hongyan and Tesfatsion, 2009].

A simulação baseada em agentes é também uma forma interessante para explicar conceitos em sala de aula nas disciplinas ligadas a vários campos do saber. Nas ciências exatas e mesmo nas ciências sociais, é frequente utilizarem-se ferramentas de programação matemática (Mathlab, ou R) para construir simulações. Estas ferramentas são limitadas em termos visuais

e de comunicação interna dos agentes. Por isso, é importante que os alunos tenham acesso a simulações já elaboradas, onde apenas é necessário configurar alguns parâmetros.

4 Plataforma para simulação de redes colaborativas

Neste capítulo vai-se descrever a aplicação criada no âmbito desta Tese, apresentando algum do código mais relevante e algumas configuração necessárias, no *Repast Simphony,* para que a aplicação seja executada com os resultados esperados.

4.1 Introdução

No âmbito desta tese foi desenvolvido um modelo que pode facilmente ser expandido e pretende simular uma situação económica real em que uma empresa lança um novo produto no mercado. Este modelo simula a forma como as pessoas aderem a um novo produto como consequência da divulgação feita pelos seus conhecidos. O modelo pretende ser uma ferramenta para ajudar a compreender a forma como a informação passa de pessoa para pessoa, bem como o tempo necessário para garantir um determinado número de adesões ao produto.

O modelo foi construído tirando proveito das potencialidades e características do *Repast Simphony* e de forma a que fosse possível expandi-lo acrescentando novas características e/ou restrições podendo-se, assim, criar simulações mais variadas, até porque os produtos não são todos iguais tal como os mercados. Desta forma a simulação foi preparada para facilmente integrar as variadas características que são comuns a este tipo de situações.

O modelo permite-nos recolher dados relativamente ao comportamento dos agentes.

Os agentes estão inseridos num ambiente artificial onde interagem uns com os outros, tentando, os consumidores do produto, “convencer” os agentes não consumidores a aderirem ao seu uso.

Pretende-se, com esta simulação, conseguir compreender de que forma um produto, quando entra no mercado, é aceite, a forma como a informação sobre esse produto é propagada através das redes sociais e o tempo que demora até que um número significativo de pessoas adira a esse produto.

4.2 Características do modelo

4.2.1 O modelo económico

O modelo económico adotado é baseado no trabalho *Strategies for the diffusion of Innovations on Social Networks* [Alkemade and Castaldi, 2005].

Quando uma empresa lança um novo produto no mercado nunca sabe como vai ser a reação do público a esse produto. O sucesso ou insucesso do produto está dependente de alguns fatores que afetam o comportamento do consumidor [Kotler and Armstrong, 2012].

- Fatores culturais – a cultura em que o consumidor está inserido como o comportamento das pessoas com que lida, dos amigos e da família, que têm grande influência na construção de caracter, assim como a classe social têm um grande peso na definição do tipo de consumidor.
- Fatores sociais – O seu pequeno grupo como os amigos e as pessoas com quem lida diariamente, assim como o seu *status* social também desempenham um papel importante na definição do consumidor. Outra fonte de influência podem ser grupos aos quais o consumidor não pertence mas admira como a equipa preferida de futebol ou a sua ou suas bandas musicais preferidas. A abertura a essa influência é tanto maior quanta a vontade que o consumidor tem de impressionar as pessoas que fazem parte do seu meio.
- Fatores pessoais – Nas suas opções, o consumidor, também é afetado por outras características como a idade, profissão, situação económica, personalidade e estilo de vida.
- Fatores psicológicos – Motivação, perceção, a aprendizagem, crenças e atitudes. A motivação ou razão que move a pessoa – uma pessoa motivada é levada a agir, a perceção influencia a forma como a pessoa age. A aprendizagem é a informação que as pessoas obtém da experiencia e que é capaz de influenciar e mudar o seu comportamento. A crença é a ideia que se tem sobre algumas coisas e que pode vir da aprendizagem. As crenças podem ser afetadas por conhecimento, opinião ou apenas vontade de acreditar que pode vir da fé, assim como ter ou não alguma carga emocional. A atitude descreve a relativa consistência das suas avaliações, sentimentos e tendências relativamente a um objeto ou ideia.

4.2.2 Principais características do modelo multi-agente

Serão analisadas as seguintes características do modelo:

- Os agentes
- Heterogeneidade

- Objetivo dos agentes
- Representação da informação

Nesta secção apresentam-se algumas das características próprias deste modelo comparando-os com as características gerais comuns a todos os agentes nos vários módulos.

4.2.2.1 Os agentes

Este modelo considera dois tipos distintos de agentes consumidores: o inovador e o seguidor.

O inovador representa o consumidor que é sempre o primeiro a aderir a um novo produto ou moda. Este tipo de consumidor tem como características principais o facto de gostar de novidades e de se distinguir, além de ter algum poder financeiro. Normalmente este tipo de consumidor, quando já há muita gente a usar os produtos que ele usa, tende a procurar novos produtos para substituir os que está a usar [Alkemade and Castaldi, 2005]. Este comportamento é simulado através da comunicação entre agentes. Esta comunicação é uma característica comum a todos os agentes que fazem parte de um sistema multi-agente. Esta comunicação vai proporcionar ao agente poder influenciar outros agentes, neste caso os seguidores.

A principal característica deste agente inovador é a capacidade de influenciar os outros agentes. Esta influência depende de diversas condições tais como a sua vontade em divulgar o novo produto. Como se sabe nem todos os consumidores gostam de divulgar as novidades, alguns por desinteresse outros para evitar serem imitados podendo, assim, manter uma relativa exclusividade.

Outra condição para o sucesso na passagem de informação é o "peso" da sua opinião perante a pessoa que tenta influenciar. Por exemplo, para a compra de um leitor MP3 o melhor amigo pode, com certeza, ter muito mais influência na decisão do que outro qualquer conhecido, às vezes até mais do que um especialista.

Os seguidores, pelo seu lado, têm algumas características interessantes. Podem estar ou não abertos a sugestões e essa abertura poderá ser grande ou pequena, deixando-se influenciar com facilidade, ou não, podendo demorar muito mais tempo a aderir ao novo produto.

Além destas características a sua aderência ao novo produto irá ser mais ou menos rápida de acordo, principalmente, com dois fatores:

- O número de pessoas que o tentam converter;
- O peso ou importância que a opinião dessas pessoas tem para si.

Normalmente os seguidores, ao contrário dos inovadores, só aderem a um novo produto se já houver uma grande percentagem de pessoas pertencentes ao seu meio que já o tenham adotado [Alkemade and Castaldi, 2005].

4.2.2.2 Heterogeneidade

Para manter alguma diversidade nos resultados das simulações é muito importante que os agentes não tenham todos as mesmas características ou pelo menos que não as tenham em quantidades iguais. Os consumidores não têm todos as mesmas características e sem a heterogeneidade dos agentes a simulação não faria sentido porque o resultado seria previsível e repetitivo e não imitaria as diversas situações de mercado, possíveis.

No caso dos agentes iniciadores importa caracterizar a sua vontade em passar a informação, o seu empenho em o fazer, e o peso da sua opinião perante os potenciais seguidores.

No caso dos seguidores, pelo seu lado, não terão todos a mesma predisposição para aderir a novos produtos.

Além destes aspetos, tem ainda que ser equacionados fatores condicionantes, tais como poder económico e imagem social que poderá ser afetada pela adoção de um novo produto.

4.2.2.3 Objetivos dos agentes

Os agentes tentam alcançar apenas o objetivo que é converter todos os agentes inicializados como seguidores, em consumidores efetivos.

O tempo demorado a “convencer” os seguidores a aderir ao produto será importante para compreender o tipo de público com que se está a lidar.

4.2.2.4 Representação de informação

A informação produzida pela simulação multi-agente é de extrema importância para a compreensão da simulação e para extração de diversas conclusões. Neste caso, são gerados dois ficheiros de informação. Um deles, mais pormenorizado, contém a informação relativa a todos os instantes no tempo em que um agente é influenciado. Esta informação dá-nos a perceção de como, e quando, os agentes são afetados.

O outro ficheiro contém o registo da altura no tempo em que os agentes são convertidos em consumidores podendo, esta informação, dizer-nos, por exemplo, o espaço de tempo médio entre a conversão dos diferentes agentes.

Também é fornecido ao modelo um ficheiro com alguns dados variáveis para maior abrangência da simulação, especialmente para definição daquilo a que se chama limiar (*threshold)* [Alkemade and Castaldi, 2005].

O *limiar* é um valor que define, por exemplo, qual a resistência de um agente à inovação e qual o poder de persuasão de um agente.

Entre a entrada e a saída de informação temos uma grande margem de manobra em termos de variedade de condições de simulação podendo, assim, criar condições para simulações diferentes.

4.3 Descrição funcional do modelo

4.3.1 Análise geral

O conceito básico desta simulação é a existência de dois tipos de agentes, os inovadores e os seguidores.

X						X	
				X			
		X			X		
X							
			X				X
	X	X					
			X		X		
						X	

Figura 10 - Exemplo de cenário

- A simulação inicia com N agentes distribuídos aleatoriamente numa grelha;
- Só é permitido um agente por cada posição (célula) do cenário. Quando dois agentes estão em células contiguas é lida a informação de cada um deles. Essa informação é depois usada para influenciar ou ser influenciado;
- A cada agente é-lhe atribuído aleatoriamente características como: inovador, seguidor, poder de persuasão, limiar de mudança (facilidade com que se deixa influenciar) e nível de interesse em divulgar o produto;
- Os agentes são criados, quase na totalidade, na ignorância do produto, com exceção dos inovadores (*starters*) que vão iniciar a simulação;
- Cada agente move-se para qualquer célula da grelha atribuída aleatoriamente e contígua àquela onde se encontra;
- Se a célula estiver ocupada perde a vez de se mover;
- Se a célula não estiver ocupada move-se para lá e todos os agentes que se encontrem nessas células vizinhas podem ou não ser influenciados (baseado na vontade de divulgação do agente);
- O objetivo dos agentes inovadores é influenciarem os outros agentes (não inovadores ou seguidores) na adoção (consumo) de um produto;
- A partir dos dados do influenciador são criados os termos em que é exercida essa influência;
- A partir da informação extraída do agente a ser influenciado são definidos os termos em que essa influência vai ser exercida;
- Em resumo, é determinado o peso de influência disponibilizado pelo agente influenciado e a quantidade desse peso de influência, que o agente a ser influenciado, está disposto a aceitar;
- Qualquer agente, depois de adotar o produto pode influenciar outros agentes;

- Os agentes influenciados podem aderir, ou não, conforme a sua resistência à inovação e o peso de persuasão do agente influenciador;
- Alguns agentes inovadores podem desistir do produto quando uma determinada percentagem da população já o está a usar;
- Por fim, todos os agentes desistem do produto por ele já ter algum tempo e/ou começar a ser substituído.

4.3.2 Análise descritiva

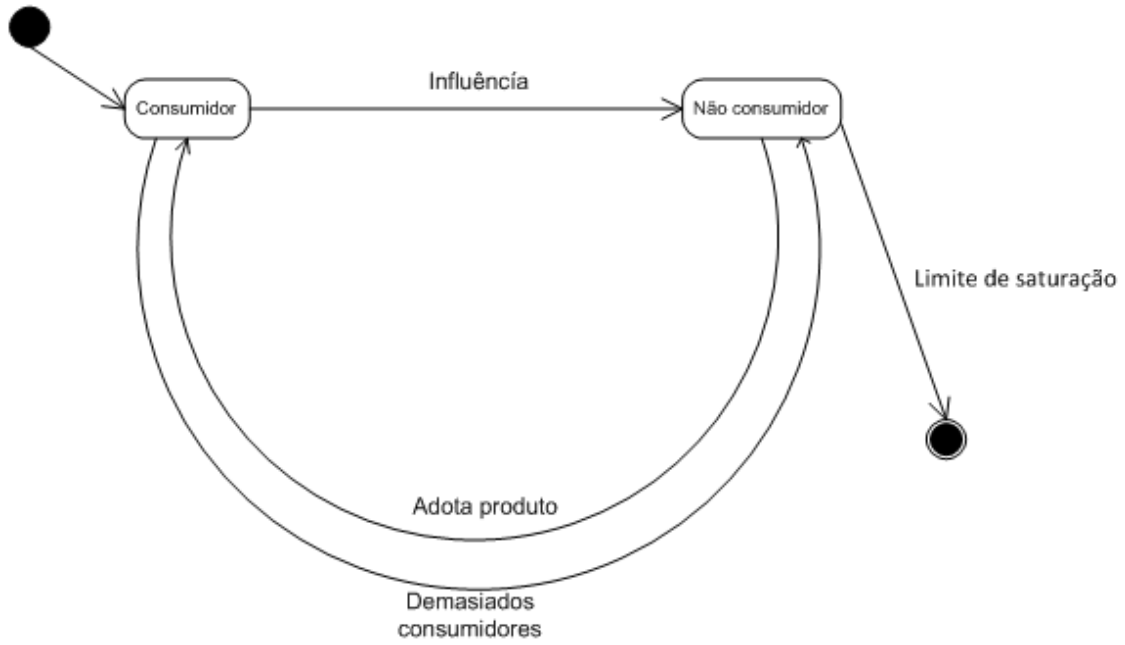


Figura 11 – Diagrama de actividade

- Não havendo informação sobre o novo produto, o consumidor olha à sua volta para extrair informação.
- O produto pode ter pouco interesse para o consumidor mas ele adotá-lo por ser moda, baseando-se no número de pessoas que já o adquiriram.
- Alguns consumidores gostam de se distinguir dos restantes por serem inovadores. Os inovadores são poucos.
- A maior parte das pessoas são "seguidoras" e são efetivamente responsáveis pela divulgação da inovação.
- O *feedback* dos consumidores pode ser positivo ou negativo e afetar, dessa forma, a decisão dos potenciais novos consumidores.
- O consumidor que divulga o novo produto a outro consumidor pode ter um determinado grau de credibilidade que influencia a decisão deste.
- O consumidor pode não se interessar por divulgar o produto.
- A inovação pode deixar de ser atrativa uma vez adotada por um grande número de pessoas e os consumidores podem deixar de usar os seus produtos quando os seus vizinhos, que os usam, ultrapassam em número o seu *over-exposure threshold* (altura

em que o produto já está a ser usado por mais pessoas do que aquelas com que o consumidor está disposto a partilhá-lo).

