Modelo de Integração de Simuladores Distribuídos Baseados em Agentes

Carlos Alberto Bragança de Oliveira

Dissertação submetida à Faculdade de Engenharia da Universidade do Porto

para obtenção do grau de Doutor em Engenharia Industrial e Gestão.

Orientador:

Professor António Ernesto da Silva Carvalho Brito

Faculdade de Engenharia da Universidade do Porto

2013

Resumo

O desempenho das empresas depende não só da qualidade das suas decisões de gestão, mas também da forma como estas são implementadas. Cada vez mais as empresas são constituídas por diferentes unidades que podem estar geograficamente distribuídas e ter diferentes níveis de autonomia. Cada unidade controla de algum modo os recursos locais e interage com outras unidades e/ou com outras empresas. Dada esta complexidade, recorre-se com frequência à simulação para analisar o comportamento de áreas críticas nas empresas, avaliando cenários alternativos antes da tomada de decisão. A dinâmica dos mercados exige constantemente a atualização dos modelos de simulação e a interação entre eles é cada vez mais uma necessidade para a obtenção de resultados credíveis. Esta realidade carece de novas abordagens que tirando partido dos modelos existentes, viabilizem a sua integração com novos modelos, facilitem a sua atualização e permitam a sua execução distribuída. Por outro lado, a utilização das capacidades das redes de computadores e o trabalho colaborativo são hoje fundamentais e devem ser considerados na pesquisa dessas novas abordagens.

A técnica de modelação com Agentes permite modelar o comportamento humano, social e organizacional bem como a tomada de decisão individual, além de que facilita o processo interativo e independente na construção de modelos. Simular processos de negócio com a técnica ABMS (Agent Based Modeling and Simulation) requer Agentes que, por um lado, contenham um alto nível de interoperabilidade e, por outro, diversas características, tais como, capacidade de interação entre topologias flexíveis, armazenamento de informação, interfaces de visualização do estado e resultados, mecanismos de comunicação e controlo do Tempo de Simulação.

Neste trabalho foi desenvolvido um modelo de integração (*Framework*) de simuladores distribuídos e baseados em agentes (ABMS) com um mecanismo de comunicação TCP/IP para a troca de mensagens. Além das mensagens trocadas, para permitir o controlo da sincronização do Tempo de Simulação dos Agentes, foi ainda criada uma linguagem para a troca de mensagens de negócio baseada no *standard* OASIS. Para a codificação de todas as mensagens foi usado o formato *standard* XML. A linguagem XML tem vindo a ser utilizada de forma generalizada por diferentes aplicações, sendo mesmo um *standard* para o armazenamento e troca de informação. A comunicação com os *Web Services* tem como base a linguagem XML. A esta linguagem recorrem igualmente os mais recentes *standards* de troca de informação eletrónica entre as empresas. Sendo assim, a utilização do XML como base na troca de

mensagens entre os componentes da *Framework* permite: simplificar a integração com *Web Services*; integrar na *Framework* simuladores desenvolvidos em diferentes plataformas; e por fim, criar uma linguagem de negócio entre os Agentes compatível com a linguagem de negócio UBL, utilizada nas aplicações de comércio eletrónico.

O mecanismo de comunicação da *Framework* foi desenvolvido a fim de beneficiar da computação distribuída. No mundo real, as interações entre empresas ou entre diferentes sectores da empresa, podem ocorrer a diferentes intervalos de tempo com uma maior ou menor amplitude. A unidade de tempo apropriada para o incremento do Tempo de Simulação a utilizar num Modelo de Simulação de uma empresa ou num sector desta, pode variar entre o segundo e o mês, podendo mesmo ser superior. A complexidade do ambiente de negócios pode ser modelada, recorrendo a elementos simples (Agentes) que interagem entre si. Uma empresa (armazém, fábrica, etc.) ou um sector da empresa é modelado como um Agente que pode ser adicionado, alterado ou removido do Modelo, a fim de analisar as diferentes configurações. O mecanismo de comunicação da *Framework* permite que cada componente (Agente) possa ser executado em diferentes computadores que estão distribuídos numa rede e cada Agente pode ter incrementos de Tempo de Simulação diferentes. Estas características da *Framework* permitem o balanceamento da simulação e encurtar o Tempo de Execução, aumentando assim o desempenho da simulação.

Abstract

The performance of a company depends on the quality of its strategic decisions but depends also on how those decisions are implemented. More and more companies are composed of different units that can be geographically distributed with different levels of autonomy. Each unit manages its local resources and interacts with other units and/or with other companies. Due to the complexity of the problem many companies use simulation to analyse the performance of critical sectors based on a number of scenarios. The dynamics of markets requires continuously updating of simulation models and the interaction between them is order to obtaining credible results. This reality requires new approaches that taking advantage of existing models enables their integration with new models, facilitate their upgrade and allow a distributed execution. Furthermore, the use of computer networks and collaborative work are now essential and should be considered in the research of these new approaches.

The Agent-Based Modelling technique allows model human behaviour, social and organizational behaviour and individual decision-making, facilitating the interactive and independent process of building models. Using the ABMS (Agent Based Modeling and Simulation) technique to simulate business processes requires agents with a high level of interoperability and, several other characteristics such as, the capacity of interaction on flexible topologies, information storage interfaces, visual interfaces of the status and outcomes, communication and Simulation Time control mechanism.

In this work we developed Framework to integrate distributed agent-based simulators (ABMS) with a mechanism of communication TCP/IP to exchange messages. In addition to the messages mechanism of the synchronization Simulation Time between Agents, a business language was created for the exchange of business messages based on the OASIS standard. The XML standard format was used for coding of all the messages. The XML has been widely used in different applications, and is really a standard for the storage and exchange of information. The XML language is the base for Web Services communications. This language is also used by the up-to-date standards for electronic information exchange between companies. Therefore, the use of XML to exchange messages between the components of the Framework allows: a nature integration with Web Services, integrating in the Framework simulators developed on different platforms, and finally create a business language between agents compatible with UBL business language that is used in electronic commerce applications.

The communication mechanism of the Framework was developed in order to benefit from distributed computing. In the real world, the interactions between companies or between different sectors of the company, may occur at different time intervals with a larger or smaller amplitude. The appropriated unit of time to timestamp used in a simulation model of a company or a sector of it, can vary between the second and the month or even greater. The complexity of the business environment can be modelled using simple elements (Agents) that interact with each other. A company (warehouse, factory, etc.) or a sector of the company modelled as an agent that can be added, changed or removed from the model in order to analyse different configurations. The communication mechanism of the Framework allows each component (Agent) to run on different computers that are distributed in a network and each Agent can have different timestamp. These features of the Framework allow a balancing simulation and reduce the execution time, in that way increasing the performance of the simulation.

.

Agradecimentos

Começo por expressar o meu reconhecimento ao Professor António Brito que aceitou orientar este trabalho fazendo-o de forma crítica e perspicaz, permitindo avanços qualitativos nos momentos mais críticos. Agradeço-lhe toda a colaboração, disponibilidade e amizade que sempre demonstrou.

Ao Professor José Sarsfield Cabral pelo entusiasmo e apoio que sempre manifestou.

Desejo também exprimir todo o meu apreço aos Professores Manuel Pina Marques e José Fernando Oliveira por todo o apoio prestado.

Ao meu amigo Carlos Corte-Real pela disponibilidade e apoio incondicional.

Finalmente, mas não de forma menos significativa, exprimo o meu reconhecimento a toda a minha família pelo estímulo e apoio incondicional que me prestaram, pela paciência e grande amizade com que sempre me ouviram e com a sensatez com que sempre me ajudaram; principalmente à minha filha Filipa pela compreensão e ternura que sempre manifestou apesar do 'débito' de atenção que teve durante estes meses, em particular, numa fase tão importante da sua vida.

Índice

Lista de Figuras

Lista de Tabelas

Glossário

ABM - *Agent-Based Modelling*

ABMS - *Agent-Based Modelling and Simulation*

AMQP - *Advanced Message Queuing Protocol*

B2B - *Business to Business*

B2C - *Business to Customer*

B2G - *Business to Governement*

BIEs - *Business Information Entities*

C2G - *Citizen to Government*

CCTS - ebXML *Core Components Technical Specification*

CD - *Configuration Data Modeller*

CGM Open - *Advancing Standards for Graphical Information Interchange*

CPPA - *Collaboration Protocol Profile and Agreement*

DAE - Agenda Digital para a Europa

DES - Simulação por Eventos Discretos

DITA - *Darwin Information Typing Architecture*

DLL - *Dynamic-link library*

DM - *Display Model*

DSR - *Design Science Research*

EA - *Environment Agent*

EAI - *Enterprise Application Integration*

ebBP - *Business Process*

ebMS - ebXML *Messaging Services*

ebXML - *Electronic Business using eXtensible Markup Language*

EDI - *Electronic Data Interchange*

EDIFACT - *Electronic Data Interchange for Administration, Commerce and Transport*

eGov - *eGovernment*

EI - *Emergency Interoperability*

EU - Comissão Europeia

HLA - *High Level Architecture*

IDtrust - *Identity and Trusted Infrastructure*

ISA - *Interoperability Solutions for European Public Administrations*

LegalXML - *Legal Data Exchange*

LSPs - *eGovernment Large Scale Pilot Projects*

OASIS - *Organization for the Advancement of Structured Information Standards*

ODF - *OpenDocument Format*

OMT - *Object Model Template*

Open CSA - *Composite Services Architecture*

RIM - ebXML *Registry Information Model*

RTI - *Runtime Infrastructure*

SA - *Simulation Agents*

SAML - *Security Assertion Markup Language*

SCA - *Service Component Architecture*

SD - Sistemas Dinâmicos

SDO - *Service Data Objects*

SI - Sistemas de Informação

SOA - *Service-Oriented Architecture*

TIC - Tecnologia da Informação e Comunicação

UBL - *Universal Business Language*

UDDI - *Universal Description Discovery & Integration*

UN - Nações Unidas

UN/CEFACT - *United Nations Centre for Trade Facilitation and Electronic Business*

UN/EDIFACT - *United Nations rules for Electronic Data Interchange for Administration, Commerce and Transport*

VV&A - Verificação, Validação e Acreditação

WS - *Web Service*

WS-BPEL - *Web Services Business Process Execution*

WS-I - *Web Services Interoperability*

WWW - *World Wide Web*

XML - *Extensible Markup Language*

Capítulo 1.

Introdução

Neste capítulo apresenta-se uma visão geral do trabalho desenvolvido no âmbito do Programa Doutoral em Engenharia Industrial e Gestão. Na primeira secção é feito o enquadramento do trabalho. Na segunda, é apresentada a motivação e os objetivos do estudo. Na terceira secção descreve-se a metodologia utilizada. Por fim, na última, é apresentada a estrutura do documento.

1.1 Enquadramento

Um processo de negócio é constituído por um conjunto de atividades e as suas interações. Nestas interações, as atividades envolvidas desempenham o papel de cliente e/ou fornecedor de informação. Dada a dependência funcional das atividades e interações, a sequência temporal da sua execução tem de ser controlada. No domínio dos negócios das empresas há uma área em que esta questão tem particular relevância que é na cadeia de abastecimento. Numa cadeia de abastecimento, constituída por vários processos de negócio, podemos encontrar um conjunto de características comuns que passamos a expor (North e Macal 2007)(Wang e outros. 2012):

- Colaborativas, integradas e distribuídas. Os vários processos de negócio da cadeia de abastecimento são distribuídos por várias localizações e desenvolvem a sua atividade de forma integrada e colaborativa;
- Dinâmicas e instáveis. Cada vez mais as empresas necessitam de uma constante adaptação das cadeias de produção, com vista à satisfação das necessidades dos clientes. Logo, a instabilidade é uma variação natural de um sistema ou de uma organização, em função das variações do ambiente que a envolvem;
- Informatizadas. Os sistemas de informação são utilizados, não só na área da produção e comercialização, mas também na gestão da qualidade. As empresas necessitam de armazenar o *feedback* dos clientes relativamente aos seus produtos, quer sejam atuais ou descontinuados

Desde os anos 60 que a Simulação por computador tem sido utilizada na análise das atividades de negócio com vista a melhorar a sua eficiência.

Numa abordagem de Simulação tradicional, o Modelo de Simulação é implementado como uma única aplicação. Por conseguinte, a modificação de um qualquer elemento do Modelo implica a alteração da aplicação. Estas alterações podem ter impacto nos restantes elementos ou na lógica do Modelo associado aos processos de negócio, sendo este impacto muitas vezes difícil de quantificar.

Como alternativa, é possível identificar os diferentes processos que podem ser independentemente modelados através de componentes, sendo as suas interações individualizadas, eliminando assim o risco de interferência inadvertida entre processos. Esta abordagem exige, no entanto, um mecanismo que garanta que os processos sejam devidamente sincronizados. Esta modelação independente pode igualmente ser generalizada aos diferentes conjuntos de atividades internas dos processos de negócio, sendo neste caso a sincronização efetuada a nível dos subprocessos.

Uma das técnicas mais utilizadas nos últimos anos que permite atingir este objetivo é a *Agent-Based Modelling and Simulation*[1] (ABMS). Uma definição simplificada de ABMS é um ambiente onde são colocados Agentes (programas de computador) que interagem entre si e com o meio ambiente, sendo a complexidade do sistema global modelado pelo conjunto dos Agentes através de uma abordagem *Bottom-up*[2] (North e Macal 2007). Neste caso, os processos de negócio também podem ser modelados utilizando Agentes, sendo o meio ambiente constituído pelo conjunto de unidades de negócio que se pretendem modelar. Como num ambiente deste tipo, as unidades de negócio apresentam uma grande autonomia, podendo pertencer à mesma empresa ou a empresas distintas e muitas vezes ter diferentes localizações geográficas, o recurso a mecanismos de Simulação Paralela e distribuída parece fazer todo o sentido.

Assim, a criação de Modelos de Simulação recorrendo à Simulação Paralela é uma opção apropriada na simulação de processos de negócios tendo em conta: a independência funcional

[1] Modelação e Simulação Baseada em Agentes

[2] De baixo para cima

interna das diferentes unidades do processo de negócio; a dificuldade de gerar um único Modelo de Simulação contendo várias unidades de negócio; a semelhança funcional e a complexidade específica de algumas das unidades; a evolução das capacidades de processamento e comunicação dos computadores atuais.

Na Simulação Paralela ou Distribuída, a abordagem por acontecimentos discretos é uma das técnicas mais utilizadas. Neste caso, o controlo do Tempo de Simulação pode ser síncrono ou assíncrono. O primeiro é caracterizado pela existência de um único relógio e por uma lista global de acontecimentos. A existência de um único relógio global na Simulação Paralela ou distribuída é apropriada para simuladores em tempo real, uma vez que que todos os simuladores se encontram na mesma unidade de tempo. Em contrapartida, no controlo do tempo assíncrono, cada simulador tem o seu próprio relógio e lista de acontecimentos, sendo a sincronização feita de duas formas: conservadora ou otimista. No mecanismo de sincronização conservadora, o processo de sincronização tem que garantir que as mensagens recebidas por um simulador são para uma unidade de tempo superior à unidade de tempo do relógio local. O mecanismo otimista é de implementação mais elaborada, pois o simulador tem de ter um mecanismo de *roll back*[3] na eventualidade de ser necessário recuar no tempo. Neste caso, se uma mensagem é recebida para uma unidade de tempo inferior ao tempo corrente do relógio local, o simulador tem de repor o estado do sistema na unidade de tempo da mensagem e reiniciar a simulação a partir desse ponto (Nicol e Liu 2002), (Niewiadomska-Szynkiewicz, Zmuda, e Malinowski 2003).

Os recursos computacionais e a tecnologia utilizada para controlar o fluxo de tempo durante a simulação, são os dois principais fatores que podem influenciar o tempo necessário para a completar. O recurso a programação paralela ou distribuída pode diminuir o Tempo de Execução, uma vez que a mesma quantidade de Tempo de Simulação pode ser concluída numa menor quantidade de Tempo Real (Niewiadomska-Szynkiewicz, Zmuda, e Malinowski 2003).

[3] Retroceder

1.2 Motivação e Objetivos

A motivação principal para a realização deste trabalho foi a criação de um modelo de integração (*framework*[4]) de simuladores baseados em agentes, adaptado às particularidades dos processos de negócio de uma cadeia de abastecimentos. Uma motivação adicional foi criar uma *framework* com a possibilidade de integrar simuladores independentes já existentes ou que possam vir a ser desenvolvidos no departamento, desenvolvidos por outros investigadores.

A globalização e internacionalização das empresas, tornou imperioso a criação de *standards* de comunicação não só ao nível tecnológico mas também ao nível da sintaxe e da semântica das mensagens de negócio trocadas entre as empresas. A utilização destes *standards* como troca de informação dos simuladores que integram a *framework* foi também uma motivação adicional, que poderá permitir, em última análise, a interação entre os simuladores e empresas reais utilizando uma linguagem comum.

A finalidade principal deste trabalho foi desenvolver uma *framework* adaptada aos processos de negócio, que desse resposta às seguintes questões de investigação

- **É a Simulação Baseada em Agentes o método adequado para simular problemas complexos de negócios?**

Uma análise comparativa de várias atividades de negócio permite encontrar características e objetivos comuns entre elas, tais como: existência de recursos humanos e mecanizados; envolvimento de várias organizações no processo de negócio; interesses que podem ser concorrentes e interligados; distribuição espacial das organizações; distribuição de responsabilidades sobre as tarefas envolvidas; sistemas de informação similares ou incompatíveis; determinação em melhorar a eficiência do negócio.

As técnicas de simulação utilizadas têm evoluído paralelamente à evolução das capacidades computacionais. Nos últimos anos, têm surgido várias técnicas de Simulação, apelando à Simulação Paralela ou Distribuída, em particular técnicas que recorrem ao conceito de Agente (2.1.2.1) como a *Agent-Based Modelling and Simulation* (ABMS).

[4] No presente trabalho, o termo *framework* é utilizado com o significado de "modelo de integração de simuladores"

A comparação das características de um processo de negócio e as dos Agentes explica a crescente utilização da técnica ABMS na Simulação de processos de negócio. As particularidades mais relevantes da técnica ABMS, é a sua capacidade de fornecer uma descrição natural do sistema e capturar os seus fenómenos emergentes a simular. Todos os sistemas ABMS usam técnicas testadas e consolidadas, tais como, Simulação por Eventos Discretos e programação orientada a objetos.

- **Como controlar o Tempo em Simulação Distribuída?**

Num processo de negócio real que envolve várias empresas, cada uma trabalha em paralelo e de forma independente, havendo em alguns intervalos de tempo algumas interações entre elas. Este comportamento pode também acontecer de forma idêntica no interior das empresas, onde os diferentes sectores interagem uns com os outros e com o exterior da empresa, em diferentes intervalos de tempo. O intervalo de tempo entre as interações das empresas ou dos sectores pode variar de segundos a semanas ou mesmo meses. Para modelar um processo de negócio nestas condições, com um simulador tradicional, a unidade de tempo utilizado tem de ser a menor presente no sistema.