- O lançamento de produtos concorrentes ou o desgaste pode levar os consumidores a perderem o interesse pelo produto.

Na Figura 11 podemos ver o diagrama de atividade da simulação, em que o produto é adotado, inicialmente, por um número restrito de consumidores, os chamados iniciadores. Estes iniciadores influenciam outros agentes até que eles se tornem consumidores. Esses agentes influenciados irão, por sua vez, influenciar outros. Quando já existe uma determinada quantidade de utilizadores do produto, o chamado limite de saturação e que varia de consumidor para consumidor, estes começam a deixar de usar o produto [Alkemade and Castaldi, 2005].

4.4 Descrição técnica do modelo

O modelo foi desenvolvido recorrendo à linguagem de programação Java, ao IDE (*Integrated Development Environment)* Eclipse e à biblioteca *Repast Simphony*.

4.4.1 Diagrama de classes

O modelo é composto por diversas classes com as seguintes funções:

- NetWorkSpreadContextBuilder (classe central da simulação);
- Consumidor (principal classe que define os agentes);
- Config (configuração do modelo);
- FileManager (lê as configurações externas para a simulação) *;*
- ConfigData (*wrapper* para manipulação de ficheiros);
- Agentestyle2D (classe usada para alteração das cores dos agentes);
- Utils (alguns métodos de âmbito geral e úteis à simulação).

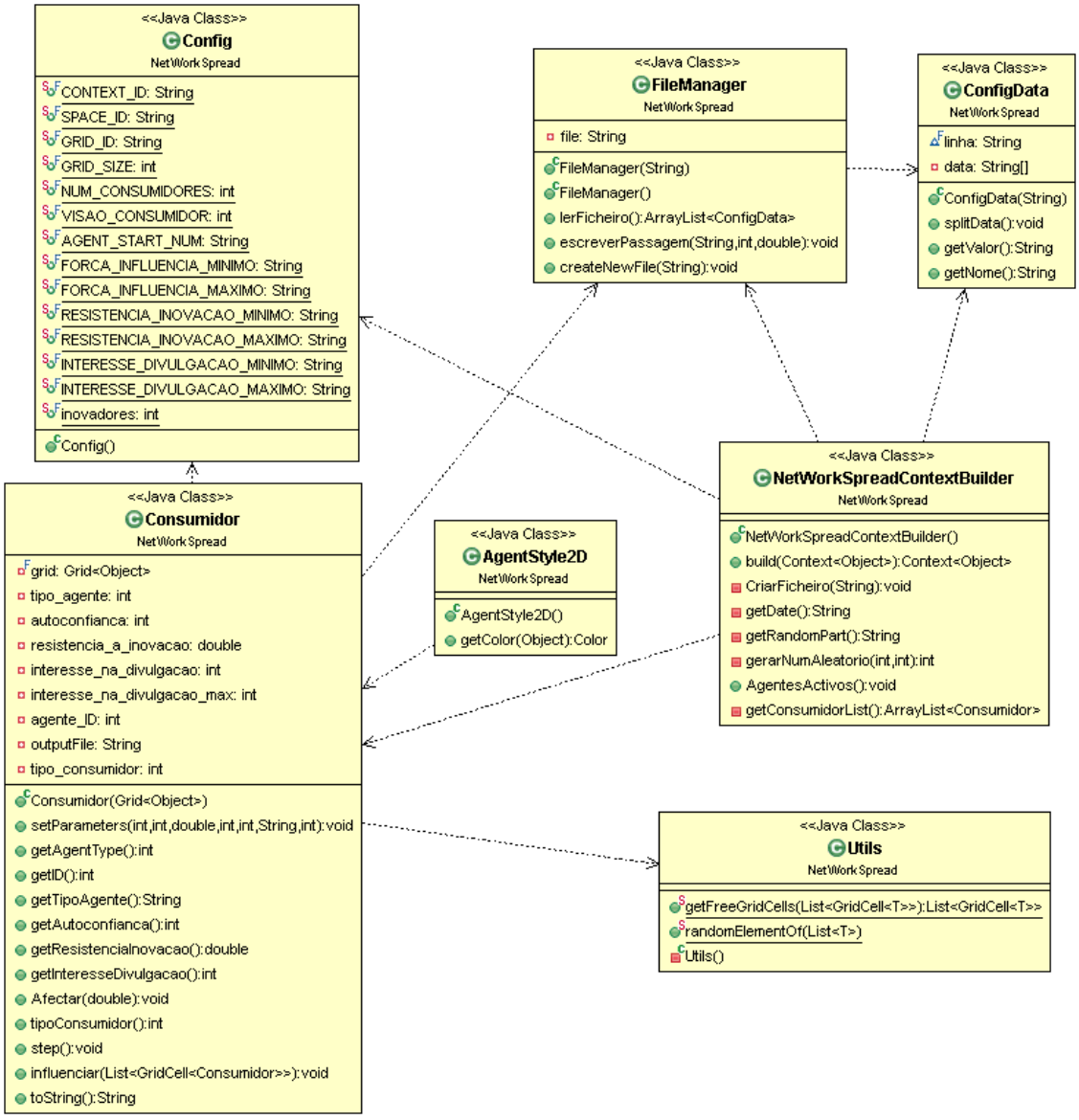


Figura 12 - Diagrama de classes da simulação

Tabela 2 - A classe *NetWorkSpreadContextBuilder*

A classe *NetWorkSpreadContextBuilder*
O arranque da simulação é feito através da classe *NetWorkSpreadContextBuilder*.
Esta classe é essencialmente a criadora de toda a simulação.
Cria o espaço contínuo e a grelha para a simulação.
Cria as instâncias da classe *Consumidor* criando, assim, os agentes necessários à simulação e definidos no ficheiro externo 'datafile.csv' que também é lido por esta classe.
Associa os agentes com o contexto e configura os seus atributos.
Lê os valores definidos nas opções do modelo.

Tabela 3 - A classe *Consumidor*

A classe *Consumidor*
Esta classe é a essência da simulação e é a classe que define os agentes.
Várias funções são definidas, umas para afetar os atributos e outras para ler os seus valores.
Dois tipos de consumidores são definidos: os iniciadores ou *starters* e os seguidores ou *followers*.
Algumas dos principais atributos definidos na classe são:
autoconfianca - esta variável define, em termos quantitativos, o poder de persuasão do agente em relação aos outros agentes consumidores
resistencia_a_inovacao - a resistência de um agente consumidor antes de aderir ao novo produto
interesse_na_divulgacao - a vontade do agente consumidor em divulgar o produto
agente_ID - identificação do agente
outputFile - ficheiro de saída dos resultados da simulação

Tabela 4 - A classe *Config*

A classe *Config*
Esta classe contém os parâmetros necessários à configuração da simulação
CONTEXT_ID - identificação do contexto onde decorre a simulação
SPACE_ID - identificação do espaço onde decorre a simulação
GRID_ID - identificação única da grelha onde decorre a simulação
GRID_SIZE - tamanho máximo da grelha
NUM_CONSUMIDORES - número inicial de agentes
VISAO_CONSUMIDOR - o número de células a que o agente tem acesso
FORCA_INFLUENCIA_MINIMO - valor mínimo relativo à influência do agente
FORCA_INFLUENCIA_MAXIMO - valor máximo relativo à influência do agente
RESISTENCIA_INOVACAO_MINIMO - valor mínimo da resistência do agente em aderir ao produto
RESISTENCIA_INOVACAO_MAXIMO - valor máximo da resistência do agente em aderir ao produto
INTERESSE_DIVULGACAO_MINIMO - valor mínimo para o interesse na divulgação
INTERESSE_DIVULGACAO_MAXIMO - valor máximo para o interesse na divulgação

Tabela 5 - A classe *FileManager*

A classe *FileManager*
Esta classe faz a gestão de ficheiros quer relativos às configurações, quer relativos à agregação e disponibilização de resultados.
O ficheiro tem o formato csv (*comma separated value – valores separados por vírgulas*).
Os campos são:
Nº de agentes – definição do número de agentes que a simulação deve usar
Nº de iniciadores – Número de iniciadores que queremos ao começar a simulação
Local para os ficheiros – caminho para armazenamento dos ficheiros com os resultados da simulação.
O ficheiro '*ModelOutput'* tem quatro campos, sendo eles:
ID – um número único que identifica o agente
TipoAgente – o nome do tipo de agente (Inovador ou seguidor)
ResistenciaInovacao – a resistência que o agente ainda mantém antes de ser convertido como um novo aderente ao produto
Tick – o lugar no tempo da simulação em que a alteração se verificou.

Tabela 6 - A classe *ConfigData*

A classe *ConfigData*
Estra classe é usada para validação de dados do ficheiro '*datafile.csv*'.
O método '*splitData'* separa os diversos valores, lidos linha a linha, do ficheiro e separa-os pelo caracter de separação definido, neste caso o ponto e vírgula.

Tabela 7 - A classe *Agentestyle2*

A classe *Agentestyle2D*
Esta classe estende a classe '*DefaultStyleOGL2D*'pertencente à biblioteca do Repast Simphony e tem como finalidade definir algumas configurações visuais.
Por exemplo para mudarmos a cor aos agentes.

Tabela 8 - A classe *Utils*

A classe *Utils*
Esta classe cria alguns métodos de utilidade geral tais como o método *'getFreeGridCells'* que devolve as células livres à volta de uma determinada célula e o método '*randomElement*' que, com base nos valores devolvidos pelo método anterior, devolve uma célula aleatória que será para onde o agente se vai mover.

Na Figura 12 pode-se ver o diagrama das classes usadas para criar esta simulação.

A classe *NetWorkSpreadContextBuilder* é a classe que inicia a simulação criando instâncias das classes *Config, FileManager, ConfigData* e *Consumidor*.

A classe *FileManager*, que lê as configurações do ficheiro externo, cria uma instância da classe *ConfigData*.

A classe *AgentStyle2D* utiliza uma instância da classe *Consumidor*, que lhe é passada como argumento, para afetar a sua cor mediante o seu nível de resistência ao novo produto.

A classe *Consumidor,* que cria os agentes na simulação, cria uma instância da classe *Config*, onde vai ler algumas variáveis de configuração da simulação e cria uma instância da classe *Utils* que contém alguns métodos generalistas.

4.4.2 O Consumidor

A classe *Consumidor* é a que define os agentes. Há dois tipos de consumidores, os *iniciadores* e os seguidores. Na criação dos agentes alguns parâmetros devem ser definidos. Para este fim foi criado o método *setParameters()*.

Tabela 9 - Variáveis usadas na definição de cada agente

Nome da variável	Função da variável
persuasão	Poder de persuasão que o agente tem sobre um futuro utilizador do produto.
resistência_a_inovação	Define a resistência que o agente tem à influência, ou seja, um valor baixo e o agente rapidamente se deixa convencer a utilizar o novo produto, um valor alto e o agente apenas aderirá ao novo produto com mais insistência.
interesse_na_divulgação	Define o interesse que o agente tem em divulgar o produto.
Id	Identificação do agente.

4.4.2.1 Comportamento

Os agentes movem-se aleatoriamente no cenário ou seja na grelha. Quando estão posicionados numa célula contígua a outra ocupada por outro agente, podem influenciá-lo ou não, isso depende da sua vontade de divulgar o novo produto definido na variável *"interesse na divulgação".* Esta variável, quanto mais alta for maior será a vontade do agente divulgar o novo produto.

A função *"Influenciar"* recebe um valor do tipo *double*: o *índice*. O índice será o valor a reduzir à resistência do agente a ser influenciado e é baseado na seguinte fórmula:

$$\frac{interesse\_na\_divulgacao}{interesse\_na\_divulgação\_MAX} * persuasao$$

O valor do interesse na divulgação atribuído ao agente a dividir pelo valor máximo possível dá um rácio que a multiplicar pelo poder de persuasão do agente vai resultar no valor a reduzir à resistência à inovação.
Por exemplo:

Assumindo que o agente tem um *interesse_na_divulgação* de 5 e o máximo que poderia ter é 20 e que o seu poder de persuasão é 7. Para estes valores teríamos:

Proporção = interesse_na_divulgação/interesse_na_divulgação_MAX = 5/20 = 0,25

Com base nesta percentagem a ser usada com a *persuasão,* irá calcular-se o valor de influência a ser subtraído à resistência do novo agente.

0,25 * persuasão = 0,25 * 7 = 1,75

Quando a resistência do agente atinge o valor 0, o agente passa a ser um utilizador do produto, ou seja, passa a seguidor.

Quando a resistência à inovação atinge o valor 0 é gravado, num ficheiro em formato de texto, o *ID* do agente que acabou de ser convertido e o momento da simulação em que isso aconteceu. O instante da simulação é obtido com base no Código 2.

```
double tick =
RunEnvironment.getInstance().getCurrentSchedule().getTickCount();
```

Código 2 - Leitura do tempo decorrido

em seguida é criada uma instancia da classe *FileManager* e chamado o método *escreverPassagem()* para escrever os valores no ficheiro.
O *step()* é executado a cada intervalo de tempo sendo o *display* atualizado convenientemente.

```
@ScheduledMethod(start = 1, interval = 1)
 public void step() {
        //Definir uma variavel que contenha a localização deste agente
        final GridPoint local = grid.getLocation(this);


        // Procura a vizinhança do agente, por outros agentes
        // A função devolve uma lista de células com a vizinhança da
célula
        final List<GridCell<Consumidor>> consumerNeighborhood = new
GridCellNgh<Consumidor>(grid, local, Consumidor.class,
Config.VISAO_CONSUMIDOR,
Config.VISAO_CONSUMIDOR).getNeighborhood(false);


        // Devolve uma lista de células vazias
        final List<GridCell<Consumidor>> celulasLivres = Utils
                    .getFreeGridCells(consumerNeighborhood);

        // O agente não se move se não houver células livres na vizinhança
        if (celulasLivres.isEmpty()) {
              return;
        }
```

Código 3 - A função cíclica *Step()*

É necessário definir quando é que o método *step()* é chamado e com que frequência.

É usada a anotação *@ScheduledMethod* com dois parâmetros: start = 1, para definir o início da simulação como a altura de começar a executar o método *step()*, e interval=1 para definir a frequência com que o método deve ser executado.

Para mover o agente para uma nova célula, é criada uma lista de células à volta do agente.

O código para criar esta lista é:

```
final List<GridCell<Consumidor>> consumerNeighborhood = new
GridCellNgh<Consumidor>(grid, local, Consumidor.class,
Config.VISAO_CONSUMIDOR, Config.VISAO_CONSUMIDOR).getNeighborhood(false);
```

Código 4 - Criação de lista

Este código cria uma instância da classe *GridCellNgh* com os parâmetros:

- Grid – a referência da grelha onde se querem listar as células;
- Local – local (referencias) na grelha onde o agente está parado;
- Consumidor.class – o tipo de objetos incluídos na lista;
- *VISAO_CONSUMIDOR* – a amplitude de células a fazer parte da lista, ou seja quantas células a partir do centro*;*
- getNeighborhood – é o método para criar a lista de células vizinhas.

A lista devolvida é passada ao método *getFreeCells* que pertence à classe *Utils* e que nos devolve outra lista com apenas as células vazias através do código seguinte.

```
final List<GridCell<Consumidor>> celulasLivres = Utils
                        .getFreeGridCells(consumerNeighborhood);
```

Código 5 - Chamada do método *getFreeGridCells()*

Se não houver células vazias o agente não se move.

Em seguida é determinada uma célula vazia aleatória para mover para lá o agente.

O código seguinte move o agente para a nova célula:

```
final GridPoint newGridPoint = chosenFreeCell.getPoint();
grid.moveTo(this, newGridPoint.getX(), newGridPoint.getY());
```

Código 6 - Deslocação do agente para a nova célula

Depois do agente estar na nova célula vai-se tentar influenciar os agentes que estejam em células adjacentes. Volta-se a criar uma lista de células à volta do agente e procura-se as células ocupadas com outros agentes. Com base nesta informação inicia-se o processo de tentar influenciar os agentes que as ocupam, através do método *influenciar()* (Código 7).

O método influenciar é muito importante porque executa aquilo de que se trata, essencialmente, esta simulação – converter consumidores em utilizadores de um novo produto.

4.4.2.2 Influência dos agentes

Os agentes são influenciados através do método *influenciar().*

```
public void influenciar(List<GridCell<Consumidor>> consumerNeighborhood) {
  for ( GridCell <Consumidor> cell : consumerNeighborhood ) {
 if (cell.size() > 0 ) {
   final GridPoint gp = cell.getPoint();
   for (Object obj : grid.getObjectsAt(gp.getX(), gp.getY())) {
        if(obj instanceof Consumidor) {
          final Consumidor consumidor = (Consumidor) obj;
          double indice_influencia = (double)this.interesse_na_divulgacao *
          20 / 100 * (double)this.autoconfianca;
          double tickCount =
RunEnvironment.getInstance().getCurrentSchedule().getTickCount();
          consumidor.Afectar(indice_influencia, tickCount);
        }
   }
      } else {
 }
  }
}
```

Código 7 - O método *influenciar()*

É lida a lista à procura de agentes em cada célula.

Quando uma célula está ocupada é preciso saber se está ocupada por um agente do tipo *Consumidor*.

Em caso afirmativo é feito o cálculo do índice de influência e é chamado o método *Influenciar()* do agente a influenciar. Ao método é passado o índice de influência (Código 8) que será descontado na variável que contém a resistência do agente ao produto.

```
double indice_influencia = (double)this.interesse_na_divulgacao *
20 / 100 * (double)this.autoconfianca;
consumidor.Afectar(indice_influencia);
```

Código 8 - Cálculo do índice de influência

O método *Afectar()*

```
 public void Afectar(double indice) {
   if(this.resistencia_a_inovacao > 0.0) {
        this.resistencia_a_inovacao -= indice;
        if (this.resistencia_a_inovacao < 0.0)
this.resistencia_a_inovacao = 0.0;
        if(this.resistencia_a_inovacao == 0.0) {
          double tick =
RunEnvironment.getInstance().getCurrentSchedule().getTickCount();
          FileManager fm = new FileManager();
          fm.escreverPassagem(this.outputFile, getID(), tick);
          this.tipo_consumidor=1;
        }
   }
 }
```

Código 9 - O método *Afectar()*

O método *Afectar()* (Código 9) é chamado quando o agente é influenciado pelo vizinho para atualizar o índice de influência sobre *resistência_a_inovacao.* Se a variável atingir o valor zero o agente passa a ser um consumidor que já aderiu ao produto e é registado o instante em que se dá esta mudança sendo chamado o método *escreverPassagem(),* da classe *FileManager,* com indicação do ficheiro de saída, ID do agente e instante em que se deu a transição.

```
double tick =
RunEnvironment.getInstance().getCurrentSchedule().getTickCount();
FileManager fm = new FileManager();
fm.escreverPassagem(this.outputFile, getID(), tick);
this.tipo_consumidor=1;
```

Código 10 - Registo, da passagem do agente a consumidor, no ficheiro

4.4.2.3 A classe FileManager

Manipulação de ficheiros

Esta classe contém todos os métodos para tratar de todos os acessos a ficheiros, tanto para escrita como para leitura.

Sempre que for necessário manipular ficheiros o ideal é criar uma classe que trata de todos os procedimentos bastando, para isso, apenas passar-lhe alguns parâmetros.

Esta classe é usada para ler um ficheiro com alguns parâmetros de configuração da simulação e para escrever num ficheiro a altura no tempo em que um agente passa a ser um consumidor do novo produto.

Tabela 10 - Ficheiro com parâmetros de configuração da simulação

Nome da variável	Valor
Nº de agentes	10
Nº de iniciadores	5
Local para os ficheiros	c:\Downloads

Na Tabela 10 pode-se ver uma amostra do ficheiro usado para configuração de alguns parâmetros da simulação como por exemplo: o número inicial de agentes, quantos agentes são iniciadores e a pasta onde será gravado o ficheiro de registo da transição dos agentes para consumidores mas muitos mais poderão ser acrescentados desde que seja acrescentado, na simulação, o respetivo código para tratamento da informação.

```
public ArrayList<ConfigData> lerFicheiro() {
 final ArrayList<ConfigData> ret = new ArrayList<ConfigData>();
 BufferedReader br = null;
        try {
               br = new BufferedReader(new FileReader(this.file));
               br.readLine();
               String linha = null;
               while ((linha = br.readLine()) != null) {
                      ret.add(new ConfigData(linha));
               }
        } catch (final FileNotFoundException e) {
               e.printStackTrace();
        } catch (final IOException e) {
               e.printStackTrace();
        } finally {
               if (br != null) {
                      try {
                             br.close();
                      } catch (final IOException e) {
                             e.printStackTrace();
                      }
               }
        }
        return ret;
}
```

Código 11 - O método *lerFicheiro()*

O método *lerFicheiro()* (Código 11)devolve um *ArrayList* com todos os dados do ficheiro.

```
public class ConfigData {
 final String linha;
 private String[] data;
 public ConfigData(final String linha) {
        this.linha = linha;
        splitData();
 }
 public void splitData() {
        // Separa a linha em vários parâmetros separados por ‘;’
        this.data = this.linha.split(";");
 }
 public String getValor() {
        return data[1];
 }
 public String getNome() {
        return data[0];
 }
}
```

Código 12 - A classe *ConfigData*

Finalmente, o código seguinte (Código 13) lê linha a linha do ficheiro e vai juntando-as ao *ArrayList* definido com o nome *ret.*

```
BufferedReader br = null;
        try {
               br = new BufferedReader(new FileReader(this.file));
               br.readLine();
               String linha = null;
               while ((linha = br.readLine()) != null) {
                      ret.add(new ConfigData(linha));
               }
```

Código 13 - Leitura linha a linha do ficheiro de configuração

O outro método definido na classe é o método *escreverPassagem()* que regista, em um ficheiro, a altura em que o agente passou a ser um consumidor e que já foi, anteriormente, mencionado.

É *instanciada* a classe que contém as funcionalidades de escrita/leitura de ficheiros e é, então, descarregada a informação contida na variável compilada anteriormente, no ficheiro, sendo, também escrita uma nova linha para que o texto, no ficheiro, esteja com uma formatação que nos proporcione uma melhor leitura (Código 14).

```
writer = new BufferedWriter( new FileWriter(fileName, true));
            writer.newLine();
            writer.write(texto);
```

Código 14 - Escrita da variável no ficheiro

4.4.3 Configurações

Classes com definições básicas da simulação

A classe *Config* apenas define algumas constantes de configuração da simulação.

```
public static final String CONTEXT_ID = "NetworkSpread";
 public static final String SPACE_ID = "space";
 public static final String GRID_ID = "grid";

 public static final int GRID_SIZE = 50;
 public static final int NUM_CONSUMIDORES = 50
 public static final int VISAO_CONSUMIDOR = 1;

 public static final String AGENT_START_NUM = "agentStartNum";

 public static final String FORCA_INFLUENCIA_MINIMO =
"forcaInfluencia_min";
 public static final String FORCA_INFLUENCIA_MAXIMO =
"forcaInfluencia_max";

 public static final String RESISTENCIA_INOVACAO_MINIMO = "restInov_min";
 public static final String RESISTENCIA_INOVACAO_MAXIMO = "restInov_max";

 public static final String INTERESSE_DIVULGACAO_MINIMO = "intDivulg_min";
 public static final String INTERESSE_DIVULGACAO_MAXIMO = "intDivulg_max";
```

Código 15 - Constantes definidas para a simulação

Tabela 11 – Descrição das constantes iniciais

Nome da constante	Descrição
CONTEXT_ID	Nome do contexto
SPACE_ID	Nome do espaço
GRID_ID	Nome da grelha
GRID_SIZE	Tamanho para a grelha
NUM_CONSUMIDORES	Número total de consumidores
VISAO_CONSUMIDOR	Alcance do consumidor à sua volta
AGENT_START_NUM	Número de iniciadores
FORCA_INFLUENCIA_MINIMO	Força mínima de influência
FORCA_INFLUENCIA_MAXIMO	Força máxima de influência
RESISTENCIA_INOVACAO_MINIMO	Resistência mínima do agente à inovação
RESISTENCIA_INOVACAO_MAXIMO	Resistência máxima do agente à inovação
INTERESSE_DIVULGACAO_MINIMO	Valor mínimo do interesse do agente em divulgar o produto
INTERESSE_DIVULGACAO_MAXIMO	Valor mínimo do interesse do agente em divulgar o produto

A classe *Utils* é usada para definir alguns métodos de utilidade geral da simulação e que podem ser usadas por qualquer outra classe

O método *getFreeCells*():

```
public static <T> List<GridCell<T>> getFreeGridCells(
 final List<GridCell<T>> neighborhood) {
        if (null == neighborhood) {
                throw new IllegalArgumentException(
                        "Parameter neighborhood cannot be null.");
        }

        final ArrayList<GridCell<T>> ret = new ArrayList<GridCell<T>>();

        for (final GridCell<T> act : neighborhood) {
                if (0 == act.size()) {
                        ret.add(act);
                }
        }

        return ret;
 }
```

Código 16 - O método *getFreeCells()* da classe *Utils*

O método *getFreeCells()* (Código 16) recebe uma lista de células e devolve uma nova lista com as células desocupadas.

O método *randomElementOf ():*

```
public static <T> T randomElementOf(final List<T> list) {
        if (null == list) {
               throw new IllegalArgumentException("A lista de parametros
não pode ser null.");
        }
        if (list.isEmpty()) {
               throw new IllegalArgumentException(
                              "Não é possível devolver uma célula aleatória
de uma lista vazia.");
        }

        return list.get(RandomHelper.nextIntFromTo(0, list.size() - 1));
 }
```

Código 17 - O método *randomElementOf()* da classe *Utils*

O método *randomElementOf()* (Código 17) recebe uma lista de células e devolve uma célula retirada aleatoriamente dessa lista. Este método é chamado para se mover o agente para uma célula vazia à sua volta. Desta forma, é assegurado que o agente não se move sempre na mesma direção.

4.4.3.1 A classe *NetWorkSpreadContext*

Inicialização da simulação criando um contexto.

Criar um contexto é preenchê-lo com agentes.

A simulação é inicializada através de uma classe que implementa o *interface ContextBuilder<T>*, sendo o *<T>* os tipos de agentes adicionados ao contexto.

Esta classe tem como função construir e devolver um contexto, definindo *ContextBuilder<T>* como *interface.*

```
public class NetWorkSpreadContextBuilder extends DefaultContext<Object>
implements     ContextBuilder<Object> {
@Override
 public Context<Object> build(final Context<Object> context) {
 }
}
```

Código 18 - A classe *NetWorkSpreadContextBuilder*

O contexto é constituído por um espaço contínuo *(ContinuousSpace)* e uma grelha *(Grid)*. Esta grelha vai ser onde os agentes se posicionam.

Os contextos podem conter projeções. As projeções associam regras aos elementos contidos no contexto.