A utilização do ABMS pode aumentar o desempenho deste tipo de simulação, uma vez que as diferentes empresas ou seus sectores podem ser modelados com o seu próprio e distinto intervalo de tempo, permitindo uma melhor adaptação a cada subsistema. Além disso, estes Modelos podem ser distribuídos por vários computadores. Logo, o uso da técnica ABMS permite modelar e testar partes do sistema que, quando integradas, permitem simular o sistema completo.

As duas principais técnicas de controlo e avanço do Tempo de Simulação utilizadas pelas ferramentas ABMS, é a *Time-step*[5] e *Next-event*[6]. A técnica *Time-step* é mais fácil de implementar, contudo geralmente o Tempo de Execução da Simulação é mais elevado que no caso da técnica *Next-event* que, embora requeira uma programação mais elaborada, permite uma representação mais realista do sistema.

Sendo um dos objetivos deste trabalho desenvolver uma *framework* de simulação adaptada aos vários processos de negócio da cadeia de abastecimento, e atendendo às características

[5] Incremento no tempo

[6] Próximo acontecimento

específicas já mencionadas, a técnica *Next-event* foi a escolhida por ser a que permite que cada Agente simulador possa correr com incrementos de Tempo de Simulação distintos.

Um dos principais objetivos desde o início foi a introdução de mecanismos que tratem as particularidades associadas aos processos de negócio, em particular, a possibilidade de os simuladores poderem avançar independentemente o seu Tempo de Simulação, de acordo com o seu ritmo individual, com vista a conseguir-se tempos de execução da simulação mais reduzidos.

- **Criar uma nova *framework* ou utilizar ferramentas existentes?**

Atualmente, várias ferramentas de simulação baseadas na tecnologia ABMS podem ser encontradas, sendo mesmo algumas delas do domínio público. Por outro lado, existem vários trabalhos de investigação que desenvolvem simuladores ABMS proprietários. Ambas as opções têm vantagens e desvantagens. Os *softwares* genéricos são usados principalmente para desenvolver Modelos síncronos orientados a objetos, porém são difíceis de se ajustar a problemas específicos. A principal vantagem dos programas proprietários é a possibilidade de, por um lado, serem personalizados pelo utilizador, e, por outro lado, incorporarem outros módulos de programação desenvolvidos pelo utilizador. O desenvolvimento de uma *framework* proprietária tem ainda a vantagem de ser possível adaptar e utilizar simuladores existentes, desenvolvidos por outros utilizadores.

Sendo uma das motivações iniciais a possibilidade de incorporar simuladores existentes na *framework* a desenvolver, a opção mais vantajosa pareceu ser o desenvolvimento de uma *framework* proprietária.

A utilização de linguagens de programação orientada a objetos permite implementar de uma forma natural o comportamento dos Agentes. Depois de avaliar várias linguagens de programação como o JAVA e o C#, foi escolhido o C# que com o *framework* .NET permite criar aplicações com as funcionalidades do C++ e do JAVA.

- **Qual a linguagem de comunicação a utilizar?**

Os standards de comunicação de documentos entre empresas e organizações governamentais mais utilizados, são baseados na linguagem de marcação XML. As comunicações com *Web*

Services têm como base a linguagem XML. A comunicação entre Agentes em algumas das ferramentas de domínio público pode ser feita com base no XML.

A troca de informação entre os Agentes da *framework* é feita pelo envio de mensagens XML, e a integração de simuladores existentes é conseguida através do envio de mensagens XML do *output* de um simulador para o *input* de outro.

Relativamente às mensagens de negócio trocadas entre os Agentes, estas respeitam a linguagem estandardizada OASIS *Universal Business Language* (UBL), de modo a permitir que outros Agentes possam ser desenvolvidos em linguagens de programação diferentes do C# - desde que permitam a criação de ficheiros XML de acordo com a especificação OASIS.

1.3 Abordagem Metodológica

O cariz multidisciplinar dos Sistemas de Informação (SI) requer métodos de investigação próprios. Isto porque, a investigação neste domínio debruça-se em diferentes áreas do saber, desde as ciências de computação, engenharia de *software*, ciências da organização, gestão, economia, ética, sociologia, psicologia, estatística, medicina, semiótica, pensamento sistémico, entre outras (Ferreira e outros. 2012).

A *Design Science Research* (DSR) (A. Hevner e outros. 2004) (Ken Peffers e outros. 2007) é uma metodologia que tem vindo a ser desenvolvida e que parece adequada a projetos desta natureza. Esta pretende unificar e ajustar as diferentes abordagens utilizadas, garantindo disciplina, rigor e transparência na condução deste tipo de projetos de investigação.

Sendo o presente trabalho um projeto de natureza tecnológica, surge a necessidade de recorrer a uma abordagem de investigação metodológica que o valide segundo os critérios de investigação científica. A escolha da DSR deve-se essencialmente aos seguintes aspetos:

- Facilidade de aplicação aos SI;
- Metodologia adequa-se ao objetivo do trabalho.

Os SI tratam-se de uma atividade de *Design* em que os resultados são artefactos cujo objetivo é a ação de melhoria de condições do mundo real, nomeadamente, profissional (K Peffers, Tuunanen, and Gengler 2006). Ou seja, geralmente são implementados com o propósito de melhorar a eficácia de organizações. Sendo este um dos objetivos do presente trabalho, recorre-se à DSR para garantir que os resultados obtidos sejam científicos, na medida em que esta

metodologia visa questões relacionadas com o rigor e o corpo de conhecimento para o qual o projeto de investigação contribui.

Além disso, na terminologia desta metodologia pretende-se ultrapassar as limitações das capacidades tanto humanas como organizacionais, criando novos artefactos ou desenvolvendo artefactos já existentes (A. Hevner e outros. 2004). Sendo assim, o objetivo da metodologia DSR confunde-se com o objetivo deste trabalho, visto que este procura criar um novo artefacto cuja instanciação (protótipo) pretende resolver problemas organizacionais existentes.

1.3.1 *Design Science Research*

Segundo A. Hevner e outros. 2004, existem dois paradigmas que caracterizam uma investigação de SI: a *behavioral science* e a *design science*. O autor defende que, apesar de distintas, estas duas ciências complementam-se e devem entrecruzar-se várias vezes ao longo de um trabalho de investigação desta natureza.

A *behavioral science* procura desenvolver e verificar teorias que explicam ou preveem o comportamento humano ou organizacional. Para isso, busca analisar, perceber e desenvolver teoricamente o uso, implementação, manutenção e *design* dos SI.

A *design science* apresenta-se como um corpo de conhecimento que visa criar saber sobre o processo de *design*, que consiste numa sequência de atividades especializadas que acabam por produzir um produto inovador, isto é, um artefacto de *design* (Ferreira et al. 2012). Além disso, na *design science*, tanto o conhecimento, como a compreensão e resolução de problemas são alcançados através da construção e aplicação de artefactos que podem ser novos ou versões melhoradas de constructos, modelos, métodos ou instanciações (A. Hevner e outros.2004).

Por sua vez, a *design science research* (DSR) refere-se ao processo de investigação para a criação do artefacto, tendo como objetivo garantir disciplina, rigor e transparência à investigação, para que o conhecimento obtido seja não só tecnológico mas também científico.

1.3.1.1 Diretrizes da Metodologia

De acordo com A. Hevner e outros.2004, a metodologia adotada é composta por sete diretrizes (Tabela 1)

Tabela 1 – Diretrizes da *design science research*

Diretriz	Descrição
Diretriz 1: *Design* de um artefacto	*design science research* deve produzir um artefacto na forma de um constructo, modelo, método ou instanciação
Diretriz 2: Problema relevante	O objetivo *design science research* é desenvolver soluções tecnológicas para importantes e relevantes problemas de negócios
Diretriz 3: Avaliação do *Design*	A utilidade, qualidade e eficácia do *Design* de um artefato deve ser rigorosamente demonstrado através corretos métodos da avaliação
Diretriz 4: Contribuições da investigação	Uma *design science research* deve fornecer contribuições claras nas áreas de investigação do artefacto quer no *Design* quer na metodologias
Diretriz 5: Rigor da investigação	A *design science research* baseia-se na aplicação de métodos rigorosos na construção e avaliação do artefacto
Diretriz 6: *Design* como processo de pesquisa	A criação por um artefacto eficaz requer a utilização dos meios disponíveis para alcançar os fins desejados, satisfazendo leis e o ambiente que o envolve
Diretriz 7: Divulgação	A *design science research* ser apresentada de forma eficaz, tanto a um público composto por técnicos como por gestores

A DSR requer a criação e proposta de um artefacto inovador (diretriz 1) que aborda e procura resolver um problema organizacional (diretriz 2). Significa que, para conseguir uma pertinente e correta implementação e aplicação do artefacto, identifica-se o problema específico, que se trata da questão da investigação. Sendo assim, simultaneamente define-se os objetivos pretendidos para o artefacto e os requisitos a implementar.

Uma avaliação da qualidade do artefacto produzido (diretriz 3) também é necessária. Ou seja, avalia-se a eficácia com que o artefacto resolve o problema a que se propôs. Em causa estão, não só o processo de conceção, implementação e arquitetura do artefacto, mas também a sua funcionalidade, qualidade, utilidade, precisão e desempenho.

Para se verificar o carácter inovador do artefacto (diretriz 4), avalia-se a forma como resolve um problema nunca antes solucionado, ou então, a maneira como consegue resolver mais eficazmente um problema existente. No fundo, procura-se perceber quais são as novas contribuições que enriquecerão as áreas de conhecimento, construção e avaliação do *design* de artefactos.

A quinta diretriz (5) está relacionada com o rigor da investigação, incentivando à precisão, coerência, consistência e formalidade na construção, avaliação e apresentação do artefacto. Isto é, procura-se executar rigorosamente as diretrizes anteriormente expostas, para que sejam consideradas válidas dentro do saber científico.

Encarar o *design* como processo de pesquisa para encontrar uma solução para um problema, insere-se na sexta diretriz (6). Inerente a este processo, está a realização de uma atividade cíclica que vai alternando entre a conceção e o posterior testar das soluções encontradas - soluções estas que são influenciadas tanto pelos meios disponibilizados, como pelas leis e o ambiente que as envolvem.

Por fim, os resultados da DSR são divulgados a um público especializado que tanto pode ser técnico como administrativo (diretriz 7). Este público pode consistir em, por um lado, investigadores que proliferam os resultados obtidos ou então testam e estudam-nos inserindo-os no contexto para o qual foram propostos. Por outro lado, podem ser profissionais que implementarão os artefactos nas suas organizações - ou pelo menos analisarão a hipótese e os benefícios que a sua aplicação poderá trazer.

1.3.1.2 Instanciação da Metodologia

Como já foi referido anteriormente, na procura de rigor científico, o presente trabalho adapta o *desing science research* como processo de investigação, logo durante a sua elaboração tem-se em consideração as suas diferentes diretrizes.

Um artefacto é criado e proposto (diretriz 1) com o objetivo de contribuir para uma melhoria do desempenho organizacional de uma empresa. Para isso, efetua-se a caracterização do problema (diretriz 2), demonstra-se a sua relevância e esclarece-se a necessidade de encontrar soluções. Simultaneamente, opta-se por avaliar alguns modelos organizacionais existentes, bem como a sua representação em termos de informação, de forma a detetar que intervenções de melhoria motivadas podem ser implementadas. Assim, consegue-se igualmente definir os requisitos que o protótipo deve satisfazer.

A avaliação do artefacto (diretriz 3) inicia-se logo após a sua conceção e implementação. Esta avaliação permite obter um *feedback*, bem como uma melhor compreensão do problema que gerou a sua criação. Reunida esta informação, consegue-se ir melhorando o artefacto, bem

como aperfeiçoando o processo de *Design* utilizado. Sendo assim, acaba por se gerar um ciclo entre a construção e a avaliação do protótipo que é repetido várias vezes até chegar à sua versão final (diretriz 6).

O artefacto criado propõe-se a resolver mais eficazmente um problema existente, propondo uma nova e diferente abordagem do *design* de artefactos de simulação (diretriz 4). Procura-se, desta forma, uma melhor performance com a aplicação desta nova ferramenta.

O processo de *design* do protótipo tem em consideração *standards* internacionais, no sentido de conferir rigor científico ao trabalho. Além disso, ao longo da investigação procurou-se ter uma atitude analítica, experimental e de constante verificação (*teste*) dos resultados obtidos, apresentando-os formal, precisa e coerentemente (diretriz 5).

Em virtude do tempo disponibilizado para a concretização deste trabalho, não foi possível comunicar os resultados a um público especializado (diretriz 7), ficando esta tarefa para trabalho futuro.

Convém referir ainda que à metodologia DSR juntou-se o recurso a outras técnicas de pesquisa, nomeadamente à pesquisa bibliográfica e à utilização de técnicas de engenharia de *software*. Além disso, o protótipo desenvolvido tem em conta o modo de funcionamento e comportamento das entidades que pretende modelar (*behavioral science*), características estas que influenciam e com os quais se vai confrontando todo o processo de *design*.

1.4 Estrutura da Tese

Este documento está dividido em sete capítulos organizados da seguinte forma:

- Neste Capítulo, é apresentado o trabalho e seus principais objetivos.
- Os próximos dois capítulos apresentam a base necessária para entender as áreas científicas e tecnológicas relacionadas com o presente trabalho. No Capítulo 2 é apresentado o estado da arte relacionada com simulação tradicional e comparação com a simulação baseada em Agentes. O Capítulo 3 apresenta o levantamento efetuado sobre o estado da arte relacionada com o comércio eletrónico em geral.
- No Capítulo 4 é apresentada a arquitetura da *Framework*, depois de analisar o seu domínio de aplicação e as particularidades mais relevantes que se propõe resolver.

- No Capítulo 5 pode ser encontrada uma descrição das bibliotecas mais relevantes que foram desenvolvidas para integrar a *Framework*.
- No Capítulo 6 são apresentados e analisados os resultados resultantes de alguns dos testes efetuados com a *Framework* desenvolvida.
- Algumas conclusões e possíveis trabalhos futuros são apresentados no Capítulo 7.

Capítulo 2.

Estado da Arte

Neste capítulo é apresentado o levantamento bibliográfico das diferentes técnicas de simulação que estão de algum modo relacionadas com o presente trabalho. Foram feitas pesquisas sobre as diferentes técnicas de simulação utilizadas tanto no passado como mais recentemente, em particular relativas à Simulação Baseada em Agentes. Na primeira secção são apresentados alguns conceitos relacionados com programas de simulação. Na segunda secção é feita uma comparação entre a programação paralela e a programação distribuída. Em seguida, na terceira secção, é feita uma breve descrição do conceito de Agente. A abordagem de simulação ABMS é apresentada na quarta secção. Posteriormente, na quinta secção, analisa-se as diferentes abordagens de sincronização entre os Agentes. Por fim, são feitas algumas considerações gerais sobre a evolução dos programas de simulação.

2.1 Modelos e Simuladores

Um Modelo é uma entidade utilizada para representar uma outra entidade, real ou virtual, com uma finalidade definida (White e Ingalls 2009). Sendo normalmente o Modelo uma representação simplificada de uma realidade, o seu nível de detalhe depende do objetivo do estudo.

Uma definição simplificada de simulação é: fazer experiências com um Modelo que representa um sistema, com o objetivo de obter conclusões indiretas sobre o comportamento do sistema. No presente trabalho, o termo simulação significa simulação por computador, no qual o Modelo é uma entidade computacional que simula o comportamento do sistema, num intervalo de tempo definido.

Muitas empresas utilizam *software* de simulação para analisar as atividades do negócio, a fim de melhorar a sua eficiência. Cada empresa tem as suas próprias características, porém algumas características comuns podem ser identificadas (N.R. Jennings, Norman, e Faratin 1998):

- Existência de várias organizações ou entidades que estão envolvidas no processo de negócio, com um objetivo comum de maximizar os seus próprios lucros.
- As organizações ou entidades envolvidas no processo de negócio podem estar fisicamente distribuídos por todo o mundo (desde a mesma cidade até continentes diferentes). Além de que é possível ainda existirem organizações virtuais sem local físico definido.
- Dentro das organizações, os recursos envolvidos no processo de negócio, também podem estar repartidos de uma forma distribuída.
- Cada organização tem o seu sistema de informação e estes podem ou não ser compatíveis entre si.
- Organizações com tarefas interligadas executadas de forma sequencial ou em simultâneo.
- O processo de negócio é um sistema dinâmico e imprevisível. Qualquer plano detalhado no tempo é muitas vezes interrompido e alterado por acontecimentos imprevistos.

2.1.1 Modelo

Um Modelo é como uma representação simplificada de um sistema. Um Modelo pode ser usado para estudar sistemas reais ou sistemas conceituais, tais como, um processo industrial ou comercial, uma cadeia de abastecimento, etc. Dependendo do sistema e dos objetivos em estudo, o sistema pode ser modelado como Modelos reduzidos (Modelos físicos à escala) ou como Modelos conceptuais (representação não física mas por um conjunto de fórmulas matemáticas) (Figura 1). Estes Modelos conceptuais podem ser implementados como contínuos, discretos ou mesmo como parcialmente discretos/contínuos. Num Modelo contínuo, os estados do sistema são representados como um conjunto contínuo de valores. Nos Modelos não contínuos, quando são utilizados métodos estatísticos, o Modelo diz-se Estatístico. Quando o Modelo representa apenas estados discretos do sistema, diz-se Modelo Discreto. A escolha do método mais adequado, depende do sistema em estudo (J. M. Teixeira 2006).

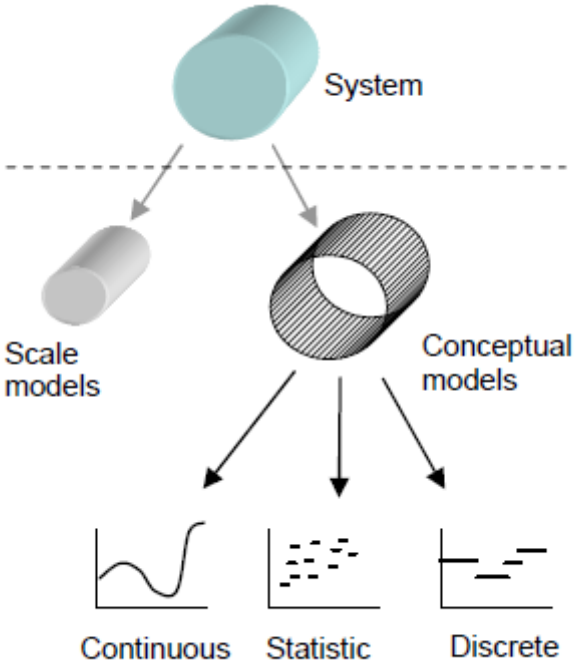


Figura 1 - Tipos de Modelos (J. M. Teixeira 2006)

Ao longo do tempo, o Modelo representa ou reproduz alguns aspetos relevantes do comportamento do sistema, mas em geral não capta toda a sua realidade e complexidade. Em algumas situações, é impossível criar um Modelo matemático que represente exatamente o sistema, descartando todas as características que não são importantes para o estudo proposto (Pereira 2010).