Para criar um espaço, o procedimento é semelhante à criação de uma grelha (Código 19). Para se criar um espaço é necessário:

- um ID único;
- o *context* para associar o espaço;
- uma instância da classe *Adder* para definir como adicionar novos agentes ao espaço. Neste caso usa-se *RandomCartesianAdder* para se adicionar novos agentes aleatoriamente;
- uma instância de *PointTranslator* para se definir como o espaço deve lidar com o agente quando ele chega ao fim da grelha. Neste caso usa-se *WrapAroundBorders* para que o agente, quando sair de um lado do espaço entrar no espaço pelo lado oposto ao da saída como se o espaço fosse circular;
- por fim, o tamanho do espaço através da definição dos pontos X e Y. Neste caso estes valores são iguais porque estamos a definir um espaço quadrado.

```
final ContinuousSpace<Object> space = ContinuousSpaceFactoryFinder
                      .createContinuousSpaceFactory(null)
                      .createContinuousSpace(
                                    Config.SPACE_ID,
                                    context,
                                    new RandomCartesianAdder<Object>(),
                                    new
repast.simphony.space.continuous.WrapAroundBorders(),
                Config.GRID_SIZE, Config.GRID_SIZE);
```

Código 19 - Criação do espaço contínuo

A criação da grelha é muito parecida com a criação do espaço mas usando instâncias diferentes (Código 20). Para o espaço usa-se uma instância da classe *repast.simphony.space.continuous.WrapAroundBorders* e para a grelha usa-se uma instância da classe *repast.simphony.space.grid.WrapAroundBorders*. Uma outra diferença é que na criação da grelha usa-se uma instância da classe GridBuilderParameters para atribuição de parâmetros.

```
final Grid<Object> grid = GridFactoryFinder
                      .createGridFactory(null)
                      .createGrid(
                      Config.GRID_ID,
                      context,
                      new GridBuilderParameters<Object>(
  new repast.simphony.space.grid.WrapAroundBorders(),
  new SimpleGridAdder<Object>(),true,
        Config.GRID_SIZE, Config.GRID_SIZE));
```

Código 20 - Criação da grelha

A adição de agentes ao contexto, conforme ilustrado no exemplo de Código 21, é feito através de um processo repetitivo em que são criadas instância do agente e adicionadas ao contexto e onde lhe é indicado o local onde se vai posicionar.

```
for (int i = 0; i < numConsumidores; ++i) {
final Consumidor consumidor = new Consumidor(grid);
 context.add(consumidor);
 final NdPoint pt = space.getLocation(consumidor);
 grid.moveTo(consumidor, (int) pt.getX(), (int) pt.getY());
}
```

Código 21 - Adicionar agentes ao contexto

O método *context.add(consumidor)* é que efetivamente adiciona o novo agente ao contexto, em seguida vai-se buscar a posição do agente no espaço e converte-se as coordenadas da sua posição em coordenadas na grelha, definindo essa como a posição corrente.

4.4.3.2 Definição e leitura de parâmetros

A parametrização da simulação é feita com base no conjunto de atributos indicados na Tabela 12.

Tabela 12 - Lista de parâmetros

Nome do parâmetro	**Descrição**
Name	O nome único interno que se quer dar ao parâmetro
Display name	O nome que se quer exibir como nome do parâmetro
Type	O tipo de parâmetro que se está a criar e pode ser do tipo *int*, *long*, *double*, ou *string*
Default Value	O valor que se atribui por defeito
Converter	Opcional. Pode converter tipos não-normalizados de e para *string*. Não é necessário a não ser que se use outro tipo que não seja *int*, *long*, *double*, ou *string*
Values	Uma lista de valores separados por espaços e do tipo escolhido que condicionam os valores a usar
Read Only	Se se quiser que o parâmetro não seja alterado

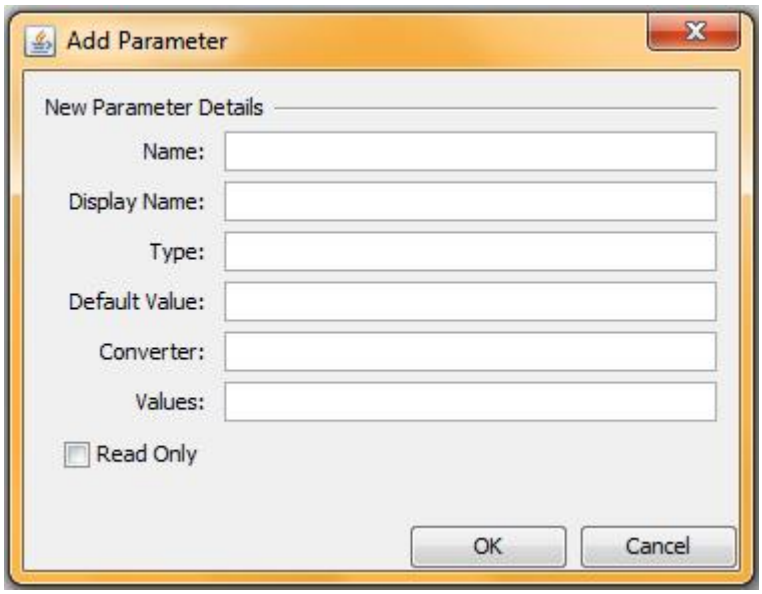


Figura 13 - Definição de parâmetros

Estes parâmetros são definidos num ficheiro chamado *parameters.xml* que está nos recursos do projeto e guarda todos os parâmetros que se definir num formato chamado XML (Código 22).

```
<parameter name="forcaInfluencia_min"
displayName="Forca de influencia (minimo)"
type="int"
 defaultValue="1"
 isReadOnly="false"
converter="repast.simphony.parameter.StringConverterFactory$IntConverter"
/>
```

Código 22 - Definição de um parâmetro no ficheiro *parameters.xml*

A leitura dos parâmetros é feita na classe NetWorkSpreadContext no instante da inicialização.

A configuração da simulação pode ser feita de forma alternativa através da especificação dos parâmetros num ficheiro externo em formato de texto. Foi usado um ficheiro com extensão *.csv* (*comma-separated values)* porque tem a vantagem de ser facilmente editado pelo *Microsoft Excel* sendo fácil de manipular.

Tabela 13 - Conteúdo do ficheiro externo de configuração de parâmetros

Nome dos campos	**Valores**
Nome da variável	Valor
Nº de agentes	10
Nº de iniciadores	5
Local para os ficheiros	c:\Downloads

Ao ficheiro externo chamou-se *datafile.csv* e consiste no número de linha que se quiser inserir mas apenas com duas colunas. Na primeira, o nome da variável e na segunda o valor dessa variável.

Os parâmetros do ficheiro são lidos para uma variável do tipo *ArrayList* chamando o método *lerFicheiro()* definido na nossa classe para manipulação de ficheiros *FileManager.*

4.4.3.3 Ficheiro de saída

Para melhor compreensão da simulação foi criado um ficheiro de saída com a informação sobre a altura da simulação em que cada agente é convertido de potencial consumidor para consumidor de facto.

O nome do ficheiro é constituído por:

- as palavras “Instante de Adesao”;
- a data da simulação (ex: 07-01-2013);
- um conjunto de três números gerados aleatoriamente de forma a que não haja dois ficheiro iguais.

O tipo de ficheiro é .csv (*comma-separated values*) que como já se viu anteriormente tem um formato que possibilita a sua abertura com uma aplicação como o *Microsoft Excel.*

Este ficheiro apenas tem duas colunas:

- o *ID* do agente;
- e a altura em que ele foi convertido.

```
final String novoConsumidorFile = ar.get(2).getValor() + "\\" +
"PontoDeViragem_" + getDate() + "-" + getRandomPart()+".csv";
```

Código 23 - Geração do nome do ficheiro

4.4.3.4 Criação de um gráfico

É possível criar um gráfico com a evolução da simulação.

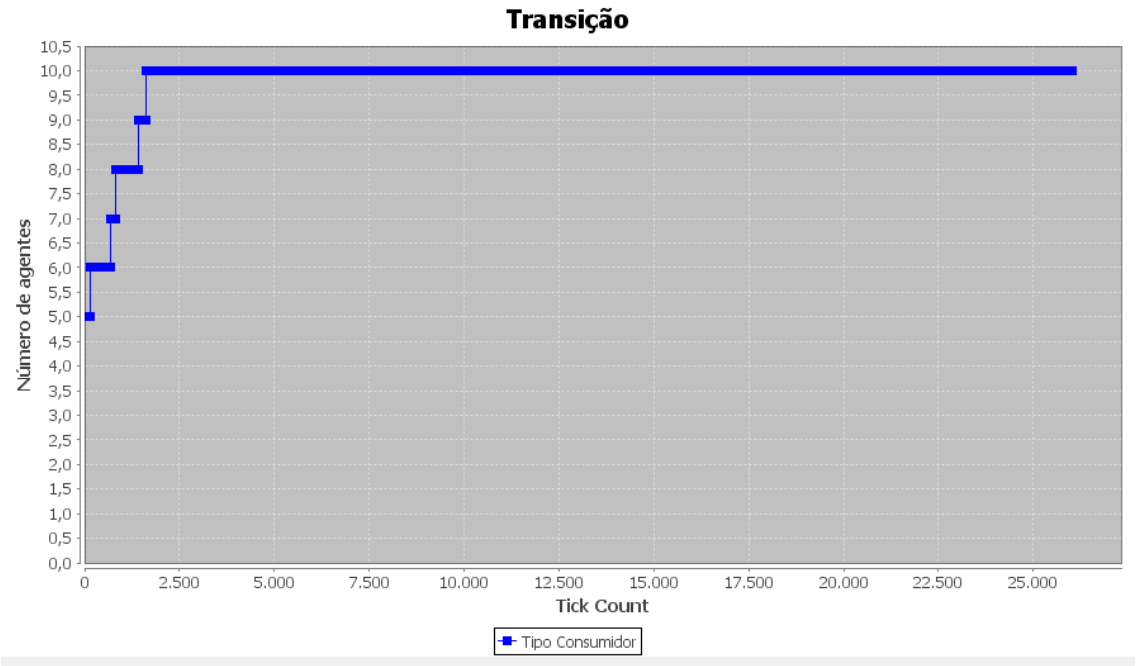


Figura 14 - Gráfico gerado com a transição dos agentes

Para a criação de um gráfico tem-se que configurar, no *interface* do *Repast Simphony,* a informação que vai ser usada para este fim.

Começa-se por criar um *Data Set.* Um *Data Set* é um ficheiro de dados criado pela simulação. Esses dados vão ser a fonte de informação para a criação do gráfico. Quando a simulação começar a correr seleciona-se o separador *Scenario Tree* e carrega-se com o botão direito do rato sobre a opção *Data Sets,* conforme *se* pode ver na Figura 15.

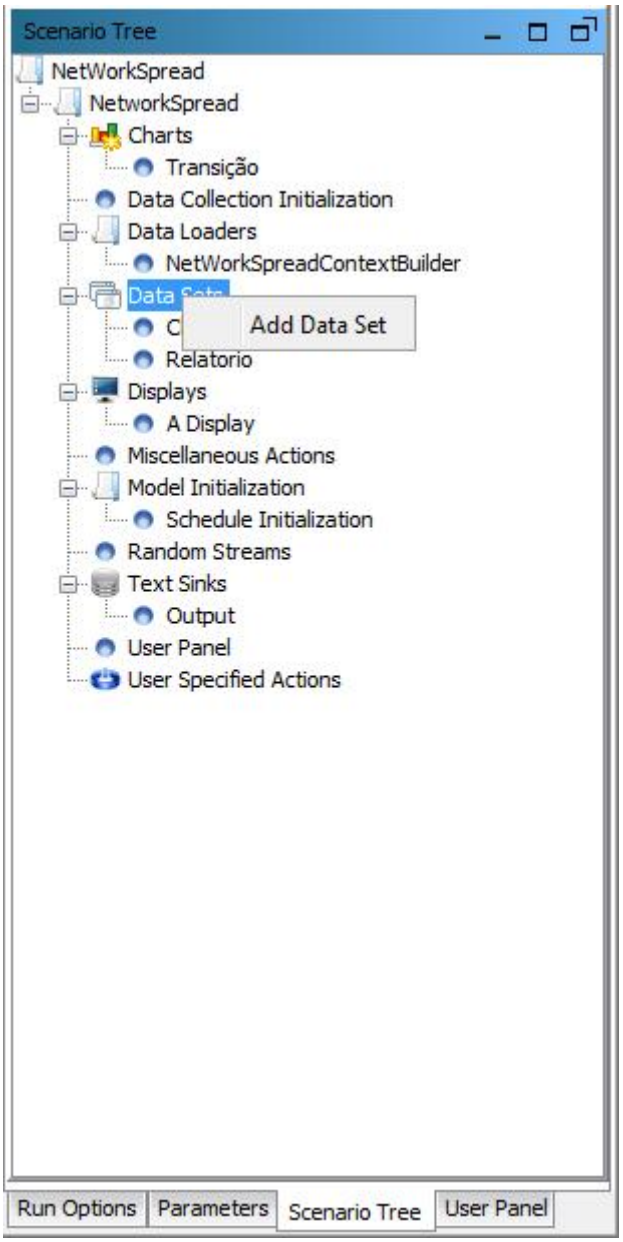


Figura 15 - Seleção da opção Data Sets

Na janela que é apresentado de seguida define-se, entre outras coisas, a fonte dos dados para a criação do gráfico.

Começa-se por dar um nome ao *Data Set*. No campo *Data Set Id* insere-se o nome *Relatório*.

No campo *Data Set Type* seleciona-se a opção *Non-Aggregate* porque não se quer que seja usada uma soma dos valores em causa.

Carrega-se em *Next* e seleciona-se a opção *Tick.* Isto diz à simulação para criar valores a cada tique do relógio.