2.1.2 Simuladores

Simulação é um termo genérico utilizado para descrever diferentes tipos de atividades (Brito 1992). Shannon (Shannon 1998) define simulação como:

> *"Simulation is the process of designing a model of a real system and conducting experiments with this model for the purpose of understanding the behavior of the system and /or evaluating various strategies for the operation of the system".*

O referido autor considera ainda fundamental que o Modelo seja concebido de tal forma que o seu comportamento possa reproduzir a resposta do sistema real de acontecimentos que ocorrem ao longo do tempo.

Note-se que Simulação por Computador não é uma técnica recente, tendo vindo a ser utilizada há alguns anos. A primeira referência indica os anos 60 como o início da Simulação por

Computador (Shannon 1998), (Pidd e Carvalho 2006), onde se podem distinguir dois grandes grupos (Tabela 2): Ambientes Virtuais e Simulações Analíticas (R. M. Fujimoto 2000).

Nos simuladores do grupo Ambientes Virtuais há uma forte interação homem-máquina em tempo real. Jogos de vídeo, simuladores de voo, simuladores de condução e simuladores de guerra, são alguns dos mais conhecidos simuladores deste tipo. Neste caso, o simulador modela o comportamento de um ambiente real ou hipotético e a interação com o ser humano é feita em tempo real. Por isso são normalmente utilizados como ferramentas de aprendizagem ou de treino.

Tabela 2 - Grupos de Simuladores

	Ambientes Virtuais	**Simulações Analíticas**
Interação homem-máquina	Forte interação	Limitada
Tempo de Simulação	Igual ao tempo real	Várias vezes superior ao tempo real
Incremento do tempo	Contínuo e igual ao tempo real	Por incrementos constantes ou varáveis
Domínio de aplicação	Jogos, Simuladores de aprendizagem, substituição de componentes	Análise de problemas complexos
Modelo	Contínuo	Discreto

Nas Simulações Analíticas o Modelo pretende reproduzir, o mais rigorosamente possível, comportamentos de um sistema real durante um dado intervalo de tempo real, várias vezes superior ao tempo necessário para a simulação. A Simulação Analítica tem normalmente três fazes: criação do Modelo; execução da simulação; análise dos resultados. Geralmente a intervenção humana está limitada ao processo de criação do Modelo e análise dos resultados. Nalguns casos, quando o simulador tem uma interface gráfica para visualizar o resultado da simulação, ao longo do tempo durante o processo de simulação pode haver alguma interação humana no controlo do avanço do tempo e alteração de alguns dos parâmetros de simulação.

Os domínios de aplicação e objetivos dos dois simuladores são diferentes. Desta forma, os Ambientes Virtuais estão fora do âmbito deste trabalho. Salvo indicação em contrário, neste

trabalho termo simulação refere-se a simulação por computador do grupo das Simulações Analíticas.

A simulação analítica tem sido utilizada na análise de problemas complexos e é uma ferramenta importante para auxiliar a tomada de decisão. Pode ser usada para testar, comparar e validar cenários alternativos de um sistema real ou de um sistema ainda em fase de projeto. Pode igualmente descrever e/ou prever as características de um sistema em diferentes circunstâncias.

De acordo com o tipo de Modelo utilizado na simulação, os simuladores podem ser classificados como contínuos ou discretos. Nos simuladores contínuos, as mudanças de estado do Modelo ocorrem de uma forma continua ao longo do Tempo de Simulação. Caso as mudanças de estado do Modelo ocorram de forma descontínua, o simulador diz-se discreto, sendo normalmente a escala do Tempo de Simulação várias vezes superior à escala do tempo real.

A simulação de um sistema real ou em fase de projeto envolve as seguintes etapas:

- Criação de um Modelo que reproduza os comportamentos do sistema;
- Efetuar vários ensaios com o Modelo para analisar o seu comportamento;
- Tentar entender, resumir e / ou generalizar esses comportamentos;
- Testar e comparar soluções alternativas do sistema.

Na maioria das metodologias de simulação utilizadas (estocástica, contínua, de Eventos Discretos, orientada para o processo, modelação baseada em Agentes) o Modelo analisa o fluxo das entidades no sistema. Estas podem ser qualquer objeto do sistema, como clientes, pedidos e informações que fluem internamente ou podem igualmente entrar e sair do sistema. Além do mais, as entidades são objetos a quem estão associados atributos que podem variar de acordo com a entidade. Os atributos podem ser estáticos, como o nome, ou dinâmicos, como o tempo de entrada no sistema ou tempo de permanência no sistema.

As duas técnicas mais utilizadas no avanço do Tempo de Simulação em simuladores discretos, são a *Time-step* e *Next-event* .

2.1.2.1 *Time-step*

Nos simuladores por *Time-step*, o avanço do Tempo de Simulação é obtido pelo incremento do tempo em intervalos predefinidos. Neste caso o estado do simulador é atualizado a cada incremento do Tempo de Simulação. Normalmente, estes incrementos são constantes desde o início até ao fim da simulação. Note-se que para os simuladores ABMS que utilizam a técnica

Time-step, o incremento do Tempo de Simulação tem de ser o mesmo para todos os Agentes envolvidos. Neste caso, o incremento de tempo utilizado na simulação deve ser o menor para todos os Agentes envolvidos no simulador, independentemente da periodicidade dos seus acontecimentos. Ou seja, se a periodicidade dos acontecimentos de um Agente for T e a de outro Agente for 2T, o incremento de tempo a utilizar na simulação ABMS *Time-step,* deve ser T, de modo a permitir que as mensagens trocadas entre os Agentes possam ocorrer na mesma unidade de tempo. Além disso, na técnica *Time-step*, o incremento de tempo é representado por um número inteiro, que pode representar o período de um segundo ou de um ano. Por outro lado, a unidade de tempo deve ser a mesma em todos os Agentes, sendo que alguns autores recomendam que se utilize a unidade de tempo inferior, para permitir uma maior flexibilidade isto é se, por exemplo, os acontecimentos ocorrem diariamente, a hora deveria ser escolhida como unidade de tempo e não o dia.

Vários acontecimentos podem ocorrer na mesma unidade de tempo em diferentes Agentes. No caso de um Agente ter agendado mais do que um acontecimento para a mesma unidade de tempo, a ordem pela qual são executados, dependendo do algoritmo implementado, pode ser: por ordem de chegada, por ordem de chegada com rotação ou de forma aleatória. Relativamente à execução por ordem de chegada, esta pode causar alguma distorção da realidade, visto que os Agentes mais simples serão os primeiros a marcar acontecimentos que serão sistematicamente os primeiros a ser executados. Assim, a introdução de um mecanismo de rotação na escolha dos acontecimentos a executar elimina, em parte, a tendência de os primeiros Agentes a gerarem acontecimentos serem sempre os primeiros Agentes cujos acontecimentos são executados. Por fim, a escolha do acontecimento de forma aleatória, dentro dos acontecimentos simultâneos, apesar de ter a vantagem de eliminar os problemas anteriormente descritos, tem a desvantagem de poder conduzir a soluções distintas em cada simulação.

2.1.2.2 *Next-event*

Na técnica *Next-event* o Tempo de Simulação avança de forma descontínua em função do Tempo de Execução dos acontecimentos. O estado do simulador é atualizado somente quando um acontecimento ocorre. Ademais, o agendamento dos acontecimentos pode ocorrer em qualquer instante de tempo não necessariamente múltiplo do incremento de tempo anterior. Normalmente esta técnica é mais eficiente pois, por um lado, adapta-se à sucessão natural dos

acontecimentos, e, por outro lado, os tempos inativos do sistema podem ser ultrapassados, sendo tratados somente os momentos relevantes. Estes incrementos do Tempo de Simulação podem ser diferentes de Agente para Agente, de modo a que seja o mais adequado para cada Agente. Todavia, caso o incremento do Tempo de Simulação seja diferente de Agente para Agente, é necessário garantir que a troca de mensagens entre os Agentes é feita na mesma unidade de tempo, isto é, certificar que a sequência de acontecimentos não é afetada pela não sincronização temporal dos Agentes.

2.2 Simulação Paralela e Distribuída

Na Simulação Paralela, a execução de programas de simulação é efetuada em computadores com capacidade de multiprocessamento, onde o processamento é distribuído pelos diferentes processadores. Na Simulação Distribuída, a execução dos simuladores é distribuída por uma rede de computadores. Em ambos os casos o objetivo é diminuir o Tempo de Execução do Modelo. No caso da Simulação Distribuída é possível ainda criar um ambiente geograficamente distribuído.

Ao longo dos anos, a Simulação Distribuída tem sido utilizada em diferentes domínios, em especial no militar. Isto é, em simulações em tempo real, onde o simulador é utilizado para substituir um elemento do sistema (um avião inimigo, uma maquina, etc.), sendo que a Simulação Distribuída permite a utilização de recursos geograficamente distribuídos. Contudo, a sua complexidade limita a sua utilização em outras áreas onde se pretende simular o comportamento futuro de um ou vários participantes na simulação. Quando se pretende simular o comportamento futuro de um elemento do sistema (por exemplo, um fornecedor ou cliente na cadeia de abastecimento), a sincronização temporal na simulação distribuída necessita de um mecanismo de sincronização adequado.

Além disso, numa Simulação Distribuída, os simuladores são normalmente implementados em computadores em rede que poderão ter sistemas operativos e linguagens de programação diferentes e a troca de informação entre os diferentes simuladores é feita pelo envio de mensagens. Note-se que a existência de diferentes sistemas operativos não significa qualquer limitação, permitindo aliás que os diferentes módulos sejam desenvolvidos no ambiente mais adequado a cada situação.

As seguintes vantagens justificam a crescente utilização dos sistemas distribuídos (R. M. Fujimoto 2000):

- **Redução do Tempo de Execução**. A divisão do Modelo de Simulação em Submodelos, distribuídos por vários computadores, permite a execução da simulação em paralelo, logo teoricamente pode reduzir o Tempo de Execução até um fator igual ao número de processadores utilizados;
- **Distribuição geográfica**. A simulação com recurso a um conjunto de computadores distribuídos geograficamente permite criar mundos virtuais com participantes, com restrições de movimentação, que estão fisicamente localizados em diferentes locais;
- **Integração de simuladores existentes**. Quando existem simuladores que se complementam e estão desenvolvidos em sistemas operativos diferentes, pode ser mais rentável criar uma Simulação Distribuída do que transferir e adaptar esses simuladores para um único computador.
- **Tolerância a falhas**. Se um processador falha, pode ser possível, em algumas situações, continuar a simulação com os restantes.

Grande parte do trabalho envolvido no desenvolvimento de simuladores paralelos ou distribuídos está relacionado com as técnicas de sincronização (R. M. Fujimoto 2000). O algoritmo de sincronização deve permitir obter o mesmo resultado tanto em simuladores distribuídos por uma rede de computadores como em simuladores num só computador.

2.3 Agente

O conceito de Agente não é consensual, sendo tema de muitos debates na comunidade científica. Uma definição que muitos autores tendem a concordar é a seguinte (Nicholas R. Jennings 2000):

> *"An agent is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives".*

As propriedades e os atributos de um *software* ou Modelo, a fim de ser considerada um Agente ou *software* Baseado em Agente, variam de autor para autor. Alguns classificam de Agente, uma entidade computacional que contem componentes independentes. Outros afirmam que, para ser considerado Agente, a entidade computacional deve ter autonomia, capacidade de raciocínio e aprendizagem. Em todo o caso, tipicamente num *software* baseado em Agentes, estes (Figura 2) têm associados atributos, métodos internos e métodos que permitem a interação com outros Agentes e com o ambiente.

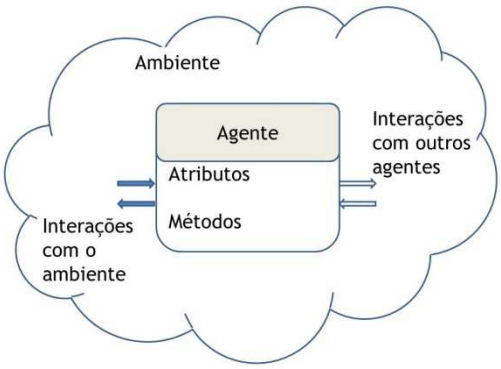


Figura 2 - Agente

Também os atributos e propriedades que se associam aos Agentes variam de acordo com os autores, porém os mais utilizados são os seguintes (C M Macal e North 2010) (Macedo 2001):

- **Identificável**

  Um Agente é um indivíduo independente, modular e exclusivamente identificável. O requisito de modularidade do *software* tipo Agente implica que um Agente tenha limites bem definidos. Aliás, deve ser possível determinar se algo é parte de um Agente ou se é um atributo comum. Ademais, o Agente tem atributos que permitem distingui-lo de outros Agentes e simultaneamente possibilitam a sua identificação por parte dos outros com quem partilha o ambiente;

- **Autónomo**

  Um Agente é autónomo e auto direcionado. Funciona de forma independente no seu ambiente e nas relações com outros Agentes. Possui controlo sobre as suas próprias ações e sobre o seu estado interno;

- **Sociável**

  Um Agente interage tanto com outros Agentes como com os humanos, tendo como objetivo de finalizar a resolução de um problema ou auxiliar os outros nas suas atividades. Esta função requer, por um lado, a existência de um meio de comunicação

que permita aos Agentes informar os outros de quais são os seus requisitos, por outro lado, um mecanismo interno de decisão sobre como e quando as interações com os outros são apropriadas. Além disso, os Agentes têm a capacidade de reconhecer e distinguir as características de outros Agentes;

- **Limitado**

  As interações com o ambiente e outros Agentes podem estar limitadas ao espaço e ao tempo;

- **Heterogéneo**

  Os comportamentos e as características de um Agente podem variar de acordo com o ambiente e a interação com os outros Agentes;

- **Reativo**

  O Agente observa o ambiente e responde, em tempo útil, a alterações que nele ocorram;

- **Pró-ativo**

  Um Agente não atua apenas em resposta à observação do ambiente, mas possui objetivos e exibe comportamentos oportunistas, sendo capaz de tomar a iniciativa sempre que for apropriado;

- **Racionalidade**

  Um Agente deve agir racionalmente na tentativa de satisfação dos seus objetivos, tentando maximizar o seu desempenho relativamente a uma função de avaliação própria;

- **Adaptativo**

  Um Agente pode ter capacidades de aprendizagem e adaptação do seu funcionamento, em função da experiência acumulada;

- **Conhecimento e Crença**

  Possuir conhecimento é muito mais que possuir informação, trata-se de recolher informação de uma forma dinâmica e de raciocinar sobre a mesma. Definir qual a melhor estratégia de raciocínio a aplicar numa determinada situação demonstra-se necessário (metaconhecimento). Relativamente a uma crença, esta representa a noção atual que o Agente possui sobre um determinado facto. Geralmente, as crenças são dinâmicas, isto é, com o tempo, o seu valor de verdade pode alterar-se;

- **Objetivo**

  Um Agente pode alterar o seu comportamento em função dos seus objetivos a longo prazo. Para tal, os Agentes devem analisar os resultados das suas ações e adaptar o seu comportamento com vista a atingir os seus objetivos. A partir do momento que o Agente expressa a sua disponibilidade para executar determinada tarefa, torna-se responsável por realizar as ações necessárias para tal fim.

- **Veracidade e Benevolência**

  Um Agente deve ser sempre verdadeiro. Benevolência significa que o Agente não deve assumir um comportamento contra produtivo. Aliás, pelo contrário, deve tentar sempre satisfazer os pedidos que lhe são dirigidos.

2.4 ABMS

Agent-Based Modelling and Simulation[7] (ABMS) é uma abordagem, relativamente recente, que modela sistemas complexos, utilizando Agentes. Tal como a definição de Agente não é consensual, também é possível encontrar várias definições de ABMS, sendo que a mais simples é: "ABMS é uma *framework* com Agentes que interagem entre eles e o ambiente onde se encontram" (North e Macal 2007) (Wallace 2009).

North e Macal distinguem ABMS (*Agent-Based Modelling and Simulation*) de ABM(*Agent-Based Modelling*). Enquanto num ABM, os Agentes interagem entre si, em tempo real, para atingir um determinado objetivo, num ABMS a interação entre os Agentes é simulada ao longo do tempo para estudar a dinâmica da interação entre os Agentes. Além disso, o termo Simulação em ABMS significa que o Agente executa as suas tarefas num certo período de tempo.

A Simulação Baseada em Agentes (ABMS) torna mais flexível a tarefa de encontrar soluções alternativas de Modelos complexos. Ademais, a ABMS é fundada na noção de que o todo de um sistema ou organização é mais complexo do que a soma dos seus componentes individuais ("*the whole of many systems or organizations is greater than the simple sum of their constituent parts*")(North e Macal 2007). Podem-se encontrar soluções alternativas, modificando alguns dos Agentes existentes do Modelo ou adicionando novos. Mesmo um ABMS simples pode fornecer informações valiosas sobre um sistema real. A introdução de Agentes com capacidade

[7] Modelação e Simulação Baseada em Agentes

de aprendizagem pode gerar ABMS mais sofisticados, tornando o Modelo de Simulação ainda mais próximos da realidade.

A possibilidade de introdução de Agentes com diferentes características, permite a utilização do ABMS para simular diferentes processos de negócio. Por exemplo, num Modelo que simula uma cadeia de abastecimento, a troca de produtos entre fornecedores e clientes e as mensagens comerciais, podem ser modeladas como troca de mensagens entre o Agentes tipo cliente e os Agentes tipo fornecedor (J. Teixeira e Brito 2003).

Numa *framework* de Agentes em tempo real, estes interagem entre eles e com o ambiente em tempo real. A sincronização das mensagens está garantida. Além disso, num Modelo de Simulação pretende-se analisar, num pequeno período de tempo, o comportamento do sistema num período de tempo real mais longo. Significa que, num período de tempo da ordem das horas ou minutos, pretende-se simular o comportamento de dias ou anos de um sistema real.

Utilizando a abordagem dos Eventos Discretos num simulador, assume-se que o sistema muda instantaneamente em resposta a determinados eventos discretos e que estes são armazenados numa única fila ordenada, em função do tempo da sua execução. Porém, num simulador constituído por uma *framework* de Agentes, cada um é executado separadamente no mesmo computador ou espalhados numa rede de computadores, sendo necessário garantir que a troca de mensagens entre eles está devidamente sincronizada no tempo. Deve-se igualmente assegurar que os eventos são processados na ordem correta, tal como num processo de simulação única (R. Fujimoto 2001).

2.5 Sincronização

Num simulador ABMS, os Agentes envolvidos na Simulação devem estar sincronizados no tempo quando enviam ou recebem uma mensagem. As duas principais abordagens utilizadas para incrementar o Tempo de Simulação são o *Time-step* e o *Next-event* (North e Macal 2007). Apesar de simples de implementar, a técnica *Time-step* tem algumas limitações nas Simulações Analíticas de problemas complexos, sendo mais apropriada a sua utilização em simulações do tipo Ambientes Virtuais. Por sua vez, apesar da utilização da técnica *Next-event* requerer uma

programação mais sofisticada, permite representar de forma mais realista a sequência de eventos complexos nas Simulações Analíticas.