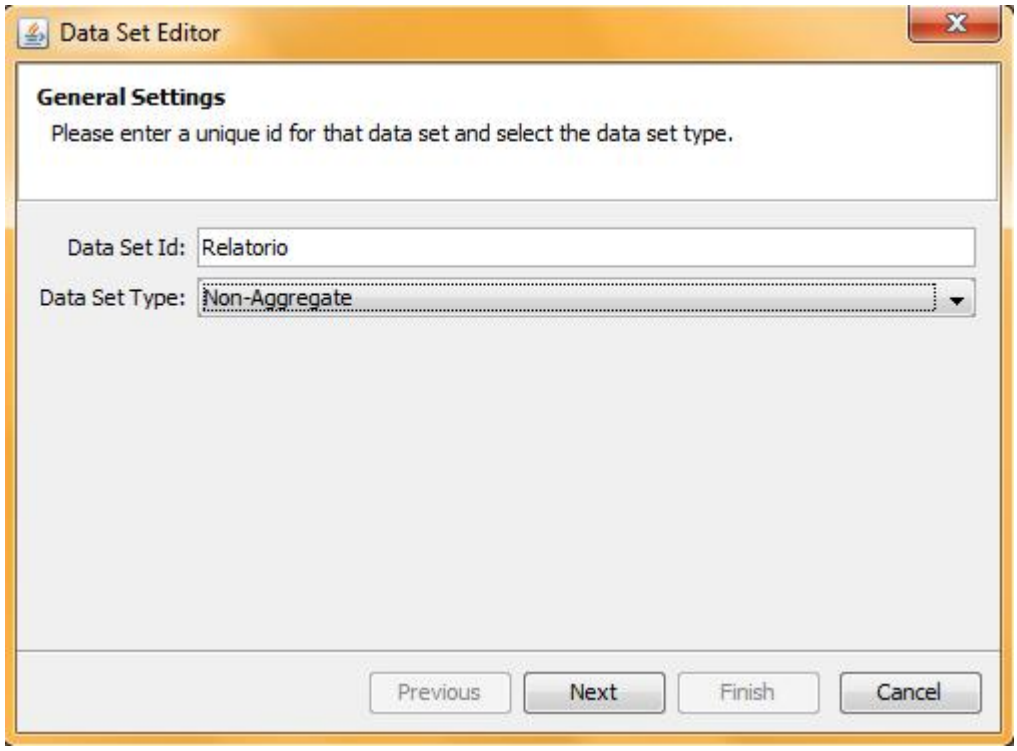


Figura 16 - Criação do Data Set

A seguir, no separador *Method Data Sources,* seleciona-se como *source class* a classe consumidor, porque é aquela onde se quer ir buscar os valores para o gráfico.

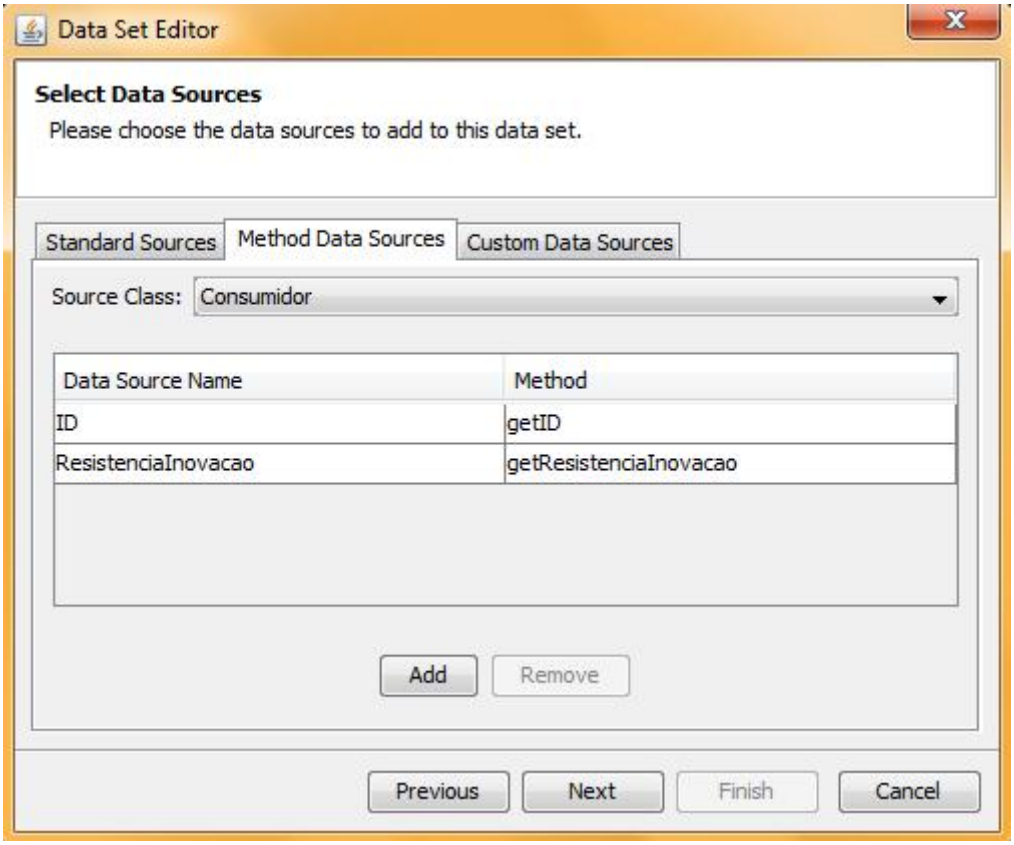


Figura 17 - Indicação dos métodos fonte de informação

No botão *Add* insere-se os métodos que se criaram na classe *Consumidor* para devolver valores.

Neste caso criaram-se dois métodos:

- *getID()* que devolve a identificação única do agente que está a dar a informação e o método;
- *getResistenciaInovação()* que devolve o valor da resistência à inovação que naquele momento aquele agente tem.

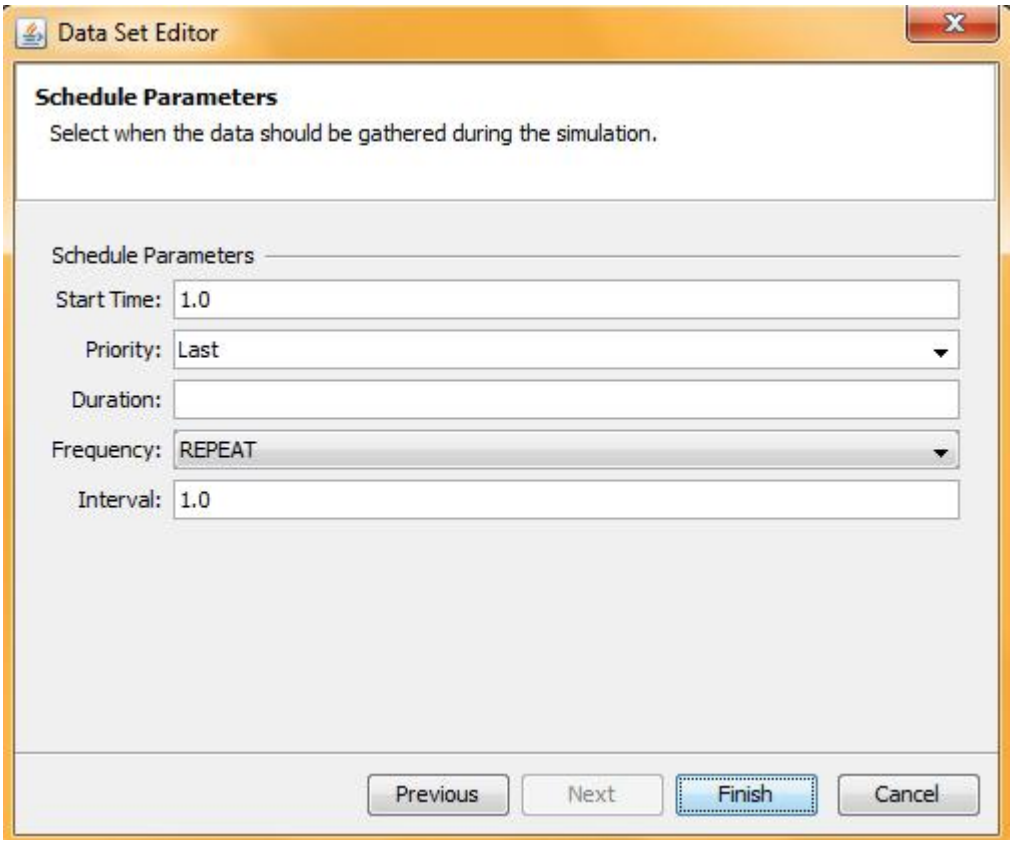


Figura 18 - Quando a informação deve ser recolhida

No último quadro diz-se quando a informação deve ser recolhida:

- *Start Time = 1.0* diz para a informação começar a ser recolhida no primeiro tique do relógio;
- *Priority = Last* diz que a prioridade do *thread* que recolhe a informação é mínima;
- *Duration* esta opção pode ficar em branco;
- *Frequency* que define a frequência com que a informação é recolhida pode ser ONE_TIME (uma vez) ou então *REPEAT* ou seja repetidamente;
- *Interval = 1.0* diz com que frequência essa informação deve ser recolhida. Neste caso a cada tique do relógio.

Com isto se termina a criação do Data Set. Carrega-se no botão *Finish* e está pronto.

Finalmente, definem-se os parâmetros para o gráfico. Um pouco mais acima, onde se selecionou a opção Data Sets, seleciona-se com o botão direito do rato a opção *Charts* e depois a opção *Add* Time Series *Chart*.

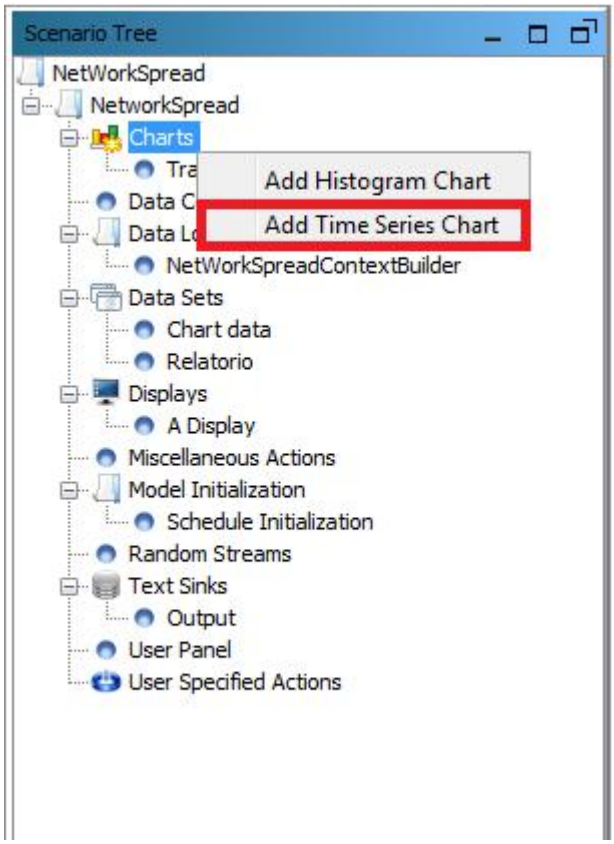


Figura 19 - Seleção do tipo de gráfico

No quadro seguinte, no campo *Name* dá-se o nome ao gráfico. Como o gráfico representa o momento da transição dos agentes para consumidores, chamou-se ao gráfico *Transição*.

No campo *Data Set* tem-se que indicar a fonte de dados para a construção do gráfico. Aqui seleciona-se a opção *Relatorio* porque foi o que se definiu no *Data Set.* Em seguida carrega-se no botão *Next.*

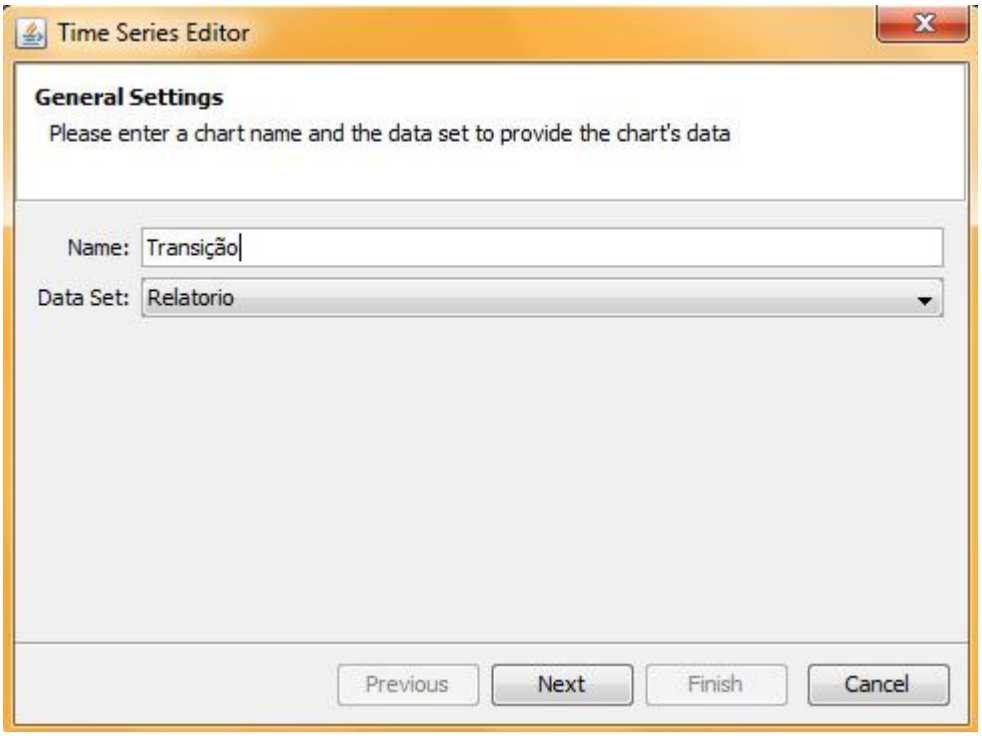


Figura 20 - Definição do nome e do Data Set para o gráfico

Na próxima janela seleciona-se como ID da série a opção ID já anteriormente definida no *Data Set* e como *Data To Display* (dados a mostrar) a opção *ResistenciaInovacao* também já anteriormente definido no *Data Set.*

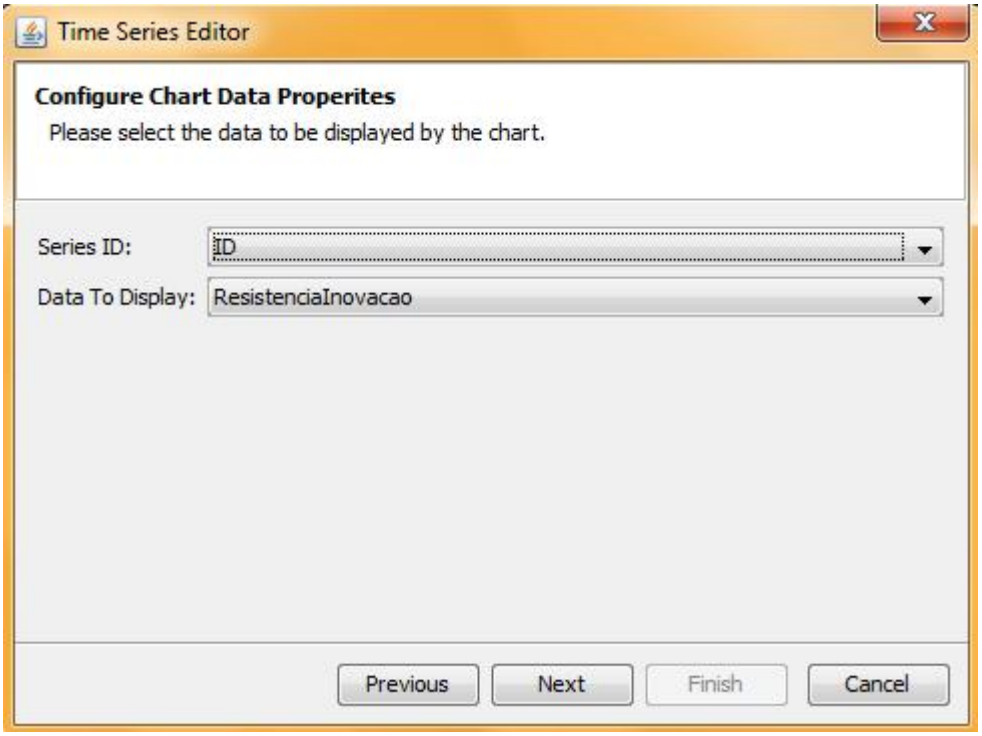


Figura 21 - Configuração das propriedades dos dados do gráfico

Finalmente, depois de se ter selecionado o botão *Next* seleciona-se na janela seguinte cada uma das opções com os dados.

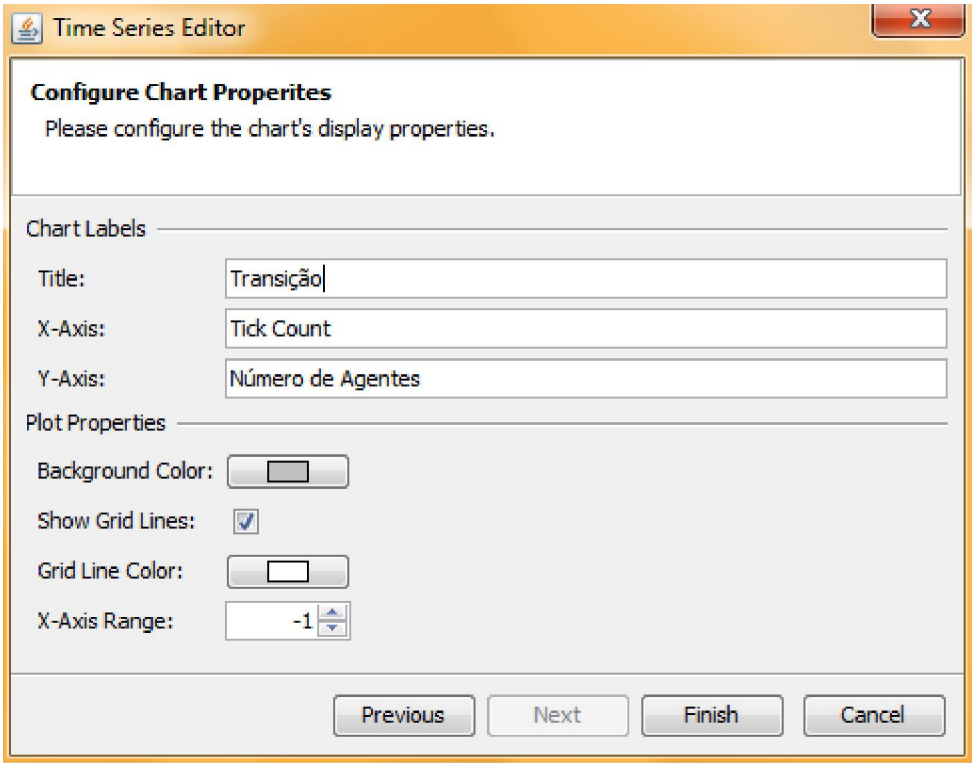


Figura 22 - Propriedades do gráfico

Para o título do gráfico escreveu-se *Transição*, para o nome do eixo dos XX deixou-se ficar o nome por defeito, isto é, *Tick Count* e para título do eixo dos YY escreveu-se *Número de Agentes.* Pode-se, se assim se desejar, selecionar uma cor diferente para o fundo e para as linhas da grelha do gráfico assim como se se quer ver as linhas da grelha, ou não.

De seguida carrega-se em *Finish* e tem-se as configurações prontas.

Põe-se a simulação a correr e pode-se ver o gráfico no separador em baixo que diz *Transição.*

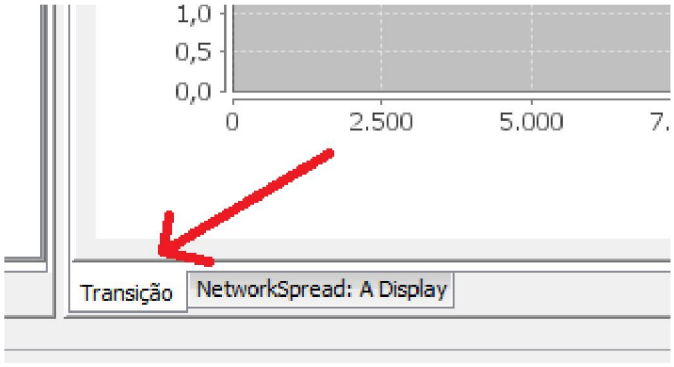


Figura 23 - O separador que contém o gráfico

4.5 Cenários de estudo

Em seguida podemos ver a imagem de uma simulação com 100 agentes: 5 iniciadores e 95 seguidores.

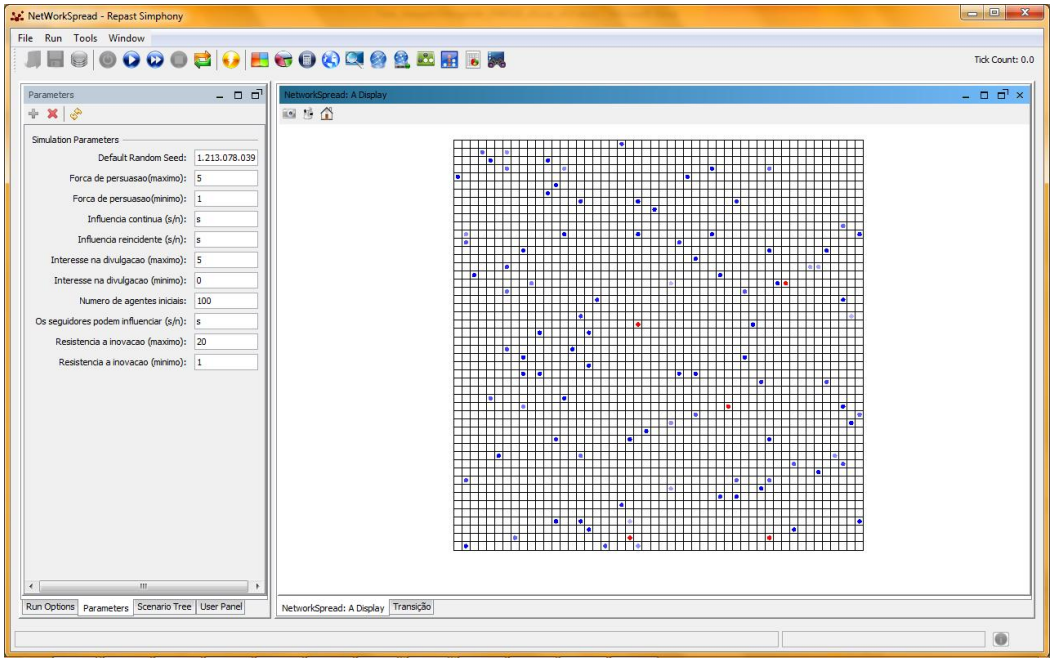


Figura 24 - O interface de simulação antes de esta ser executada

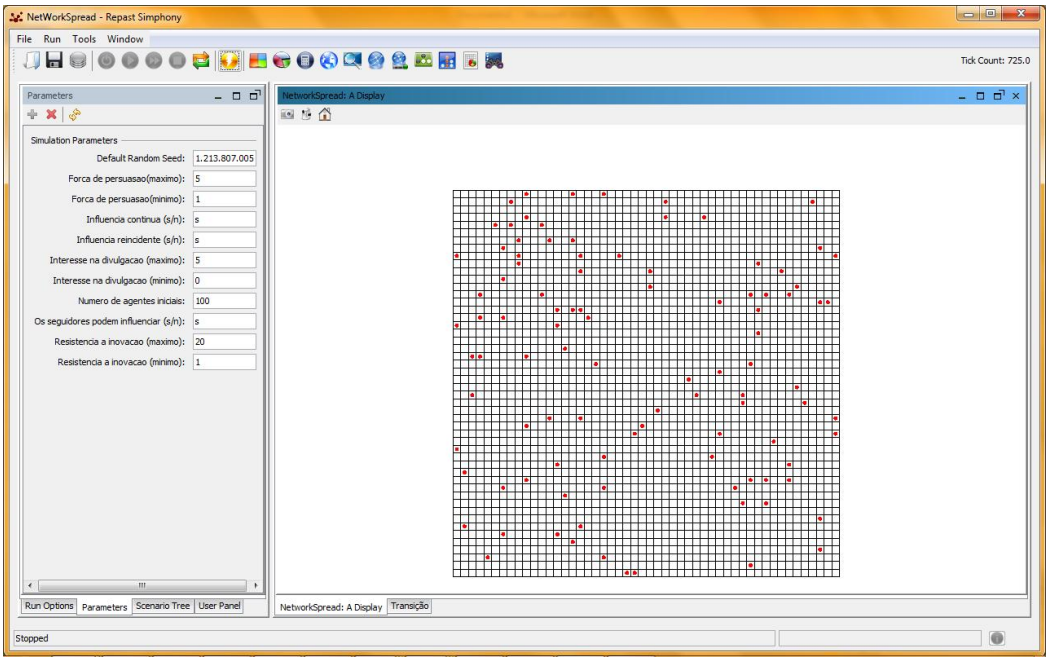
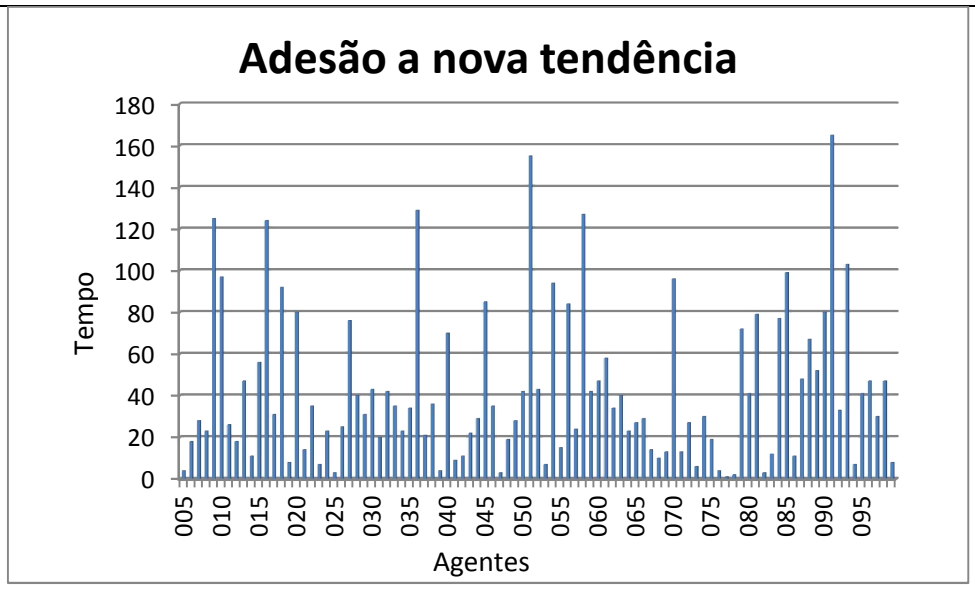


Figura 25 - O interface da simulação depois de esta ser executada

Na Figura 24 podemos ver todos os agentes iniciadores representados a vermelho (5 iniciadores) e os agentes seguidores (95 seguidores) representados a azul.

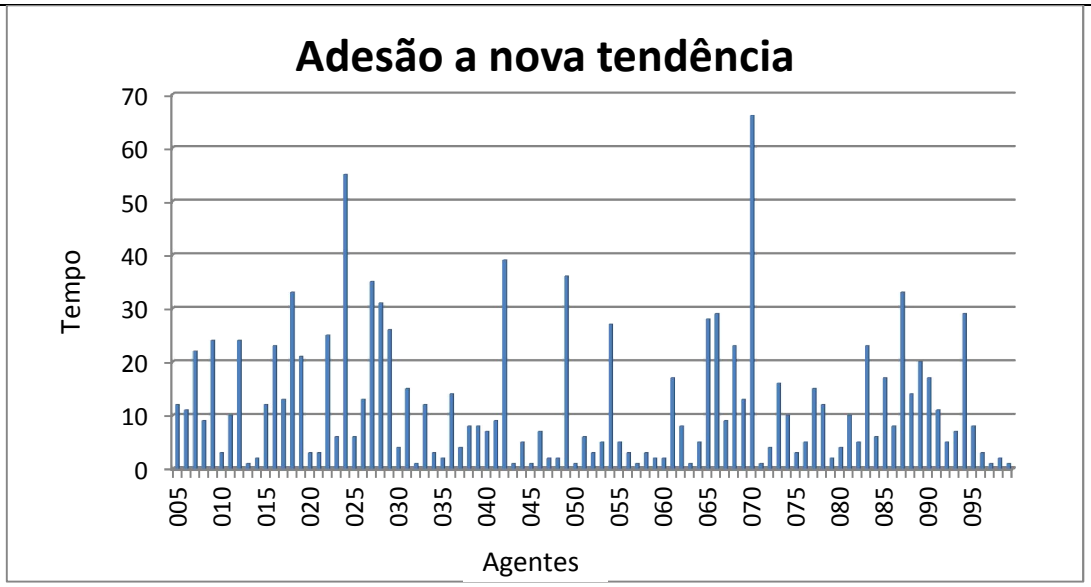
Na Figura 25 podemos observar que todos os agentes estão representados a vermelho porque todos foram já convertidos a consumidores.