Em Simulação por computador, existem três definições de tempo, com significados diferentes (R. M. Fujimoto 2000) :

- Tempo Real[8], quando se refere ao tempo do sistema ou tempo atual.
- Tempo de Simulação[9], que diz respeito ao tempo virtual em que um Modelo se encontra durante o processo de simulação.
- Tempo Global[10] quando se refere ao tempo virtual do Modelo completo de simulação.

Em Simulações Ambientes Virtuais, os três tipos de tempo são iguais durante toda a simulação.

2.6 HLA

Departamentos de defesa de todo o Mundo recorrem a simuladores em computador para a formação militar, por serem uma alternativa mais económica do que a simulação real e sem risco humano.

O *High Level Architecture* (HLA) é o *standard* IEEE 1516, para a interoperabilidade entre simuladores heterogêneos. Estes podem ser desenvolvidos com diferentes linguagens e/ou plataformas, permitindo uma maior flexibilidade na construção de diversos cenários de simulação. Em particular, o HLA promove a interoperabilidade entre os simuladores e permite a reutilização de Modelos em diferentes contextos (IEEE Std. 2000)(Pereira 2010). Os dois principais componentes do HLA são o *Object Model Template* (OMT) e a *Federate Interface Specification* - este último descreve uma interface de comunicações entre os Modelos de Simulação. A implementação de um conjunto de simuladores (federados) baseados na arquitetura HLA, necessita de um programa denominado *Runtime Infrastructure* (RTI). Este fornece um conjunto de serviços federados que coordenam as operações e troca de dados durante a simulação. Os serviços do RTI são fornecidos pelos federados de acordo com a *Federate Interface Specification.*

[8] Tradução adotada de *Physical time*

[9] Tradução adotada de *Simulation time*

[10] Tradução adotada de *Wallclock time*

2.7 *Software* de Simulação

A evolução dos programas de Simulação está diretamente relacionada com a evolução do *hardware* e *software*. No início da Simulação por computador, o poder de processamento dos computadores era incomparavelmente inferior ao dos nossos dias. As linguagens de programação eram bastante limitadas, requerendo técnicos especializados para a programação. Desta forma, os custos do *hardware* e do seu desenvolvimento tornavam a Simulação acessível somente a empresas com grandes capacidades financeiras.

As primeiras linguagens utilizadas na simulação de computador foram FORTRAN e COBOL, sem a capacidade das atuais linguagens de programação por objetos. Nos anos 60, o esforço necessário para desenvolver um programa de Simulação era considerável, e detetar um erro de programação poderia mesmo ser mais complicado e mais demorado do que desenvolver o Simulador.

Nos anos 80, com o aparecimento dos microcomputadores a custo mais reduzido, com poder de processamento e capacidades gráficas consideráveis, generalizou-se a utilização da Simulação por computador. Entretanto, nos últimos anos, tornaram-se comuns conceitos como: Modelação Visual Interativa; Otimização por Simulação; Realidade Virtual; interação entre diferentes simuladores; bem como simulação nas mais diversas áreas, Simulação distribuída e por fim Simulação com recurso à *World Wide Web* (WWW)(Robinson 2004).

As linguagens de programação atuais, em particular as linguagens de programação por objetos, permitem criar Sistemas Visuais Interativos de Modelação e Simulação com base em pequenos módulos reutilizáveis. Com estas ferramentas, pode ser desenvolvido um Modelo de Simulação que exige, pouca ou nenhuma programação, sendo por isso dada maior ênfase à conceção e à utilização dos Modelos.

Hoje em dia, a Simulação por computador é utilizada em âmbitos tão diversos como a Investigação Científica, Ciências Sociais e Gestão. Em algumas áreas, as ferramentas de Simulação evoluíram de tal modo que passou a ser possível a criação de Modelos de forma interativa, com dimensão e complexidade consideráveis, requerendo poucos ou mesmo nenhuns conhecimentos informáticos (Pidd e Carvalho 2006).

As técnicas utilizadas pelos simuladores podem ser classificadas como Simulação por Eventos Discretos (DES) ou Sistemas Dinâmicos(SD) (Tako e Robinson 2009)(Brailsford e Hilton 2001). Existem algumas diferenças fundamentais entre as duas, resultantes dos princípios subjacentes a cada uma das abordagens de simulação e ao *software* utilizados. Isto é, enquanto o DES modela o sistema com conjunto de entidades e acontecimentos que mudam o estado do sistema em intervalos de tempo discretos, o SD consiste num conjunto de recipientes e fluxos em que as mudanças de estado do sistema ocorrem de forma contínua ao longo do tempo.

Hoje em dia, utilizando linguagens orientadas a objetos, a maioria das ferramentas de Simulação utiliza a metodologia dos Eventos Discretos e em alguns casos o conceito de Agente. Estas ferramentas incluem: um executivo de simulação; um conjunto de ferramentas de modelação e análise; um conjunto de *links* para outros *softwares*, como base de dados, folhas de cálculo e outros *softwares*. (Figura 3) (Pidd e Carvalho 2006)

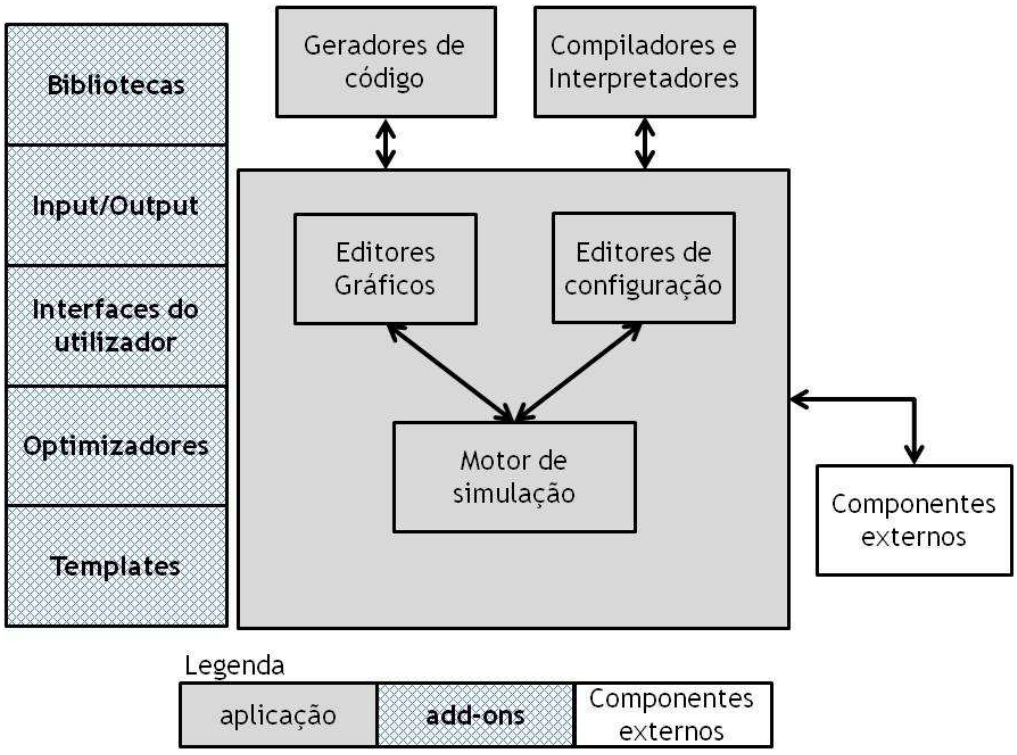


Figura 3 - Ferramenta típica de simulação. Adaptada de Pidd e Carvalho 2006

A evolução dos sistemas informáticos, da capacidade de processamento dos computadores, das linguagens de programação, bem como a existência de meios de comunicação cada vez mais

eficientes, levaram ao aparecimento de ferramentas de Simulação que utilizam o conceito de Agente. Entre as mais conhecidas estão: *Repast, NetLogo, StarLogo, MASON e AnyLogo*. Algumas destas ferramentas são *open source* e permitem a criação de Modelos simples. Contudo, apesar de terem vindo a alargar o seu domínio de aplicação, estas estão normalmente limitadas quanto à complexidade dos sistemas a modelar. Em casos específicos é necessário desenvolver simuladores dedicados, sendo normalmente utilizadas linguagens de programação orientadas a objetos como o JAVA, C++ ou C#.

2.8 Verificação e Validação

A Simulação tornou-se uma poderosa ferramenta para a resolução de problemas e auxiliar na tomada de decisão. A principal preocupação dos programadores e utilizadores de simuladores é verificar a fiabilidade dos resultados. Logo, o utilizador procura criar um Modelo que represente de forma fidedigna o Sistema e assim poder confiar nos resultados da Simulação. Aliás, a Marinha dos EUA afirma que os Modelos devem simular, estimular e emular homólogos do mundo real com a maior precisão possível para ganhar confiança e credibilidade dos utilizadores (Navy 2004). Para tal, é necessária a sua verificação e validação, isto é, os Modelos de Simulação devem ser testados para garantir que os resultados correspondem de facto ao problema em estudo. Alguns autores (Sargent 2011) (Pereira 2010) definem este como um processo de três fases: Verificação, Validação e Acreditação (VV&A).

O processo de desenvolvimento do Modelo da Figura 4 reproduz as interligações entre o Sistema e os Modelos Conceptuais e Computacional. O Sistema representa a entidade a ser modelada, quer seja um sistema real ou em projeto. O Modelo Conceptual é a representação matemática (Simulação) do Sistema em estudo. O Modelo Computacional é a implementação informática do Modelo conceptual. Note-se que, enquanto o Modelo Conceptual é desenvolvido por análise e modelação do Sistema, o Modelo Computacional é obtido com o desenvolvimento de um programa informático que se baseia na informação do Modelo Conceptual. Por fim, o Modelo Computacional desenvolvido é avaliado através da comparação dos resultados obtidos com o Modelo e os dados existentes do sistema real.

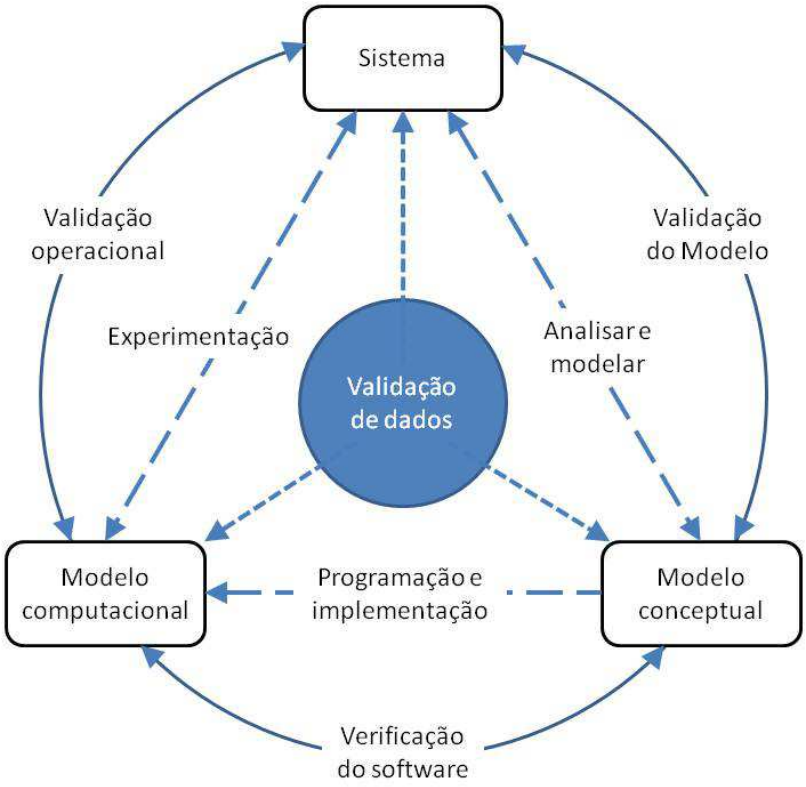


Figura 4 - Verificação e Validação – Adaptada de Sargent 2011

A Marinha dos EUA define Verificação, Validação e Acreditação como sendo:

- Verificação – O processo de determinar que a implementação de um simulador e os dados associados, representam com precisão as especificações do projetista. Após um conjunto de testes sobre as funcionalidades do programa, pode ser afirmado que este funciona da forma como foi projetado.
- Validação - O processo de determinar a correlação entre o simulador e seus dados associados, e a representação do mundo real que se pretende representar. Depois de avaliar a sua fidelidade, pode afirmar-se que o Modelo parece e reage como o sistema real.
- Acreditação – o processo de obter a confirmação que um programa e os dados associados são aceitáveis para utilização num fim específico. A credibilidade do programa permite afirmar que este pode ser utilizado para uma tarefa específica.

A correta implementação do VV&A permite utilizar o simulador com um nível aceitável de confiança, fiabilidade e exatidão.

Capítulo 3.

Linguagens de Negócio

Este capítulo pretende apresentar uma visão geral da linguagem de negócio implementada na *Framework* desenvolvida neste trabalho. Com base nas recomendações da OASIS *Universal Business Language* (UBL), foi desenvolvida uma biblioteca de funções para promover a interoperabilidade semântica entre os simuladores.

Depois de, na primeira secção, se realizar uma breve apresentação de alguns marcos no domínio da normalização, na segunda secção é feita uma apresentação da organização OASIS bem como de vários projetos que lhe estão relacionados direta ou indiretamente. Por fim, são expostas algumas das linguagens de negócio mais utilizadas e também a linguagem adotada neste trabalho.

3.1 Introdução

Nos últimos anos, várias organizações e grupos de empresas tentaram criar *standards* para a troca de mensagens, quer ao nível da sintaxe da linguagem como ao nível do tipo de informação que as mensagens devem conter. O exemplo mais conhecido será possivelmente o hipertexto utilizado na WWW (*World Wide Web*). Outro dos *standards*, menos conhecido pela população em geral mas bastante conhecido da comunidade científica ou ligada às tecnologias da informação, é a linguagem de marcação XML (*Extensible Markup Language*).

A economia digital, por um lado, está a crescer a uma taxa muito superior à da restante economia, por outro a globalização é uma realidade nos dias de hoje. Por isso, acresce a necessidade de criar *standards* que facilitem as trocas de informação entre as organizações, independentemente da sua origem linguística, territorial e sistema de informação utilizados.

Nas mais variadas áreas (Finanças, Transportes, Comércio Eletrónico, etc.), as organizações envolvidas no processo de estandardização surgiram com a criação de consórcios entre empresas e organismos governamentais. As Nações Unidas (UN) e a Comissão Europeia (EU) são duas dessas entidades governamentais. A *Organization for the Advancement of Structured Information Standards* (OASIS) é o consórcio com mais *standards* disponibilizados gratuitamente e o patrocínio de empresas internacionais das mais diversas áreas de interesse.

3.2 Normalização de Comunicações

A OASIS (*Organization for the Advancement of Structured Information Standards*) foi fundada em 1993, inicialmente com o nome de "*SGML Open*", adquirindo posteriormente, em 1998, a designação atual. Trata-se de um consórcio, sem fins lucrativos, que impulsiona o desenvolvimento de padrões abertos para a sociedade da informação. Em 2002, a OASIS e quatro organizações internacionais juntaram-se num memorando de entendimento (MoU)[11], tornando-se na organização, que disponibiliza gratuitamente *standards*, mais reconhecida mundialmente tanto por entidades governamentais como por entidades ligadas ao desenvolvimento de *software*.

As organizações que fazem parte do consórcio são (OASIS 2013):

- AMQP (*Advanced Message Queuing Protocol*): que define um padrão aberto para a troca de mensagens entre as aplicações e organizações;
- CGM Open (*Advancing Standards for Graphical Information Interchange*): desenvolve e mantem a aplicação WebCGM - um padrão para a criação de documentos electrónicos;
- eGov (*eGovernment*): promove a discussão dos requisitos de padronização dos *e-business* das administrações governamentais;
- EI (*Emergency Interoperability*): fomenta o desenvolvimento, adoção, utilização e implementação de padrões de interoperabilidade e comunicação de emergência;

[11] http://xml.coverpages.org/MOU-OASIS-200202.html

- IDtrust (*Identity and Trusted Infrastructure*): promove a compreensão e adoção de normas e identificação baseadas em infraestruturas seguras;
- LegalXML (*Legal Data Exchange*): incentiva a criação de padrões para a troca eletrónica de dados jurídicos;
- Open CSA (*Composite Services Architecture*): fomenta o desenvolvimento de padrões de desenvolvimento de aplicações *Service-Oriented Architecture* (SOA), com recurso às especificações dos *Service Component Architecture* (SCA) e *Service Data Objects* (SDO);
- WS-I (*Web Services Interoperability*) : promove a utilização de "Melhores Práticas" em diversas plataformas, sistemas operativos e linguagens de programação.

A necessidade de criar *standards* para as comunicações entre as diferentes entidades (empresas, organismos governamentais, cidadãos, sistemas informáticos) é patente, nomeadamente pelo número de iniciativas em curso em todo o mundo. Criada em 2010 pela União Europeia, a Agenda Digital para a Europa (DAE) tem como objetivo revitalizar a economia dos seus Estados Membros e ajudar os seus cidadãos e empresas a tirar o máximo proveito das tecnologias digitais. Como áreas prioritárias destacam-se: a promoção da interoperabilidade entre os sistemas; a concessão do acesso a redes mais rápidas; questões de confiança e segurança; o desenvolvimento das competências e apoio à investigação e, por fim, a inovação no domínio das TIC. Sendo assim, no calendário dos objetivos da DAE (Figura 5) torna-se elucidativo a necessidade da existência de *standards* que façam com que a sua concretização seja possível.

Com um orçamento de 164.1 milhões de euros, para o período entre 2010 e 2015, o "*Interoperability Solutions for European Public Administrations*" (ISA), é um programa que procura lidar com as questões da DAE relacionadas com a interoperabilidade, a reutilização e a partilha de informação entre as Administrações Públicas Europeias. Um dos projetos deste programa é o "*eGovernment Large Scale Pilot Projects*" (LSPs). Este envolve as entidades públicas, prestadores de serviços e centros de investigação da União Europeia, na implementação de soluções comuns para integrar serviços públicos *online* e torná-los acessíveis em toda a Europa. Os projetos em curso, em 2013, são:

- e-CODEX: Projeto na área da justiça que tem como objetivo a criação de funcionalidades digitais para facilitar a troca de informações legais entre os países da União Europeia;

- epSOS: Projeto na área da saúde que, já tendo um protótipo em teste em ambiente real, procura promover a troca de documentos de saúde eletrónicos entre países;
- STORK 2.0: Projeto que tem como objetivo estabelecer uma plataforma eletrónica de identificação na União Europeia que permita que qualquer um dos seus cidadãos se identifique em qualquer país da União através do seu documento nacional;
- PEPPOL: Projeto com o objetivo de criar as especificações para os processos de compra da Administração Pública dos membros da União Europeia;
- SPOCS: Projeto que procura criar procedimentos eletrónicos transfronteiriços compatíveis com os procedimentos de todos os países da União Europeia.