Nos gráficos em baixo podemos ver o resultado de quatro simulações, de uma rede em malha, que foram feitas com 100 agentes mas variando os outros parâmetros, nomeadamente a *persuasão,* o *interesse na divulgação* e a *resistência.* Foram simulados quatro cenários correspondendo a quatro situações que se acharam pertinentes: o cenário 1 corresponde ao cenário base, em que os parâmetros foram deixados na sua máxima amplitude. No cenário 2, a persuasão e interesse foram aumentados para o máximo possível dos seus valores enquanto a resistência pode variar entre os valores mínimo e máximo, no cenário 3 todos os valores estão no máximo e finalmente no cenário 4 a persuasão e o interesse estão no mínimo e a resistência está no máximo.



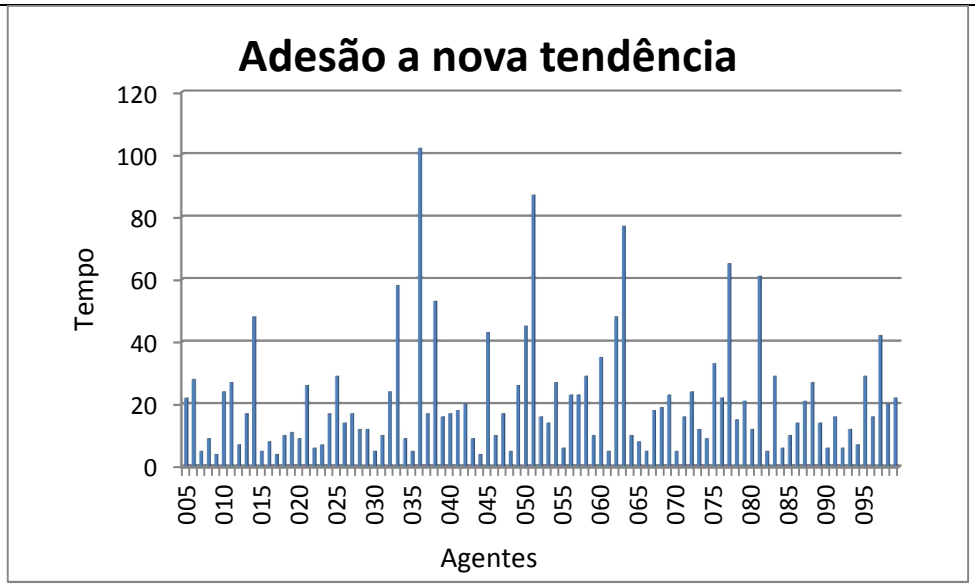
Agentes:	100
Persuasão (min.-max.):	1-5
Interesse na divulgação (min.-max.):	0-5
Resistência (min.-max.)	1-20

Figura 26 - Cenário 1: parâmetros iniciais na sua amplitude máxima



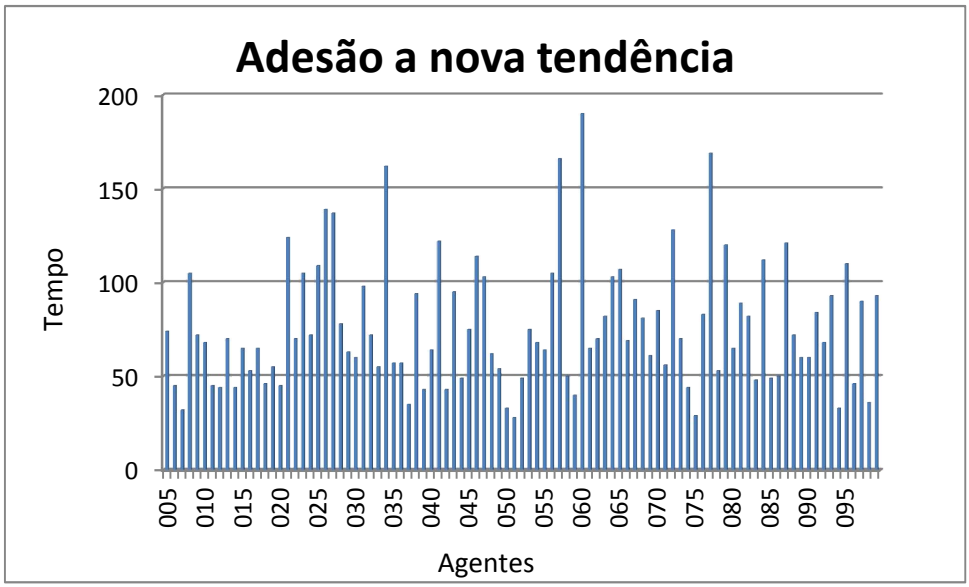
Agentes:	100
Persuasão (min.-max.):	5-5
Interesse na divulgação (min.-max.):	5-5
Resistência (min.-max.)	1-20

Figura 27 - Cenário 2: máxima persuasão e interesse



Agentes:	100
Persuasão (min.-max.):	5-5
Interesse na divulgação (min.-max.):	5-5
Resistência (min.-max.)	20-20

Figura 28 - Cenário 3: persuasão, interesse e resistência no seu valor máximo



Agentes:	100
Persuasão (min.-max.):	1-1
Interesse na divulgação (min.-max.):	1-1
Resistência (min.-max.)	20-20

Figura 29 - Cenário 4: persuasão e interesse no seu valor mínimo e resistência no seu valor máximo

Note-se que dos 100 agentes, 5 são os iniciadores, logo não estão representados nestes gráficos que se referem apenas aos seguidores. Foram definidos estes valores para as variáveis com o intuito de obter uns resultados diversificados e que abrangessem todo o espectro de possibilidades definindo situações como persuasão bastante variável e resistência bastante variável, máximo de persuasão e resistência bastante variável, máximo de persuasão e máximo de resistência e mínimo de persuasão e máximo de resistência.

Na Figura 26 temos uma amostra que abrange todo o espectro de possibilidades definidas ou seja a *persuasão, o interesse na divulgação* e a *resistência* vão desde o mínimo definido (0 ou 1) até ao máximo possível (5 ou 20) definindo um conjunto de indivíduos com características heterogéneas, isto é, com diversos níveis de interesse em divulgar em poder de persuasão e predisposição à influencia de novos produtos.

Como resultado podemos observar que temos uma distribuição na linha do tempo bastante diversificada, com agentes a serem convertidos em consumidores em toda essa linha em tempos bastante diferentes (a variabilidade temporal é grande), mas o que tem maior relevância é que o maior tempo usado para converter um possível consumidor num consumidor efetivo foi de 165 unidades de tempo e tivemos 14 agentes a serem convertidos depois de metade do tempo máximo ter ocorrido.

Na Figura 27 temos uns iniciadores empenhados ao máximo na divulgação do produto e com um nível de credibilidade também no máximo. Por outro lado, temos uns possíveis novos

clientes com uma predisposição em aderir ao novo produto ou tendência distribuídos por toda a escala (1 a 20). Resumindo temos uns iniciadores com características homogéneas e uns seguidores com características heterogéneas. Aqui a maior entrave à adesão será a resistência dos seguidores.

Como resultado, podemos observar pelo gráfico que o último agente foi convertido ao fim de apenas 66 unidades de tempo, menos de metade do tempo decorrido até à conversão do último agente na Figura 26, tendo 88 agentes sido convertidos antes de metade do tempo decorrido e apenas 7 foram convertidos depois de metade do tempo máximo ter decorrido.

Na Figura 28 todos parâmetros estão nos seus valores máximos. Nestes resultados, podemos observar uma tendência muito interessante, que é o facto de apesar de a resistência à inovação dos agentes a influenciar (seguidores) estar no máximo e os agentes influenciadores estarem, também, com o empenho e a credibilidade definidos para o valor máximo, conseguiu-se que o máximo de tempo necessário à conversão de todos os agentes fosse de 102 unidades de tempo.

Podemos também observar que 88 dos agentes foram convertidos até metade do tempo total que demorou o último agente a ser convertido e os restantes 7 dos 95 agentes representados no gráfico só foram convertidos na segunda metade do total de tempo decorrido. Assim podemos deduzir que, mesmo com grande resistência, o tempo que demora a influenciar um grupo de agentes, quando há um grande empenho por parte dos influenciadores, não apresenta um resultado final muito diferente daquele que se obtém quando temos uns influenciadores muito empenhados e uns seguidores com um leque de possibilidades em aderir que varia entre o mínimo e o máximo dos valores definidos.

Na Figura 29 temos uns influenciadores com o empenho em influenciar reduzido ao mínimo, e uns seguidores com o máximo de resistência possível em aderir.

Neste caso os resultados são muito interessantes porque podemos ver que a primeira conversão só foi feita ao fim de 28 unidades de tempo e o tempo total que levou a influenciar todos os agentes seguidores foi de 190 unidades de tempo. Apenas 72 agentes foram convertidos na primeira metade do tempo decorrido tendo os restantes 23 agentes sido convertidos apenas na segunda metade do tempo total que levou a converter todos os agentes.

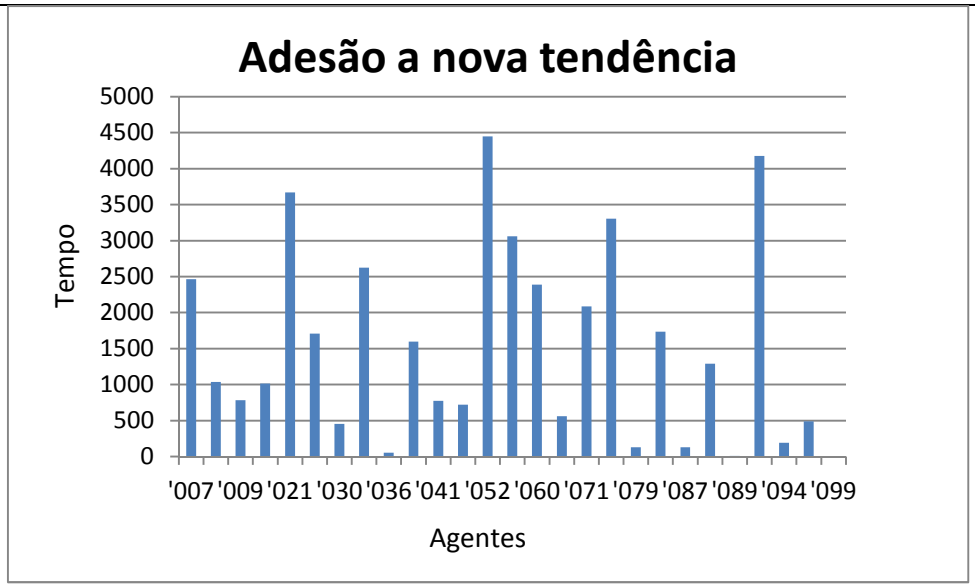
Será importante lembrar que, nestas simulações, todos os agentes convertidos passam também a tentar influenciar os agentes ainda não convertidos, o que acelera a simulação.

Estes valores para os parâmetros da simulação foram escolhidos como exemplo da diversidade de condições que podem ser definidas e da maneira como essas condições podem influenciar o resultado final.

Para comparação, apresenta-se, de seguida, os resultados da simulação com os mesmos parâmetros mas numa rede em estrela.

Agentes:	100
Persuasão (min.-max.):	1-5
Interesse na divulgação (min.-max.):	0-5
Resistência (min.-max.)	1-20

Figura 30 - Cenário 1: parâmetros iniciais na sua amplitude máxima



Agentes:	100
Persuasão (min.-max.):	5-5
Interesse na divulgação (min.-max.):	5-5
Resistência (min.-max.)	1-20

Figura 31 - Cenário 2: máxima persuasão e interesse

Pode-se observar pelos gráficos acima que, logo à partida, o que chama a atenção é a diferença de valores, no eixo do tempo, e no número de agentes, no eixo dos agentes. Deve-se notar que, nestas simulações de rede em estrela, apenas um agente, o iniciador, divulga a informação. Os restantes agentes influenciados, e segundo as características de uma rede desta natureza, que tem apenas um influenciador central, não influenciam os outros agentes. No cenário 1 da rede em malha, em que todos os parâmetros iniciais estão na sua amplitude máxima, podemos ver que, ao fim de pouco mais de 160 unidades de tempo, todos os agentes foram convertidos em consumidores, enquanto que, na rede em estrela coma mesma configuração de parâmetros, depois de deixar a simulação executar durante cerca de 7500 unidades de tempo, foram precisos 3914 unidades de tempo para converter apenas 8 seguidores, de 99, em consumidores.

No cenário 2 da rede em malha, em que temos a persuasão e o interesse na divulgação nos seus valores máximos, enquanto que a resistência dos seguidores tem uma amplitude máxima, podemos observar que, ao fim de 66 unidades de tempo, todos os agentes foram convertidos em consumidores, enquanto que, na rede em estrela, depois de deixarmos a simulação executar durante 7500 unidades de tempo, apenas ao fim de 4448 unidades de tempo decorridas é que se conseguiu converter 26 seguidores, de 99, em consumidores.

Os resultados são os esperados, uma vez que, na rede em estrela, temos apenas um influenciador (o iniciador) e os seguidores convertidos não podem influenciar e no cenário 1 temos uma amplitude máxima de todos os parâmetros, enquanto que, no cenário 2 temos um maior empenho do iniciador e uma maior variação na resistência, uma vez que tem uma amplitude máxima, à influencia por parte do iniciador, sendo perfeitamente aceitável que, no cenário 2, mais do dobro dos agentes do cenário 1, tenha sido convertido.

Para os cenários 3 e 4 da rede em malha, não são apresentados gráficos comparativos da rede em estrela porque, depois de cerca de 100000 unidades de tempo a executar a simulação, não houve qualquer seguidor convertido em consumidor. Sublinha-se o facto de nestes 2 cenários termos a resistência dos seguidores no máximo o que apesar dos “esforços” do iniciador, mesmo quando estes estão no seu máximo, faz com que não seja fácil converter os seguidores em consumidores.