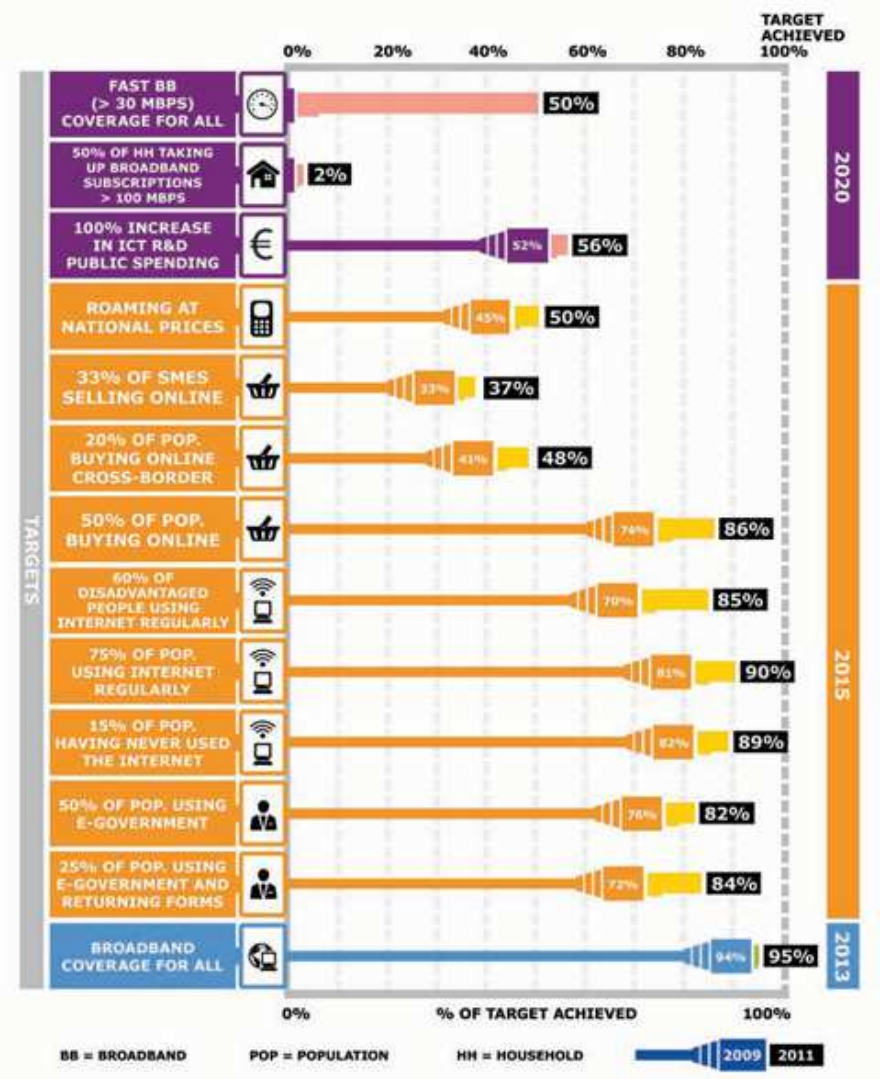


Figura 5 - Calendário dos objetivos da Agenda Digital para a Europa (DAE 2013)

A Tabela 3 mostra, sucintamente, alguns dos *standards* OASIS que estão de alguma forma relacionados com o presente trabalho.

Tabela 3 - OASIS XML *Standards*

Standard	Descrição	
WS-BPEL	BPEL usa *Web services standards* para descrever as atividades de processos de negócios. Define como as atividades podem ser compostas para realizar tarefas específicas (WS-BPEL 2007).	
DITA	DITA (*Darwin Information Typing Architecture*) circunscreve um conjunto de documentos tipo para a criação, organização e publicação de informações (OASIS DITA 2010).	
ebXML	ebXML (*Electronic Business using eXtensible Markup Language*) estabelece um conjunto de especificações para as empresas realizarem negócios através da Internet, independente da sua dimensão e localização geográfica (ebXML 2013). ebXML inclui especificações que podem ser implementadas separadamente ou em conjunto:	
	ebBP	O ebBP (*Business Process*) define uma base de processos de negócios que promovem a troca de mensagens entre negócios usando XML. (OASIS ebBP 2002)
	CPPA	O CPPA (ebXML *Collaboration Protocol Profile and Agreement*) define como identificar os parceiros comerciais envolvidos na troca de mensagens eletrônicas (OASIS CPPA 2002).
	ebMS	O ebMS (ebXML *Messaging Services*) fixa o protocolo de comunicações para a troca de mensagens utilizando a internet.(OASIS ebMS 2002)
	RIM	O RIM (ebXML *Registry Information Model*) define os repositórios e os registos para permitir a consulta e edição dos dados do repositório.(OASIS RIM 2002)
	CCTS	O CCTS (ebXML *Core Components Technical Specification*) apresenta uma metodologia para o desenvolvimento de um conjunto de blocos semânticos que procuram representar dados corporativos genéricos.(OASIS CCTS 2003)
SAML	O SAML (*Security Assertion Markup Language*) define uma linguagem XML-*based* para criar e trocar, de forma segura, informação entre entidades que estão ligadas entre si pela internet (OASIS SAML 2005).	
UBL	A UBL (*Universal Business Language*) estabelece o formato XML dos documentos de negócio adaptável às necessidades específicas das mais variadas indústrias (OASIS UBL 2011).	
ODF	O ODF (*OpenDocument Format*) é um formato XML para aplicações que geram ficheiros do tipo documentos de texto, folhas de cálculo e elementos gráficos, de modo a permitir a troca de ficheiros entre aplicações de diferentes fornecedores (OASIS ODF 2006).	
UDDI	O UDDI (*Universal DescriptionDiscovery & Integration*) define um método para descobrir e invocar *Web services* de forma dinâmica (OASIS UDDI 2005).	

3.3 Comércio Eletrónico

Num processo de negócio envolvendo várias empresas, cada empresa trabalha de forma independente, efetuando trocas de informações com outras empresas. Estas informações podem estar associadas a encomendas, documentos contabilísticos, produtos e / ou serviços. A troca de informação entre elas é cada vez mais um processo eletrónico, onde a troca de mensagens recorre a uma linguagem entendida pelas partes envolvidas.

As entidades envolvidas, no mundo real ou numa simulação, podem ser bastante heterogéneas. Por exemplo, numa fábrica existem processos de transformação de matérias-primas em produtos finais, enquanto num armazém existem apenas atividades de armazenamento e manuseamento e, no distribuidor final apenas ações de compra e venda (J. Teixeira e Brito 2003). A Figura 6 apresenta os diferentes tipos de mensagens existentes numa cadeia de abastecimentos:

- EAI (*Enterprise Application Integration*): viabiliza a interação entre os diferentes sistemas dentro das organizações. Note-se que, mesmo sendo vantajoso a utilização de *standard*, não é obrigatória, pois o domínio de aplicação está limitado à própria organização;
- B2G (*Business to Governement*): representa as trocas de mensagens entre empresa e entidades governamentais. O projeto PEPPOL, referido no Capítulo 3.2 é um exemplo da importância da existência de um *standard* para troca deste tipo de mensagens;
- C2G (*Citizen to Government*): representa a troca de mensagens entre o cidadão e as instituições governamentais;
- B2C (*Business to Customer*): representa as trocas de mensagens entre a entidade fornecedora de produtos ou serviços e a entidade cliente. A entidade fornecedora tanto pode ser uma empresa produtora do produto ou serviço ou uma entidade revendedora. A entidade cliente é o cliente final;
- B2B (*Business to Business*): também consiste nas trocas de mensagens entre a entidade fornecedora de produtos ou serviços e a entidade cliente. Porém, tanto a entidade fornecedora como a entidade cliente, pode ser uma empresa produtora do produto ou serviço ou uma entidade revendedora.

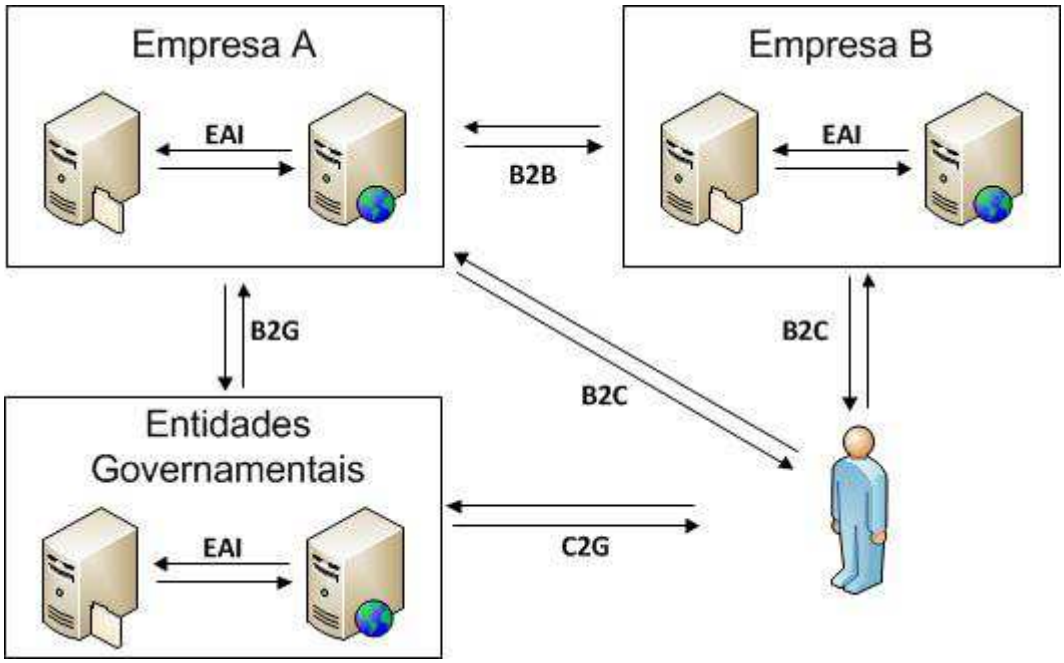


Figura 6 – Tipos de mensagens comerciais

Possivelmente, o EDI (*Electronic Data Interchange*) foi o primeiro *standard* para a troca de mensagens eletrónicas, utilizado no envio de encomendas, faturas e informação sobre os produtos transacionados. A geração seguinte de *standards* B2B recorre ao ebXML (*Electronic Business Using XML*) que utiliza a linguagem de marcação XML para o envio de mensagens. Este, além das mensagens do EDI, inclui também um conjunto de mensagens descritivas do processo de negócio ebBP (ebXML *Business Process*).

3.3.1 UN/EDIFACT

Nos anos 80, surgiu o EDI (*Electronic Data Interchange*), um *standard* que propunha uma sintaxe para a troca de mensagens eletrónicas. Contudo, o EDI tinha algumas limitações nas transações internacionais, em particular nas intercontinentais. Assim, nos anos 90, sobre o patrocínio das Nações Unidas, o UN/EDIFACT (*the United Nations rules for Electronic Data Interchange for Administration, Commerce and Transport*) tornou-se um *standard* internacionalmente reconhecido que contém um conjunto de diretrizes para a troca de dados eletrónicos entre sistemas informáticos independentes.

A Tabela 4 mostra um exemplo de uma mensagem EDIFACT que representa a resposta a um envio de catálogo de preços de um fornecedor. Mesmo conhecendo a sintaxe deste *standard*, a

leitura da mensagem é impossível sem o recurso ao dicionário da norma para identificar o significado das siglas da mensagem.

Tabela 4 - Mensagem EDIFACT

Mensagem	Significado
UNH+ME000052+PRICAT:D:01B:UN:EAN009'	Cabeçalho da mensagem
BGM+51+PC34888+29'	Preço do catálogo de venda número PC34888
DTM+137:20021221:102'	Data da mensagem
RFF+PL:AVS321'	Referencia do pedido AVS321
DTM+171:20021215:102'	Data do preço 15 de Dezembro de 2002
NAD+BY+5412345000020::9'	Identificação do comprador GLN 5412345000020
NAD+SU+4012345500004::9'	Identificação do fornecedor GLN 4012345500004
UNT+8+ME000052'	Número de linhas da mensagem 8

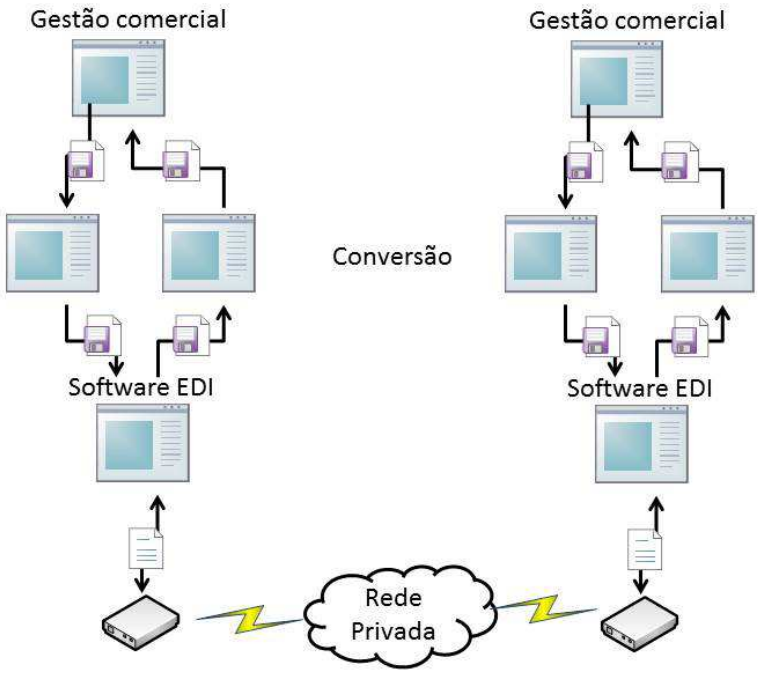


Figura 7 – Arquitetura de uma implementação EDI. Adaptada de Schlegel 2005

A arquitetura de uma implementação EDI é relativamente simples (Figura 7). Necessita de um conjunto de *software* de conversão da informação dos sistemas de informação da empresa e a linguagem EDI, de uma rede de comunicações e do equipamento de comunicação. Nos anos 80 e 90, a fiabilidade das comunicações era reduzida em comparação com as comunicações atuais.

Além dos problemas de comunicação, o maior problema estava no *software* de conversão que nem sempre era de fácil implementação, ocorrendo erros de conversão com frequência.

Apesar de o EDIFACT ter sido adotado por muitas empresas, maioritariamente de grande dimensão, tinha limitações (Tabela 5) que impediram o seu uso generalizado.

Tabela 5 - Limitações do EDIFACT

Nível	Problema
Sintaxe	• A estrutura das mensagens é ambígua em algumas situações • Cada nova mensagem requer um grande esforço de implementação para a sua normalização
Semântica	• Nem sempre a finalidade da mensagem é clara • As regras de validação do negócio são expressas em linguagem natural • Mensagens genéricas nem sempre de fácil interpretação
Financeiro	• Custo elevado de implementação • Necessário muito tempo para implementar uma solução inicial e introdução de novas mensagens
Domínio	• Generalizada nas grandes empresas • Utilizada em empresas de média dimensão para comunicarem com as de grande dimensão • Não utilizada em empresas de pequena dimensão dados os custos da solução

Em contrapartida, a linguagem de marcação XML, aprovada em 1988 pela W3C, tem sido uma alternativa para a resolução dos problemas de sintaxe do EDIFACT, em conjunto com a utilização de *standards* de comunicação definidos pela OASIS, utilizados na resolução dos problemas de semântica.

3.3.2 ebXML

A introdução da linguagem de marcação XML como suporte das mensagens e esquemas de negócio XML (XML *business schemas*) generalizou-se rapidamente com as seguintes vantagens (OASIS UBL 2011):

- Baixo custo de integração, conseguido com a partilha entre várias empresas de estruturas de dados comuns;

- Menor custo do *software* comercial, pois o custo de criação de ficheiros XML é menor do que o de codificar de acordo com o *standard* EDI;
- Curva de aprendizagem mais fácil;
- Generalização da utilização em pequenas e médias empresas, em virtude do menor investimento necessário à implementação;
- Formação padronizada permite a formação de mais técnicos qualificados.
- Fácil integração nos mais variados sistemas;
- Disponíveis várias ferramentas de baixo custo ou mesmo do domínio público, para tratar e analisar *inputs* e *outputs*.

O ebXML (*Electronic Business Using* XML) é o resultado de uma iniciativa, bem sucedida, da OASIS e da UN/CEFACT, iniciada em 1999 com o objetivo de criar um standard XML para a troca de mensagens eletrónicas entre organizações, independentemente da sua dimensão e do sistema informático utilizado. Além disso, fornece um conjunto de especificações para a criação de ferramentas B2B.

Ademais, o ebXML pode ser utilizado em empresas ou organizações de qualquer dimensão, organizações sem fins lucrativos, governos e mesmo entidades individuais (Schlegel 2005).

3.3.3 UBL

O OASIS *Universal Business Language* (UBL) é um *standard* genérico que recorre à linguagem XML para especificar documentos de negócio tipo catálogos, encomendas, faturas, etc. O UBL responde às necessidades particulares de uma grande maioria das empresas envolvidas numa cadeia de abastecimento (OASIS UBL 2011). Para diversas organizações industriais, documentos tipo encomendas ou faturas podem ser gerados em formato XML, de acordo com as especificações UBL.

O *standard* UBL foi projetado para fornecer, universal e gratuitamente, uma sintaxe que respeita os requisitos legais dos *standards* de negócio ISO 15000 (ebXML) e também para facultar a empresas de qualquer dimensão, uma solução completa com os benefícios do anterior *standard* EDI. Sendo assim, o UBL fornece (OASIS UBL 2011):

- Um formato genérico XML para a permuta de documentos de negócios, com possibilidade de atender aos requisitos das mais variadas áreas de negócio.
- Uma biblioteca de esquemas XML para componentes de dados reutilizáveis, tais como "Endereço", "*Item*" e "Pagamento", bem como de outros elementos de dados comuns a documentos de negócios de todos os dias.
- Um conjunto de esquemas XML para documentos de negócios, como "Encomendas", "Faturas", etc., que são construídos a partir dos componentes da biblioteca UBL.

Tal como o *strandard* ebXML, a Biblioteca UBL é baseada num modelo conceitual de componentes de informação, conhecido como *Business Information Entities* (BIEs) (Figura 8).

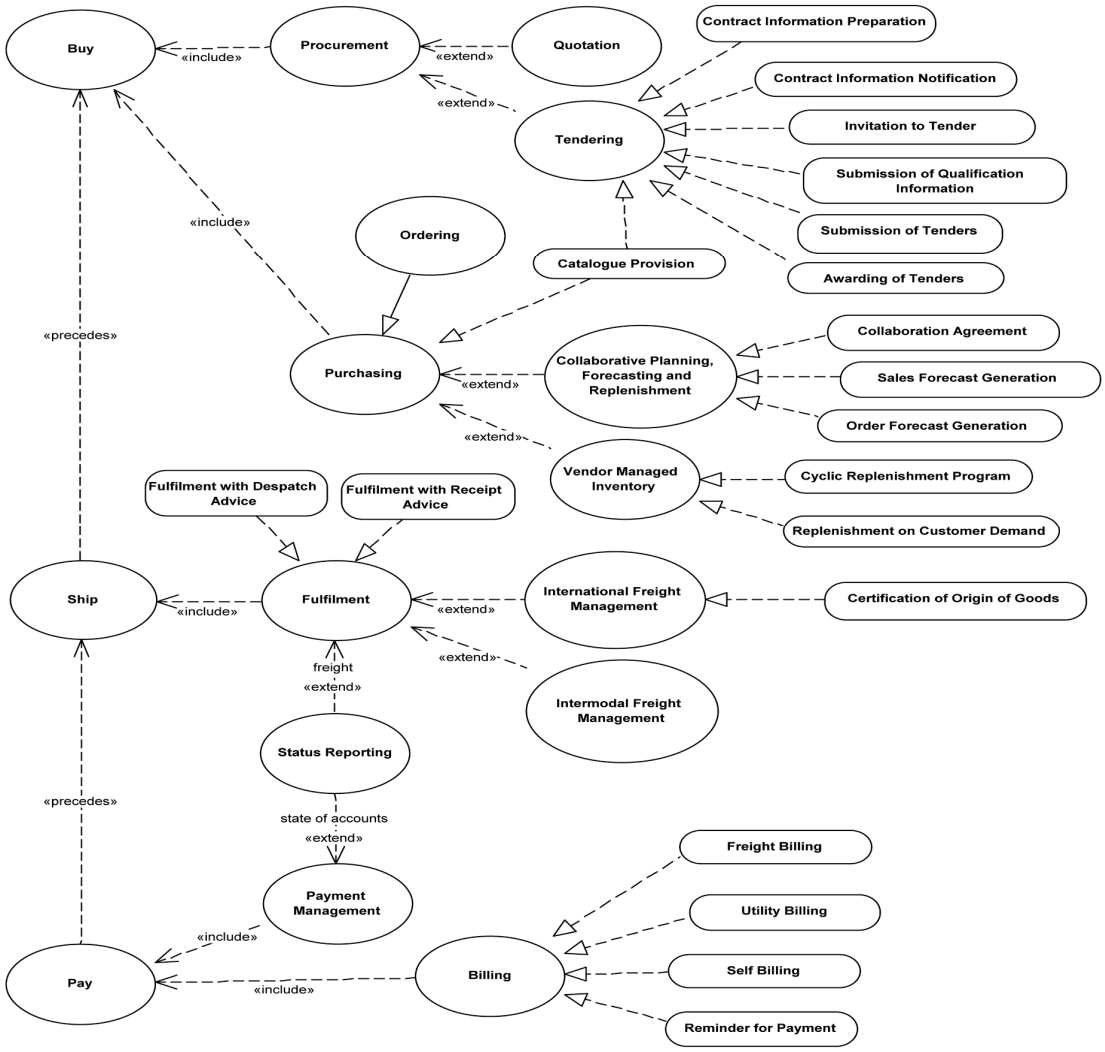


Figura 8 - UBL 2.1 Use Case (OASIS UBL 2011)

Estes componentes são utilizados em vários documentos tais como Encomendas, Faturas, e outros documentos de negócio (Anexo A-Tipos de documentos UBL 2.1).

O UBL define um conjunto de componentes utilizados em diferentes documentos. Alguns dos mais utilizados são:

- *Parties* - Um *Party* pode identificar uma entidade ou grupo de entidades envolvidas no processo de negócio (Tabela 6).
- *Items* - Um *Item* pode ser um produto ou um serviço (Tabela 7).

Tabela 6 - Exemplo *Party*

```
<Party>
    <PartyName>
        <Name>Consortial</Name>
    </PartyName>
    <PostalAddress>
        <StreetName>Busy Street</StreetName>
            ……
    </PostalAddress>
    <PartyTaxScheme>
        …….
    </PartyTaxScheme>
    <Contact>
        ….
    </Contact>
</Party>
```

Tabela 7 - Exemplo *Item*

```
<LineItem>
    <ID>1</ID>
    <SalesOrderID>A</SalesOrderID>
    <LineStatusCode>NoStatus</LineStatusCode>
    <Quantity unitCode="KGM">100</Quantity>
    <LineExtensionAmount currencyID="GBP">100.00</LineExtensionAmount>
    <TotalTaxAmount currencyID="GBP">17.50</TotalTaxAmount>
    <Price>
        <PriceAmount currencyID="GBP">100.00</PriceAmount>
        <BaseQuantity unitCode="KGM">1</BaseQuantity>
    </Price>
    <Item>
        <Description>Acme beeswax</Description>
        <Name>beeswax</Name>
        <BuyersItemIdentification>
            <ID>6578489</ID>
        </BuyersItemIdentification>
        <SellersItemIdentification>
            <ID>17589683</ID>
        </SellersItemIdentification>
    </Item>
</LineItem>
```

Os componentes definidos no UBL são utilizados de igual modo nos diferentes documentos. Estes estão organizados em processos, de acordo com as suas características e área de aplicação (Tabela 8).

Tabela 8 - Tipos de documentos UBL

Área de aplicação	**Processos**
Aprovisionamento	Concursos Catálogo Cotação Encomenda Distribuição Faturação Pagamento Reabastecimento
Transportes	Gestão de Fretes nacionais e internacionais Gestão de fretes intermodais
Comércio Internacional	Certificação de origem das mercadorias

Cada processo tem um objetivo, documentos associados e regras para a sua utilização. Por exemplo, o Catálogo é um documento produzido por uma entidade da cadeia de abastecimentos que descreve os itens e preços. Os documentos associados ao processo "*Catalogue*" são: *"Catalogue Request", "Application Response", "Catalogue Item Specification Update", "Catalogue Pricing Update",* e *"Catalogue Deletion"*. A Figura 9 mostra o processo de criação de um Catálogo a pedido de um cliente. Neste processo são trocados, entre o fornecedor e o cliente, documentos do tipo *"Catalogue Request", Application Response", "Catalogue"* e *"Application Response"*.

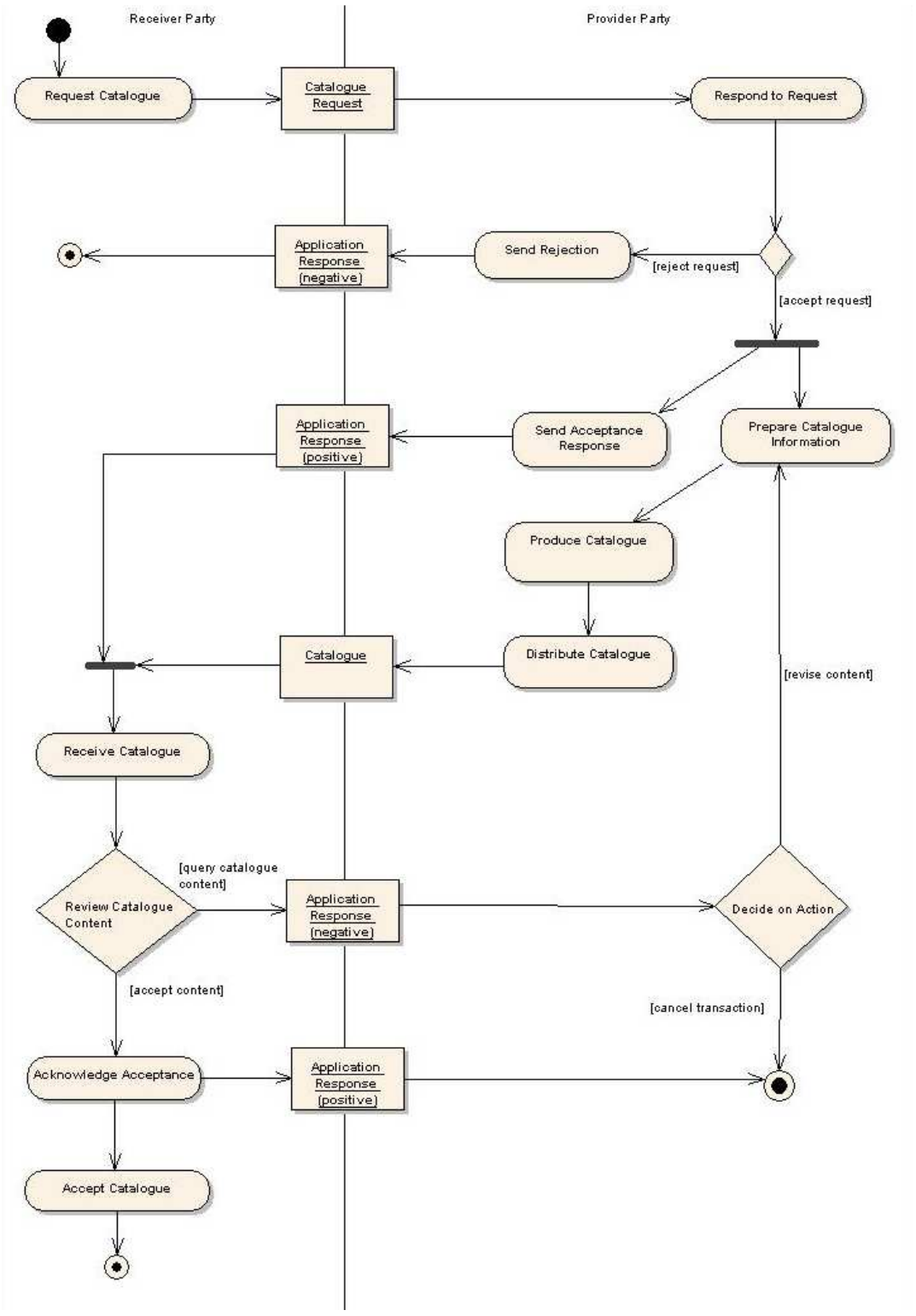


Figura 9 – Processo de criação do Catálogo (OASIS UBL 2011)

Relativamente ao *CatalogueRequest, trata-se* um documento utilizado para solicitar um catálogo de um vendedor.

Tabela 9 - Exemplo *CatalogueRequest*

```
<CatalogueRequest …... >
    <UBLVersionID>2.1</UBLVersionID>
    <ID>479e66f0-043e-44a9-a7d5-c8e0d6c12201</ID>
    <Name>CatalogueRequest (1)</Name>
    <IssueDate>0001-01-01</IssueDate>
    <ReceiverParty>
        <PartyIdentification>
            <ID>6e1baa64-dc1e-4e60-b95f-dd619699fef3</ID>
        </PartyIdentification>
        <PartyName>
            <Name>Vendedor</Name>
        </PartyName>
    </ReceiverParty>
</CatalogueRequest>
```

Por sua vez, o *ApplicationResponse* é um documento para indicar a resposta a uma mensagem recebida que indica se a mensagem foi aceite ou recusada.

Tabela 10 - Exemplo *ApplicationResponse*

```
<ApplicationResponse ……>
    <UBLVersionID>2.1</UBLVersionID>
    <ID schemeID="7c9e0baf-015c-4a8f-b5c5-618fe71dfb5c">
        ApplicationResponse (AgentBaseFactory_1 - 1)
    </ID>
    <IssueDate>2013-02-13</IssueDate>
    <DocumentResponse>
        <Response>
            <ResponseCode>
                Accepted
            </ResponseCode>
        </Response>
        <DocumentReference>
            <ID>428b386e-7e7a-4095-8967-b83eb52fbb1b</ID>
            <DocumentType>CATALOGUEREQUESTTYPE</DocumentType>
        </DocumentReference>
    </DocumentResponse>
</ApplicationResponse>
```

Catalogue é um documento que descreve os artigos disponíveis para venda, por um fornecedor. A informação disponibilizada pode ser mais ou menos completa, isto é, além do nome e referências do artigo, pode conter informação como o preço, unidade de medida, prazos de entrega, condições de utilização, etc.

Tabela 11 - Exemplo *Catalogue*

```
<Catalogue xmlns:….>
   <UBLVersionID>2.1</UBLVersionID>
   <ID schemeID="79a2cd7f-0ae1-45ff-b30d-da36c8212eeb">
      Catalogue (AgentBaseFactory_1 - 1)
   </ID>
   <IssueDate>2013-02-13</IssueDate>
   <Note>catalog 1</Note>
   <DocumentReference>
      <ID>428b386e-7e7a-4095-8967-b83eb52fbb1b</ID>
      <DocumentType>CATALOGUEREQUESTTYPE</DocumentType>
   </DocumentReference>
   <ProviderParty>
      <PartyIdentification>
         <ID>6e1baa64-dc1e-4e60-b95f-dd619699fef3</ID>
      </PartyIdentification>
      <PartyName>
         <Name>AgentBaseFactory_1</Name>
      </PartyName>
   </ProviderParty>
   <CatalogueLine>
      <ID>fc498e89-8c25-44ee-99b8-1e58d0a5ec3a</ID>
      <OrderableUnit>36</OrderableUnit>
      <MinimumOrderQuantity>3</MinimumOrderQuantity>
      <MaximumOrderQuantity>1</MaximumOrderQuantity>
      <RequiredItemLocationQuantity>
         <Price>
            <PriceAmount currencyID="AFN">2.74</PriceAmount>
            <BaseQuantity unitCode="36">0.25</BaseQuantity>
         </Price>
      </RequiredItemLocationQuantity>
      <Item>
         <Description>Description 4</Description>
         <Name>Name 4</Name>
         <SellersItemIdentification>
            <ID>4</ID>
         </SellersItemIdentification>
      </Item>
   </CatalogueLine>
   <CatalogueLine>
      …
      …
   </CatalogueLine>
   …
   …
</Catalogue>
```

Outros exemplos de processos de negócios estão definidos no UBL V2.1 (OASIS UBL 2011) para as mais variadas indústrias.

Capítulo 4.

Arquitetura da *Framework*

Este capítulo pretende apresentar uma visão geral do domínio de aplicação da *Framework* e da sua arquitetura. Depois de, na primeira secção, se realizar uma breve apresentação do tipo e características dos elementos envolvidos na simulação, na segunda secção é feita uma apresentação da abordagem seguida na resolução dos problemas de sincronização entre os Agentes envolvidos na simulação. Por fim é apresentada a arquitetura proposta para a *Framework* em desenvolvimento.

4.1 Considerações Gerais

Uma Cadeia de Abastecimento[12] é o conjunto de entidades que inclui desde o fornecedor da matéria-prima ou componentes até ao consumidor do produto final, bem como as ligações entre as diferentes entidades (Figura 10). Numa cadeia de abastecimento, temos vários tipos de entidades tipo: Fornecedor; Produtor; Armazenista; Retalhista e Consumidor final.

Independentemente das suas características, numa cadeia de abastecimento, duas entidades que estão interligadas, desempenham entre elas o papel de cliente e/ou fornecedor. Na Figura 10, as entidades do grupo Fornecedores são fornecedoras das entidades do grupo Fabricantes, enquanto que as entidades do grupo Fabricantes são clientes das entidades do grupo Fornecedores e por sua vez são fornecedoras do grupo Distribuição, sendo esta logística transversal a todas as entidades.

[12] Tradução adotada de *supply chain*

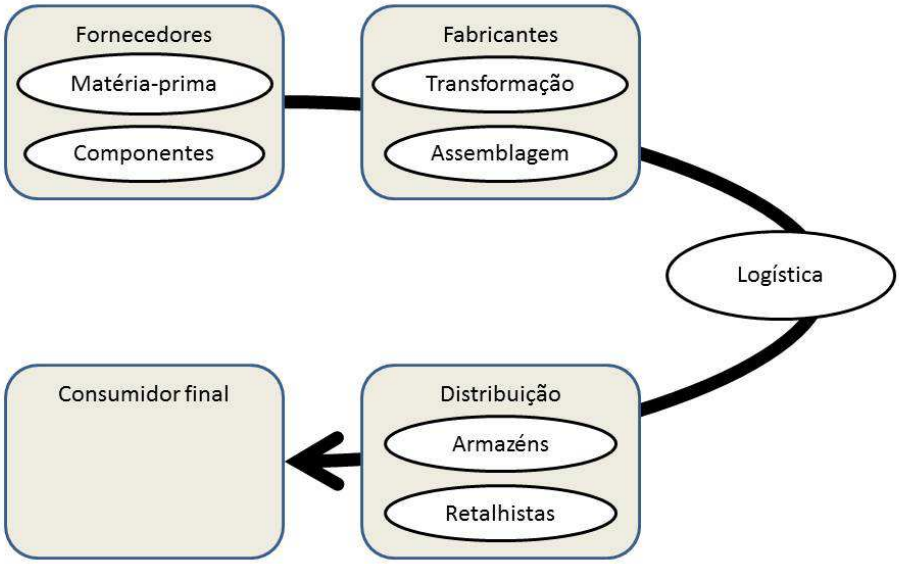


Figura 10 - Cadeia de abastecimento

Num Modelo simplificado de uma qualquer subparte da cadeia de abastecimento, as mensagens trocadas entre duas entidades interligadas são semelhantes, independentemente da sua posição na cadeia de abastecimento. Por exemplo, no sub-Modelo que contém uma fábrica e os seus clientes e fornecedores (Figura 11), as mensagens trocadas entre os clientes e a fábrica são semelhantes às mensagens trocadas entre a fábrica e os fornecedores.

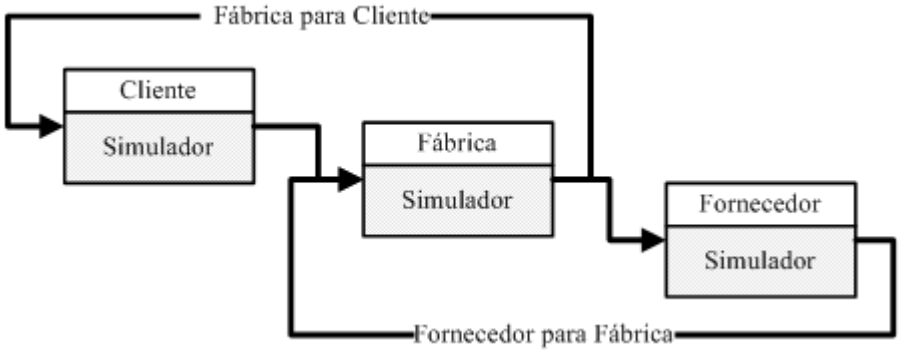


Figura 11 - Modelo de negócio simples

Entre os clientes e a fábrica trocam-se as seguintes mensagens:

- Pedidos de catálogo dos produtos ou serviços e condições de fornecimento.
- Receção de catálogos
- Pedidos de encomendas.