Destas simulações com tipos de redes diferentes, podemos concluir que numa rede em malha, em que os agentes se influenciam uns aos outros, é substancialmente mais rápida a conversão dos seguidores em consumidores do que numa rede em estrela em que apenas um seguidor pode influenciar os seguidores, sendo em alguns casos, até, dificílimo, se não quase impossível.

5 Conclusão

Este trabalho pretende demonstrar como podemos utilizar o computador, uma ferramenta cada vez com maior impacto e importância no dia-a-dia das pessoas e empresas, para ajudar a compreender um fenómeno social com impacto a nível económico – a divulgação de um novo produto através da comunicação direta entre pessoas. Para tal, foi desenvolvida uma simulação e foi utilizado um sistema multiagente, cada vez mais popular em estudos baseados em simulações em áreas tão diversas como física, biologia, informática, economia, etc.

A aplicação desenvolvida simula o tempo que demora a informação sobre um novo produto a passar entre potenciais clientes até que adiram ao novo produto. Esta pode ser facilmente configurada para simular situações diversas.

Entre os objetivos na criação desta aplicação, está o seu uso em sala de aulas para simulação de hipotéticas situações de mercado. Além disso, é intenção fazer com que esta aplicação seja mais do que uma ferramenta de estudo em laboratório, podendo ser uma ferramenta viável para que empresas reais possam fazer os seus estudos e retirarem daí as suas conclusões.

A utilização do *Repast Simphony* como plataforma para simulação de sistemas multiagente teve como objetivo utilizar um sistema que pudesse simular, tanto situações simples, como situações de grande complexidade, não havendo, assim, necessidade de gastar tempo na aprendizagem de diferentes plataformas. A escolha de uma plataforma, neste caso, programável em JAVA teve como finalidade a possibilidade de utilizar o mesmo código em qualquer sistema operativo.

Esta aplicação pode simular um enorme número de situações com características comuns apenas com um mínimo de alterações mas, evidentemente, que em situações radicalmente diferentes haverá necessidade de desenvolver código específico, o que leva à necessidade de um programador conhecedor da plataforma e da linguagem JAVA com alguma profundidade.

A aplicação em causa pode beneficiar de alguns melhoramentos na apresentação gráfica, apesar de não serem essenciais para a simulação de novos produtos, assim como a informação de saída para ficheiros externos poderá ser mais elaborada.

No final, temos uma aplicação, relativamente simples no seu manejo, apesar de complexa na sua conceção, que poderá ser uma ferramenta de grande utilidade para alunos e empresas.

Como trabalho futuro será criado um painel de configuração de parâmetros mais completo e um editor do ficheiro de configuração para maior simplicidade na definição de parâmetros externos, para que a aplicação possa simular situações diferentes e mais complexas com pouco esforço do utilizador. A apresentação gráfica será melhorada com o intuito de tornar a aplicação mais apelativa. Será acrescentada uma opção para configuração do tamanho do cenário. Os resultados da simulação serão registados numa base de dados e será criado um

módulo para análise estatística desses dados. Finalmente será criado um módulo para avaliação de regras de forma a expandir as possibilidades de simulação da aplicação.

A informação que existe sobre a plataforma de simulação *Repast Simphony* é muito dispersa e em língua inglesa, não havendo um manual dedicado para quem quiser aprender a trabalhar com o *Repast Simphony* e tenha poucos conhecimentos de sistemas informáticos, de programação e até da língua inglesa. Esta informação não cobre, de forma taxativa, muitos pormenores da plataforma que se vão tornando de importante relevância à medida que se desenvolve uma aplicação, por isso, na perspetiva de contribuir para a melhor compreensão desta plataforma foi apresentado, com algum detalhe e maior cuidado, o código e os procedimentos usados na criação desta aplicação.

Referências

[Alkemade and Castaldi, 2005] Alkemade F. and Castaldi C., Strategies for the Diffusion of Innovations on Social Networks, Computational Economics 25: 3–23, 2005

[Banerjee, 1992] Banerjee A.V., A simple model of herd behaviour. Quarterly Journal of Economics, 107(3), 797–818, 1992

[Camarinha-Matos and Afsarmanesh, 2009] Camarinha-Matos .LM. and Afsarmanesh H., Towards Modeling a Collaborative Environment for Extension of Professional Active Life. In: Leveraging Knowledge for Innovation in Collaborative Networks. Springer, pp. 721-732. ISBN 978-3-642-04567-7. ISSN 1868-4238. eISSN 1868-422X. URL: http://dx.doi.org/10.1007/978-3-642-04568-4_73. (ISI Web of Science), 2009

[Camarinha-Matos and Afsarmanesh, 2012] Camarinha-Matos L.M. and Afsarmanesh H., Taxonomy of Collaborative Networks Forms, Draft Working Document, In collaboration with SOCOLNET (Society of Collaborative Networks), 2012

[Carley and Hill, 2001] Carley K.M. and Hill V., Structural Change and Learning Within Organizations. In: Dynamics of Organizations: Computational Modeling and Organizational Theories, A. Lomi and E.R. Larsen, Eds., 63-92. MIT Press/AAAI Press/Live Oak, 2001

[Casti, 1998] Casti J.L., Would-Be Worlds: How Simulation Is Changing The World of Science. New York: Wiley, 1998

[Davidsson, 2000] Davidsson P., Multi Agent Based Simulation: Beyond social simulation, in Multi Agent Based Simulation (LNCS Vol. 1979), Springer Verlag, 2000

[Dawid, 1999] Dawid H., On the Convergence of Genetic Learning in a Double Auction Market, Journal of Economic Dynamics and Control, No. 23, 1999

[Durfee and Lesser, 1991] Durfee E.H., Lesser V.R., Partial global planning: a coordination framework for distributed hypothesis formation Systems, Man and Cybernetics, IEEE Transactions on Volume: 21 , Issue: 5 Digital Object Identifier: 10.1109/21.120067 , Page(s): 1167 - 1183 Cited by: Papers (26), 1991

[Durfee and Lesser, 1991] Durfee E.H. and Lesser V.R., Partial Global Planning: A Coordination Framework for Distributed Hypothesis Formation, IEEE transactions on systems, man, and cybernetics, vol. 21, no. 5, 1991

[Ferber, 1999] Ferber J., Multi-Agent System: An Introduction to Distributed Artificial Intelligence, Harlow: Addison Wesley Longman 1999

[Goldenberg, et al., 2001] Goldenberg J., Libai B., and Muller, E. Talk of the Network: A Complex Systems Look at the Underlying Process of Word-of-Mouth. Marketing Letters, 12(3), 211-223, 2001

[Heckbert, 2009] Heckbert S., Experimental economics and agent-based models, 18th World IMACS/MODSIM Congress, Cairns, Australia 13-17 July 2009, available at: http://mssanz.org.au/modsim09, 2009

[Hongyan and Tesfatsion, 2009] Hongyan Li, Tesfatsion L., The AMES wholesale power market test bed: A computational laboratory for research, teaching, and training, Power & Energy Society General Meeting, 2009. PES '09. IEEE, Digital Object Identifier: 10.1109/PES.2009.5275969, Page(s): 1 – 8, 2009

[Hussain and Wood, 2009] Hussain F. and Wood S., Modeling the Performance of Children on the Attentional Network Test, in A. Howes, D. Peebles, R. Cooper (eds.), 9th International Conference on Cognitive Modeling – ICCM09, July 2009, Manchester, UK: University of Manchester, pp. 211-216, 2009

[Jennings and Wooldridge, 1995] Jennings N.R. and Wooldridge M., Applications of Intelligent Agents, Queen Mary & Westfield College, University of London, URL: http://agents.umbc.edu/introduction/jennings98.pdf, 1995

[Jennings, et al., 1995] Jennings N. R., Corera J. M. and Laresgoiti I. Developing Industrial Multiagent Systems. In Proceedings of the First International Conference on Multiagent Systems, 423–430. Menlo Park, Calif.: AAAI Press, 1995

[Kotler and Armstrong, 2012] Kotler P. and Armstrong G., Principles of Marketing, Pearson Education, Inc., publishing as Prentice Hall, 2012

[Lee, et al., 2013] Lee K., Kim S., Kim C.O. and Park T., An Agent-Based Competitive Product Diffusion Model for the Estimation and Sensitivity Analysis of Social Network Structure and Purchase Time Distribution, Journal of Artificial Societies and Social Simulation 16 (1) 3, 2013

[Ljungberg and Lucas, 1992] Ljunberg M. and Lucas A. The OASIS air traffic management system. In Proceedings of the 2nd Pacific Rim International Conference on AI (PRICAI-92), Seoul, Korea, 1992

[Maalal and Addou, 2011] Maalal S. and Addou M., A new approach of designing Multi-Agent Systems (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 2, No. 11, 2011

[Moreno, 2010] António Moreno, Artificial Intelligence II – Multi-Agent Systems – Introduction to Multi-Agent Systems URV, Winter-Spring 2010, http://www.slideshare.net/ToniMorenoURV/introduction-to-agents-and-multiagent-systems, 2010

[North and Macal, 2007] North M. and Macal C., Managing Business Complexity: Discovering Strategic Solutions with Agent-Based Modeling and Simulation, Oxford University Press: New York, NY, 2007

[Parunak, 1987] Van Dyke Parunak, Manufacturing Experience with the Contract Net. In Distributed Artificial Intelligence, Volume 1, ed. M. Huhns, 285–310. London: Pitman, 1987

[Seah, et al., 2011] Seah C., Sierhuis M., Clancey W.J., Multi-agent Modelling and Simulation Approach for Design and Analysis of MER Mission Operations, NASA papers, 2011

[Sycara, 1998] Sycara K.P., Multiagent Systems, URL: http://www.perada.eu/documents/articles-perspectives/multi-agent-systems.pdf, 1998

[Wallace, et al., 2011] Wallace C., Jajn G. and McArthur S., Multi-agent System for Nuclear Condition Monitoring, proc of the 2nd International Workshop on Agent Technologies for Energy System (ATES'11), a workshop of the 10th International Conference of Agent and Multi-agent System (AAMAS'11), 2nd of May 2011, in Taipei, Taiwan.

[Wood, 1990] Wood S., Planning in a Rapidly Changing Environment, PhD thesis, School of Cognitive and Computing Sciences, University of Sussex, Brighton, UK, 1990

[Wooldridge, 2002] Wooldridge M., An Introduction to Multiagent Systems, John Wiley

& Sons, Ltd., 2002

[Zhang, et al., 2009] Zhang C., Hammad A., Bahnassi H., Collaborative Multi-agent Systems for Construction Equipment Based on Real-time Field Data Capturing, Journal of Information Technology in Construction, Vol. 14, 2009

Referências URL

[Amazon, ULR]	Amazon.com: Online Shopping for Electronics, Apparel, Computers, Books, DVDs & more, http://www.amazon.com [ultimo acesso: Set 2013]
[Ebay, URL]	Electronics, Cars, Fashion, Collectibles, Coupons and More Online Shopping | eBay, http://www.ebay.com [ultimo acesso: Set 2013]
[Entrepreneur, URL]	Word-of-Mouth Advertising Definition | Small Business Encyclopedia | Entrepreneur.com, http://www.entrepreneur.com/encyclopedia/word-of-mouth-advertising [ultimo acesso: Out 2013]
[Eurace, URL]	The Eurace Project, http://www.eurace.org [ultimo acesso: Set 2013]
[Forbes, URL]	Fasten Your Seatbelts: Google's Driverless Car Is Worth Trillions (Part 1) – Forbes, http://www.forbes.com/sites/chunkamui/2013/01/22/fasten-your-seatbelts-googles-driverless-car-is-worth-trillions/ [ultimo acesso: Out 2013]
[Howstuffworks, URL]	HowStuffWorks "How Amazon Works", http://money.howstuffworks.com/amazon.htm [ultimo acesso: Set 2013]
[JADE, URL]	Jade - Java Agent DEvelopment Framework , http://jade.tilab.com/ [ultimo acesso: Out 2013]
[Mathematica, URL]	Wolfram Mathematica: Technical Computing Software-Taking You from Idea to Solution, http://www.wolfram.com/mathematica/ [ultimo acesso: Fev 2013]
[Matlab, URL]	MATLAB - The Language of Technical Computing, http://www.mathworks.com/products/matlab/ [ultimo acesso: Out 2013]
[MICROECONOMIA II 1E108, URL]	Fep.up.pt, http://www.fep.up.pt/docentes/joao/material/micro2/micro2_maxlucro.pdf, [ultimo acesso: Out 2013]
[Milind Tambe, URL].	https://www.cs.drexel.edu/~greenie/cs510/bdilogic.pdf, Milind Tambe, Beliefs, Desires, Intentions (BDI), Chapter 2 of CS499 course reader [ultimo acesso: Out 2013]
[NetLogo, URL]	NetLogo itself: Wilensky, U. 1999. NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL [ultimo acesso: Ago 2013]
[NICO, URL]	NetLogo in Science, Education, and Policy | Northwestern Institute on Complex Systems (NICO), http://www.nico.northwestern.edu.html, [ultimo acesso, Out 2013]
[Pcworld, URL]	Rakuten, Japan's 'Amazon,' launches U.S. brand, the former Buy.com | PCWorld, http://www.pcworld.com/article/2026877/rakuten-japans-amazon-launches-u-s-brand-the-former-buy-com.html [ultimo acesso: Out 2013]
[Repast Simphony, URL]	Repast Suite, http://repast.sourceforge.net/ [ultimo acesso: Out 2013]
[StarLogo, URL]	StarLogo on the Web, http://education.mit.edu/starlogo [ultimo acesso: Set 2013]
[Swarm, URL]	The Swarm Simulation System: A Toolkit for Building Multi-agent Simulations, http://alumni.media.mit.edu/~nelson/research/swarm/ [ultimo acesso: Out 2013]
[Webstore Amazon, URL]	Customer Facing and Backend API functionality from Amazon Webstore, http://webstore.amazon.com/APIs-and-Extensibility/b/6368801011 [ultimo acesso: Out 2013]