- Receção de resposta de aceitação ou recusa de encomenda.
- Receção dos produtos e a respetiva fatura.

Por sua vez, entre a fábrica e os fornecedores são trocadas as mensagens:

- Envio o catálogo dos produtos ou serviços e condições de fornecimento.
- Receção de pedidos de encomendas.
- Envio da resposta ao pedido de encomenda.
- Envio dos produtos e a respetiva fatura.

Numa *framework* de simulação ABMS, uma cadeia de abastecimento pode ser modelada com um conjunto de Agentes programados para enviar e receber as mensagens anteriormente referidas. Se o comportamento interno do Agente for parametrizável, este pode desempenhar qualquer papel na cadeia de abastecimento. Sendo assim, um mesmo Agente tanto pode desempenhar o papel de cliente como de fornecedor. A cadeia de abastecimento pode assim ser modelada, numa *framework* ABMS, com um conjunto de clones do mesmo Agente e um gestor das comunicações.

Com um Agente configurável, um Modelo mais complexo pode ser modelado, envolvendo uma fábrica e vários clientes e fornecedores (Figura 12), sem necessidade de programação de cada entidade.

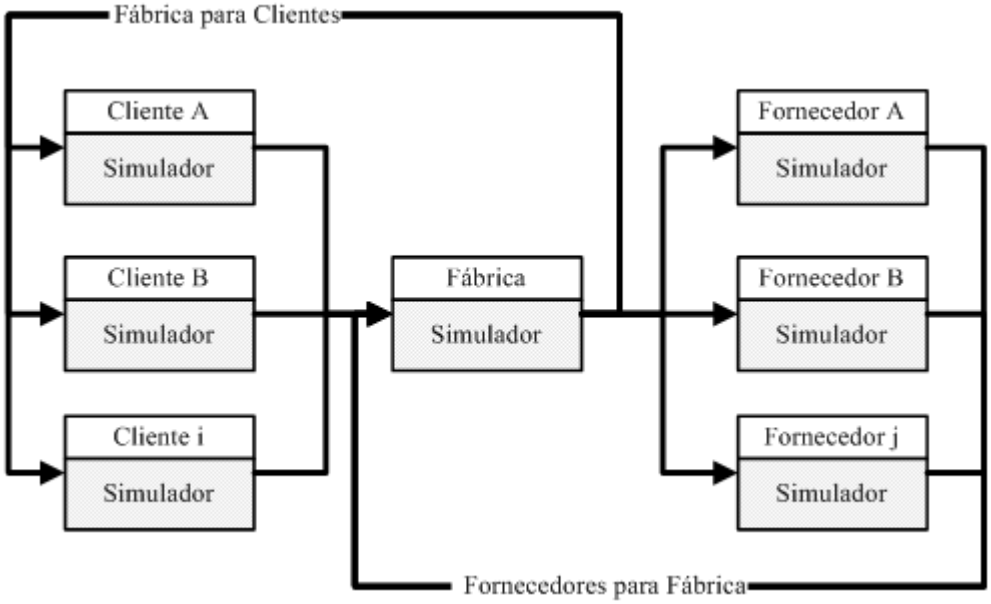


Figura 12 - Modelo de negócio com Agentes independentes

A complexidade e nível de detalhe dos Modelos de simulação podem variar entre uma representação simplificada e uma representação complexa do Sistema. Porém, dificilmente se pode definir qual dos Modelos é a melhor escolha. Isto é, um Modelo simples, devido ao seu pouco detalhe, pode conduzir a maus resultados. Por outro lado, um Modelo complexo, apesar de ser natural que conduza a melhores resultados devido ao seu maior detalhe, poderá exigir demasiado tempo para a sua geração, execução e análise de resultados.

A simulação por ABMS permite modelar de uma forma independente cada uma das entidades envolvidas no Sistema e criar um Modelo global com maior ou menor detalhe, alterando ou substituindo individualmente cada um dos Agentes. Cada uma das entidades pode ser modelada por Agentes com um nível de detalhe variável. Além disso, num estudo inicial, pode ser criado um Modelo simplificado (Figura 13) e progressivamente serem introduzidos maiores níveis de detalhe (Figura 14), de acordo com o objetivo do estudo.

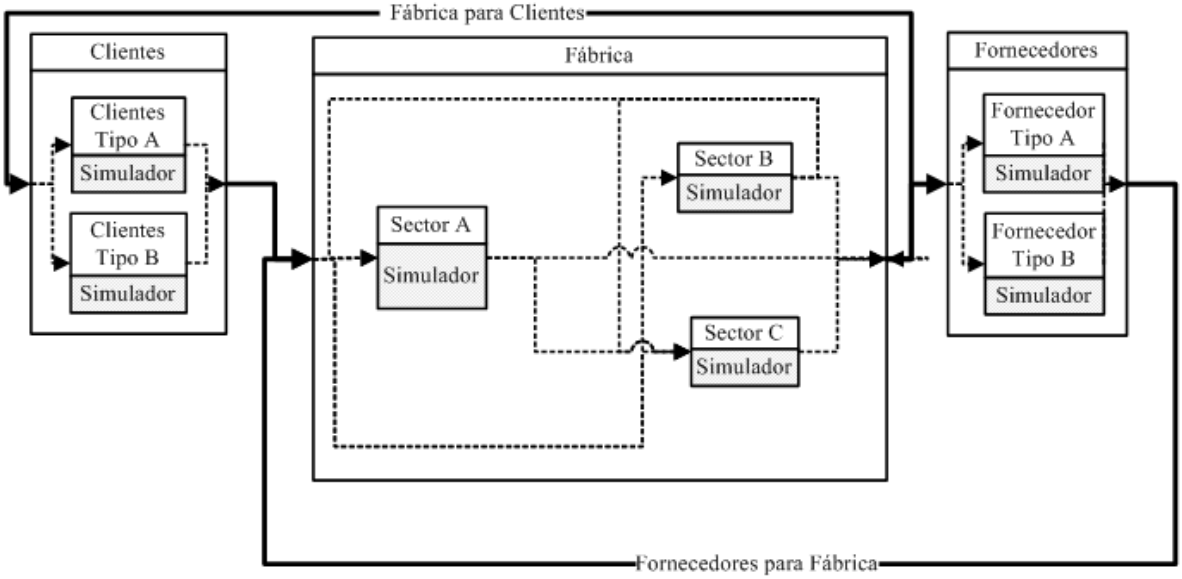


Figura 13 - Modelo Simples

Novos Modelos de Simulação podem ser criados pela troca de um Agente por outro com um nível de detalhe mais elevado, sem necessidade de alterar o restante Modelo (Figura 14). No exemplo demonstrado, a fábrica troca mensagens com os clientes e fornecedores, independentemente do grau de detalhe utilizado no Modelo.

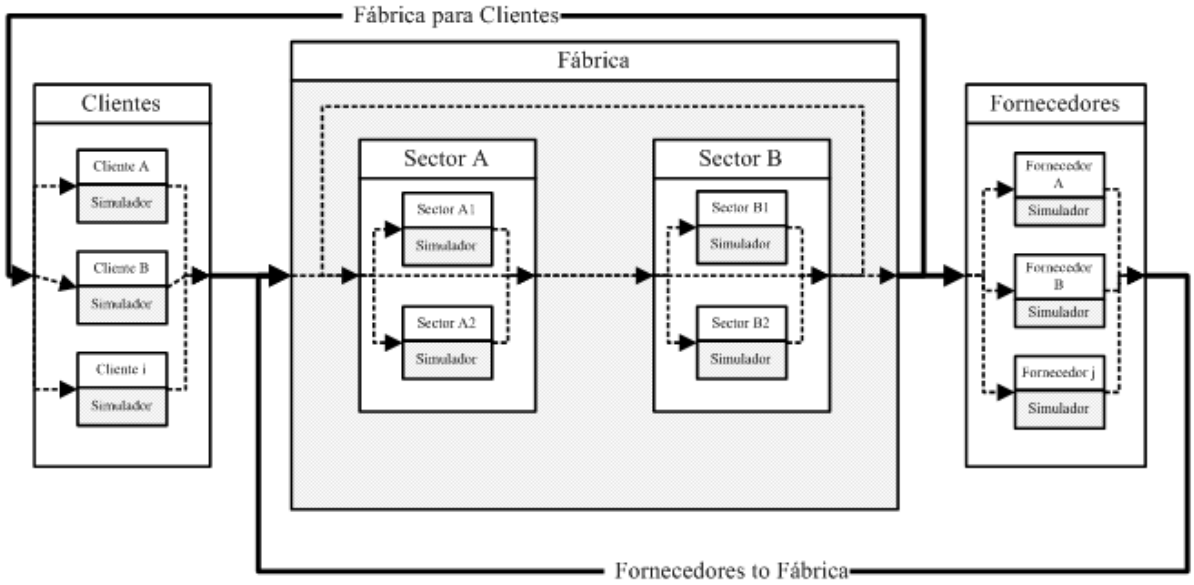


Figura 14 - Modelo detalhado

Utilizando uma abordagem recursiva, é possível modelar um sistema com o nível de pormenor pretendido, ou seja pode-se iniciar com uma solução de alto nível e depois aumentar gradualmente o detalhe do Modelo em setores críticos.

4.2 Sincronização

Conforme já referido do ponto 2.5, *Time-step* e *Next-event* são as duas principais abordagens utilizadas para incrementar o Tempo de Simulação. Apesar da abordagem *Time-step* ser de implementação mais simples, a *Next-event* permite representar de forma mais realista a sequência de Eventos complexos em Simulações Analíticas.

Numa cadeia de abastecimentos constituída por vários elementos (fábricas, armazéns, distribuidores, etc.), a frequência de acontecimentos variam de elemento para elemento. Podemos ter elementos com uma frequência de acontecimentos elevada e outros com frequência baixa. Na abordagem *time-step*, o incremento do Tempo de Simulação tem de ser o mesmo para todos os elementos envolvidos. Isto é, se a unidade de tempo de um dos elementos envolvidos na cadeia de abastecimento for a hora, então todos os outros elementos têm de adotar a hora como unidade de tempo. O mesmo acontece independentemente da unidade de tempo ser o dia ou outra superior. Em contrapartida, na abordagem *next-event* o avanço do Tempo de Simulação é definido pela ocorrência do próximo Evento.

Um simulador ABMS distribuído de uma cadeia de abastecimento é constituído por um conjunto de Agentes que interagem entre si. Cada elemento é modelado por um Agente independente que necessita de trocar mensagens com outros Agentes em determinados intervalos de tempo. Sendo os Agentes entidades computacionais independentes que podem residir no mesmo computador ou numa rede de computadores, é necessário garantir que a troca de mensagens entre os Agentes é feita na mesma unidade de Tempo de Simulação, ou seja, assegurar que a execução dos acontecimentos de simulação segue a ordem temporal. Um simulador deve assegurar que a sequência de acontecimentos ocorre na ordem correta. Por exemplo, o início de preparação da encomenda para expedição, só pode ocorrer depois da receção da encomenda e a expedição da encomenda só pode ocorrer depois da mesma estar pronta para expedição.

Num simulador da cadeia de abastecimento que seja executado num só simulador, os acontecimentos são armazenados numa lista local ordenada, sendo assim garantida a sequência correta de execução dos acontecimentos. Contudo, em Simulação Paralela com vários simuladores executados de forma independente, apesar da sequência de acontecimentos ser garantida dentro do simulador, é igualmente necessário garantir que a ordem dos acontecimentos, em resultado da troca de mensagens entre os simuladores, também é respeitada.

Aliás, mais importante do que assegurar que a troca de mensagens é efetuada na mesma unidade de tempo, é garantir que a sequência de acontecimentos do Modelo de Simulação é executada na ordem que melhor representa a realidade do sistema real. Além disso, como já foi referido, numa cadeia de abastecimentos, há uma grande variedade de elementos envolvidos. Analisando-os, em particular a nível do seu comportamento interno e do seu relacionamento com os outros elementos, podemos encontrar:

- Elementos que recebem mensagens mas só as tratam em períodos pré-definidos. Por exemplo, uma fábrica, com um planeamento semanal, pode receber encomendas ao longo da semana. Neste caso, o planeamento depende das encomendas recebidas até ao momento em que se inicia planeamento, independentemente do dia a que foram recebidas. Do ponto de vista do simulador da fábrica, as encomendas podem ser concentradas imediatamente antes do início do planeamento, desde que seja respeitada a ordem de chegada das mesmas;

- Elementos que só enviam mensagens em períodos pré-definidos. Por exemplo, se o meio de expedição de um armazém é por via marítima ou outro meio de transporte com uma amplitude temporal igualmente elevada em comparação com o restante Sistema, o simulador do armazém pode avançar até ao tempo da próxima expedição. A sequência dos acontecimentos internos, independentes do ambiente, são controlados pelo simulador do armazém, que pode prosseguir a simulação até ao tempo de expedição, e nessa altura aguardar pela sincronização temporal com os restantes simuladores;
- Elementos com intervalos de tempo entre acontecimentos de diferentes amplitudes. Podemos ter na cadeia de abastecimentos, fábricas que tratam as encomendas recebidas ao fim do dia, enquanto outras fábricas só tratam as encomendas semanalmente. Neste caso, cada simulador pode avançar o Tempo de Simulação de acordo como o seu ritmo e posteriormente aguardar pela sincronização ao fim de cada ciclo;
- Elementos com acontecimentos variáveis em frequência e amplitude. Neste caso é necessária uma sincronização mais refinada, entre os simuladores envolvidos, dada a imprevisibilidade dos futuros acontecimentos relacionado com o ambiente.

4.2.1 Tipos de Agentes

De acordo com o modo como os elementos comunicam entre si, os Agentes que os modelam podem ser classificados num dos seguintes tipos:

A - Enviam e recebem mensagens em intervalos de tempo fixos.

B - Enviam mensagens a intervalos de tempo variáveis mas só recebem mensagens em intervalos de tempo fixos.

C - Enviam e recebem mensagens em intervalos de tempo variável.

Ao modelar um Sistema real, como uma cadeia de abastecimentos, cada elemento pode ser modelado como um Agente do tipo A, B ou C. O Modelo global pode ter uma combinação de qualquer tipo de Agentes.

Num simulador ABMS, a sincronização do tempo durante a troca de mensagens é fundamental. Quando um Agente envia uma mensagem, o tempo associado a esta é o Tempo de Simulação associado ao seu relógio interno. O incremento do Tempo de Simulação do Agente recetor da mensagem depende do tipo de Agente recetor. Isto é, se este for:

- Agente tipo A ou B – O incremento do Tempo de Simulação tem de ser igual ou inferior ao intervalo de tempo de receção de mensagens do próprio Agente.

- Agente tipo C – O incremento Tempo de Simulação do Agente recetor tem de ser igual ou inferior ao incremento do Tempo de Simulação do Agente emissor da mensagem.

Note-se que, o Agente tipo A é um caso particular do Agente tipo B. Visto que o problemático na sincronização é a receção da mensagem, qualquer sistema pode ser modelado com Agentes tipo B e/ou C.

4.2.2 Sincronização do Tempo

O caso mais frequente é aquele em que os tempos para a receção de mensagens não estão alinhados durante a Simulação. Vejamos o exemplo: a Figura 15 mostra as trocas de mensagens entre dois Agentes do tipo B. Neste caso, o Agente **i** envia a mensagem **$M_0(i,j)$** para o Agente **j**, e inicia a simulação avançando até ao Tempo de Simulação **t_{i1}**. Ao atingi-lo, o Agente **i** aguarda que o Agente **j** atinja o Tempo de Simulação igual ou superior ao seu.

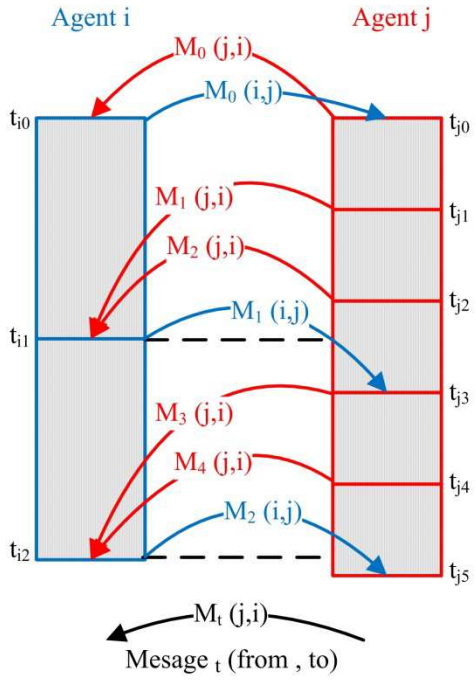


Figura 15 – Modelo com dois Agentes tipo B

Entretanto, o Agente **j** inicia a simulação enviando a mensagem **$M_0(j,i)$** para o Agente **i** e prossegue, enviando mensagens até que o seu Tempo de Simulação seja superior ou igual ao Tempo de Simulação do Agente **i**. Só quando as mensagens **$M_1(j,i)$** e **$M_2(j,i)$** são recebidas pelo

Agente **i,** é que são processadas por este no Tempo de Simulação $\mathbf{t_{i1}}$. Depois do Agente **j** ter ultrapassado o Tempo de Simulação $\mathbf{t_{i1}}$, o Agente **i** envia a mensagem $\mathbf{M_1(i,j)}$ em resposta às mensagens recebidas e continua a simulação até ao tempo $\mathbf{t_{i2}}$, aguardando uma nova sincronização . Quando o Agente **j** atinge o Tempo de Simulação $\mathbf{t_{j3}}$, espera que o Agente **i** envie uma mensagem para poder continuar a simulação até ao tempo $\mathbf{t_{j5}}$. O processo repete-se até ao fim da Simulação.

Todavia, nem sempre é possível criar um Modelo só com Agentes do tipo B. No Modelo da Figura 16, temos um Agente do tipo C e um Agente do tipo B. O Agente **j**, sendo do tipo B, tem um comportamento idêntico aos da figura anterior. Porém, o Agente **i**, como é do tipo C, pode receber e processar mensagens em qualquer unidade de tempo, contudo só pode avançar o Tempo de Simulação em incrementos de tempo iguais aos do Agente **j**. Sendo assim, os Agentes iniciam a Simulação e avançam ambos até ao tempo $\mathbf{t_{j1}}$, posteriormente até ao tempo $\mathbf{t_{j2}}$, e assim por diante. Desta forma, a introdução de Agentes do tipo C aumentará o número de paragens para sincronização entre Agentes, isto porque este tipo de Agentes, mesmo que haja um grande intervalo de tempo entre o envio das várias mensagens, têm de parar para sincronizar a intervalos de tempo iguais aos dos Agentes emissores.

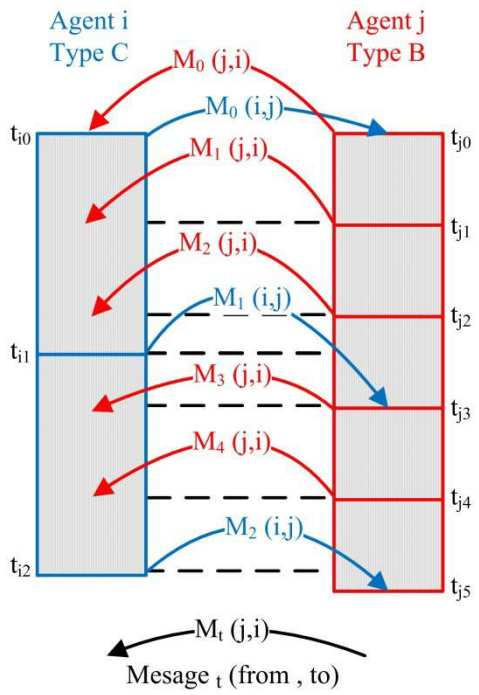


Figura 16 - Modelo com Agentes tipo B e C

Sendo os Agentes independentes executados em processos separados, necessitam de um mecanismo que os informe de qual o Tempo de Simulação dos outros Agentes envolvidos na Simulação. Para tal, é necessário um Agente especial responsável por recolher informação dos intervalos de envio de mensagens de todos os Agentes envolvidos e é esse que informa os Agentes dos Tempos de Simulação de cada um deles.

4.3 Arquitetura da *Framework*

Na história da Simulação, as linguagens de programação utilizadas evoluíram em simultâneo com os meios informáticos. A evolução das capacidades de comunicação entre sistemas e o aparecimento de novas linguagens de programação, permitiu criar simuladores com capacidades impensáveis no início da simulação.

Quase todas as linguagens de programação já foram utilizadas na simulação, tais como TURBOPASCAL, SQPC, C++, SOAR, Z, DYNAMO, e JAVA, bem como algumas linguagens dedicadas (Gilbert e Bankes 2002) (J. M. Teixeira 2006).

Nas últimas décadas, as ferramentas de Simulação mais utilizadas foram desenvolvidas maioritariamente em C++, JAVA e linguagens de programação dedicadas. As vantagens e desvantagens das mais conhecidas como o MASON, NetLogo, Repast, Swarm, podem ser encontradas na literatura (Berryman 2008) (Railsback, Lytinen e Jackson 2006) (Gilbert e Bankes 2002) (Castle e Crooks 2006) (Tobias e Hofmann 2004).

Entre as linguagens generalistas, o C++ tem a grande vantagem da velocidade de processamento. O JAVA, mesmo sendo mais lento, tem o benefício de ser grátis e fácil de encontrar um grande número de bibliotecas do domínio público.

Os Agentes de Simulação do protótipo deste trabalho foram desenvolvidos utilizando a linguagem de programação C#. Como qualquer linguagem, o C# tem vantagens e desvantagens: é mais lento que o C++ mas mais rápido que o JAVA. Trata-se de uma linguagem baseada em objetos que, combinada com o .NET *Framework*, permite criar aplicações com as funcionalidades do C++ e do JAVA. A tecnologia .NET *Framework*, possibilita gerar, de uma

forma simples, aplicações com capacidades de comunicação entre si, recorrendo a bases de dados e *Web Services*.

A *Framework* associada a este trabalho foi desenvolvida com vista em, por um lado, tirar partido da execução distribuída por uma rede de computadores e, por outro lado, permitir o trabalho colaborativo. Cada componente da *Framework* e módulos de apoio, podem ser executados em diferentes computadores ligados em rede, e cada Agente simulador pode ser manipulado por diferentes utilizadores de forma colaborativa.

A Figura 17 representa as diferentes componentes da *Framework*:

- *Environment Agent* (EA): Agente responsável pela gestão das comunicações entre os Agentes e controlo do tempo global de simulação;
- *Simulation Agents* (SA_i): Representa cada um dos Agentes simuladores da *Framework* que modelam o comportamento dos diferentes subsistemas;
- *Web Service* (WS): Módulo utilizado para armazenar e recolher informação durante o processo de simulação;
- *Configuration Data Modeller* (CD): Módulo que cria e altera os diferentes cenários da Simulação.
- *Display Model* (DM): Módulo para visualização da Simulação.
- *Statistical Data* (SD): Módulo para análise estatística dos *outputs* da Simulação.

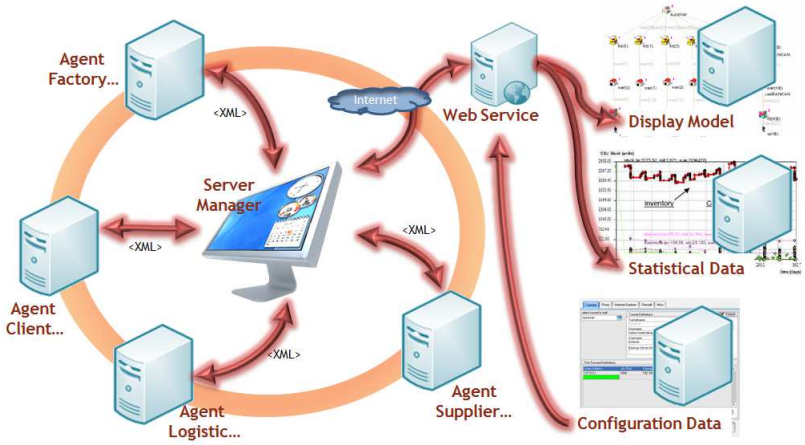


Figura 17 - Arquitetura da *Framework*

O *Environment Agent* (EA), tal como os *Simulation Agents* (SA_i), utiliza a abordagem *Next-event* para o avanço do Tempo de Simulação. Cada SA_i pode avançar no Tempo de

Simulação até ao tempo previsto para a interação com outros $SA_{i.}$ Quando este é atingido, o SA aguarda indicação do EA de que pode avançar a Simulação.

A linguagem de programação utilizada no desenvolvimento do EA e dos SA_i foi o C#, porém a *Framework* permite a utilização de Agentes desenvolvidos em qualquer outra linguagem desde que suporte TCP/IP para troca de mensagens XML com o EA.

A Figura 18 mostra, de uma forma simplificada, o fluxograma da interação entre *Environment Agent* e os *Simulation Agents*. Depois da configuração inicial, gerada com o módulo de configuração, o EA inicia a execução e aguarda pela ligação dos SA. Se esta ligação é aceite pelo EA, os SA enviam para o EA uma mensagem com as suas informações que podem vir a ser necessárias durante a Simulação. Depois desta sincronização inicial, o EA envia uma mensagem para todos os SA com informação do Tempo de Simulação atual e o tempo do próximo evento de sincronização. Em função do tipo de Agente, cada SA inicia a simulação e avança no Tempo de Simulação até ao tempo de sincronização do EA ou até ao Tempo de Simulação do seu próximo evento de sincronização.

O EA trata as mensagens de sincronização com um mecanismo idêntico ao dos simuladores por Eventos Discretos. As mensagens recebidas são guardadas numa lista de acontecimentos ordenadas em função do seu Tempo de Execução. Note-e que as mensagens a enviar são armazenadas na mesma lista de acontecimentos.

A lista de acontecimentos dos SA pode conter três tipos de acontecimentos:

- Acontecimentos internos - são resultantes do Modelo de simulação do Agente;
- Acontecimentos externos - tipicamente são mensagens a enviar a outros SA, ou mensagens recebidas de outros SA;
- Acontecimentos de sincronização - tratam-se de mensagens recebidas do EA com a informação dos próximos acontecimentos de sincronização ou então mensagens a enviar ao EA com o Tempo de Simulação do próximo acontecimento de sincronização dos SA.

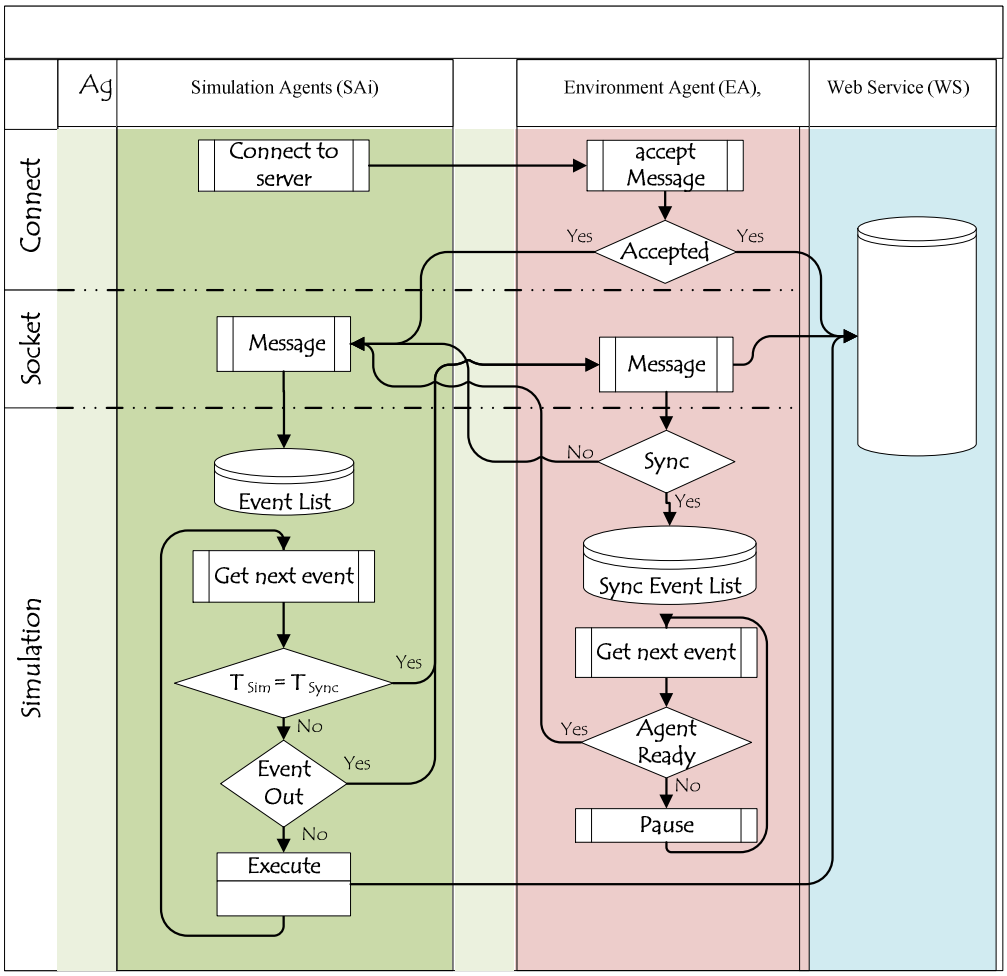


Figura 18 - Executivo do *Environment Agent* e *Simulation Agents*

Durante o processo de simulação dos SA, os acontecimentos são executados respeitando a ordem temporal dos acontecimentos da lista de acontecimentos. Relativamente ao comportamento do simulador, este depende do tipo de acontecimento a executar, podendo ser:

- Acontecimento interno - o SA executa o acontecimento e retira o próximo acontecimento da lista;
- Acontecimento externo - se é uma mensagem a enviar de um SA para outro SA, o primeiro envia a mensagem ao EA que trata de a reencaminhar ao destinatário final. Se é uma mensagem recebida de outro SA, o acontecimento é tratado como acontecimento interno;
- Acontecimento de sincronização - tanto pode ser uma mensagem recebida do EA, como uma mensagem de sincronização gerada pelo próprio SA. Em ambos os casos, o SA envia uma mensagem ao EA indicando o seu Tempo de Simulação e aguarda uma mensagem deste último para poder continuar a Simulação.

O Agente de Simulação desenvolvido pode ser configurado de modo a modelar os vários elementos de uma cadeia de abastecimentos, tais como, uma fábrica, um armazém, um distribuidor, etc. O configurador do Agente, define tanto os serviços que disponibiliza como os que necessita para o seu funcionamento.

A fim de simplificar a integração de componentes de negócios, foi desenvolvida uma biblioteca, de acordo com a recomendação OASIS, com o objetivo de suportar a troca de documentos do processo de negócio. Esta biblioteca inclui módulos para gerar catálogos, encomendas, confirmação de expedição dos pedidos, faturas etc. Os documentos de negócio são trocados ou entre um fornecedor e um recetor, ou entre um comprador e um vendedor.

Além das tradicionais filas existentes num simulador tradicional, o Agente desenvolvido tem duas filas (*OutQueue* e *InQueue*) onde são armazenadas as mensagens trocadas entre os simuladores. As mensagens armazenadas na fila *OutQueue*, são enviadas para o *Environment Agent*, que por sua vez as reencaminha para o Agente de destino. Quando uma mensagem é recebida, esta é armazenada na fila *InQueue* para posterior tratamento.

A Figura 19 apresenta a troca de mensagens *"Catalogue Request", "Application Response"* e "*Catalogue*" associados ao processo de negócio "*Catalogue*" . O Agente *"Buyer"* gera o pedido de catálogo que é armazenado na lista *"OutQueue"*. Quando o simulador atinge o Tempo de Simulação de envio do pedido de catálogo, o Agente envia a mensagem para o EA que a reencaminha para o Agente de destino. Este último, que se trata do ,*"Seller"*, recebe a mensagem e armazena-a na "*InQueue*" até ao seu processamento. Depois de gerado o catálogo o Agente *"Seller"* envia a resposta para a sua *"OutQueue"* de forma a que possa ser reencaminhada ao *"Buyer"* através do EA.

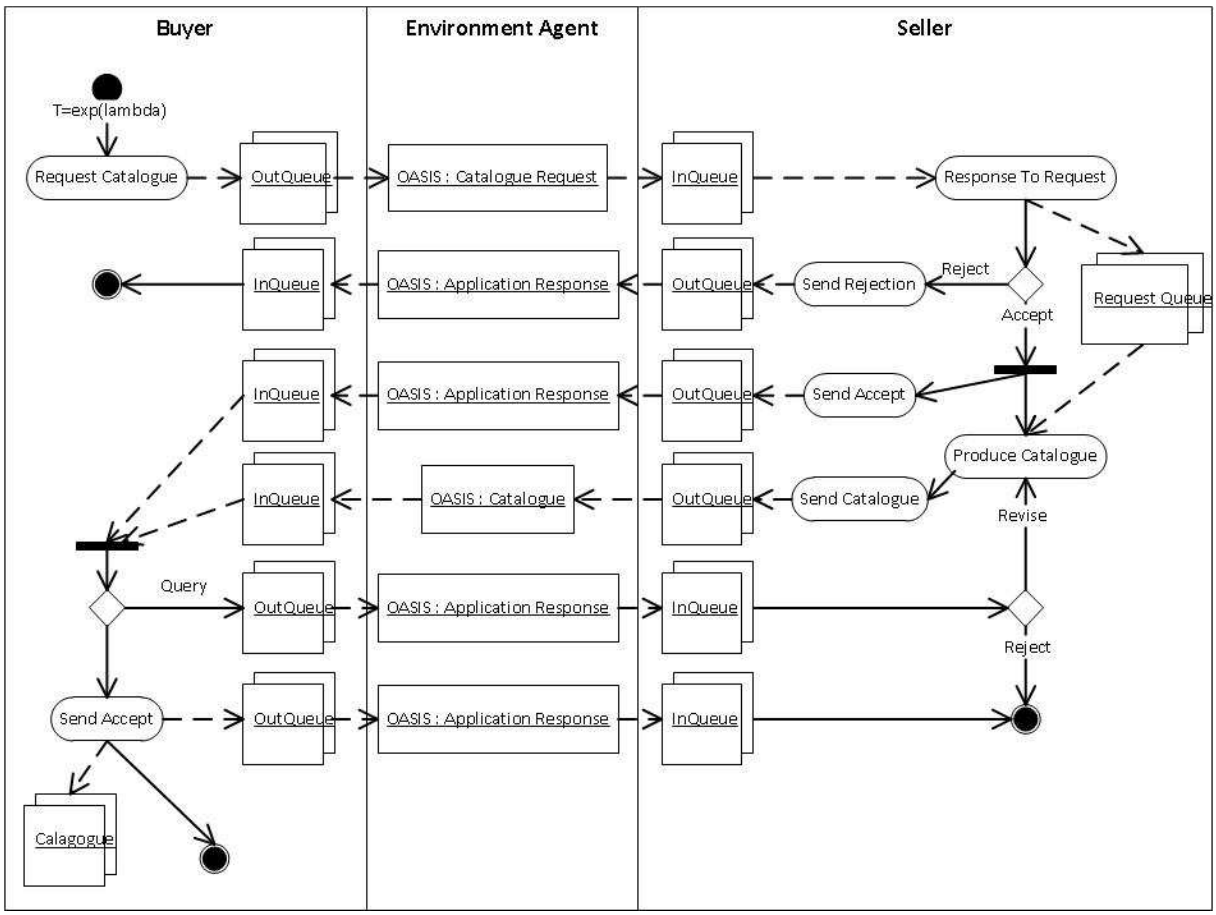


Figura 19 - Catálogo adaptado de (OASIS 2011)

As outras mensagens do processo de negócio, definido pelo OASIS, são tratadas pelos Agentes de modo idêntico ao do processo *"Catalogue"*. Isto é, as mensagens geradas por um Agente em que o destinatário é um outro Agente, são armazenadas na *"OutQueue"* e as mensagens recebidas são armazenadas na "*InQueue"* antes de serem processadas pelo Agente. Note-se que cada Agente tem um mecanismo de envio de mensagens (*"OutQueue"*) para o EA. Por sua vez, quando o EA recebe uma mensagem em que o destino é um Agente, reencaminha-a para este último, que por sua vez a armazena na fila "*InQueue"*.

Capítulo 5.

Desenvolvimento da *Framework*

Neste capítulo apresentam-se os aspetos mais relevantes da *Framework* desenvolvida neste trabalho. Na primeira secção é apresentada a sua arquitetura global. Na segunda, é exposta a arquitetura do *Environment Agent*. Em seguida, na terceira secção, é apresentada a arquitetura do Agente. O Modelo de comunicações é descrito na quarta secção, sendo que na última secção é apresentada a estrutura da linguagem de negócio desenvolvida.

5.1 Arquitetura da *Framework*

A *Framework* foi desenvolvida em C#, de uma forma modular, de modo a facilitar o desenvolvimento de novos Agentes e a integração de simuladores existentes. Aliás, a utilização da linguagem C# permite a criação de bibliotecas DLL (*dynamic-link library*), que podem ser partilhadas com Agentes ou simuladores existentes, mesmo que estes estejam desenvolvidos noutras linguagens compatíveis, tais como o *Microsoft* C++ ou *Microsoft Visual Basic*.

Na Figura 20 estão representadas as bibliotecas mais relevantes da *Framework* relacionadas com o *Environment Agent* (EA) e os *Simulation Agents* (SA). Embora algumas das bibliotecas DLL sejam partilhadas pelo EA e os SA, outras são utilizadas exclusivamente pelo EA ou pelos SA (Tabela 12). Para simplificar o diagrama, as dependências de algumas das bibliotecas não estão representadas, por serem de utilização generalizada, como é o caso das bibliotecas *Distributions.dll*, *SerializeToFromString.dll* e *XmlTreeView.dll*.

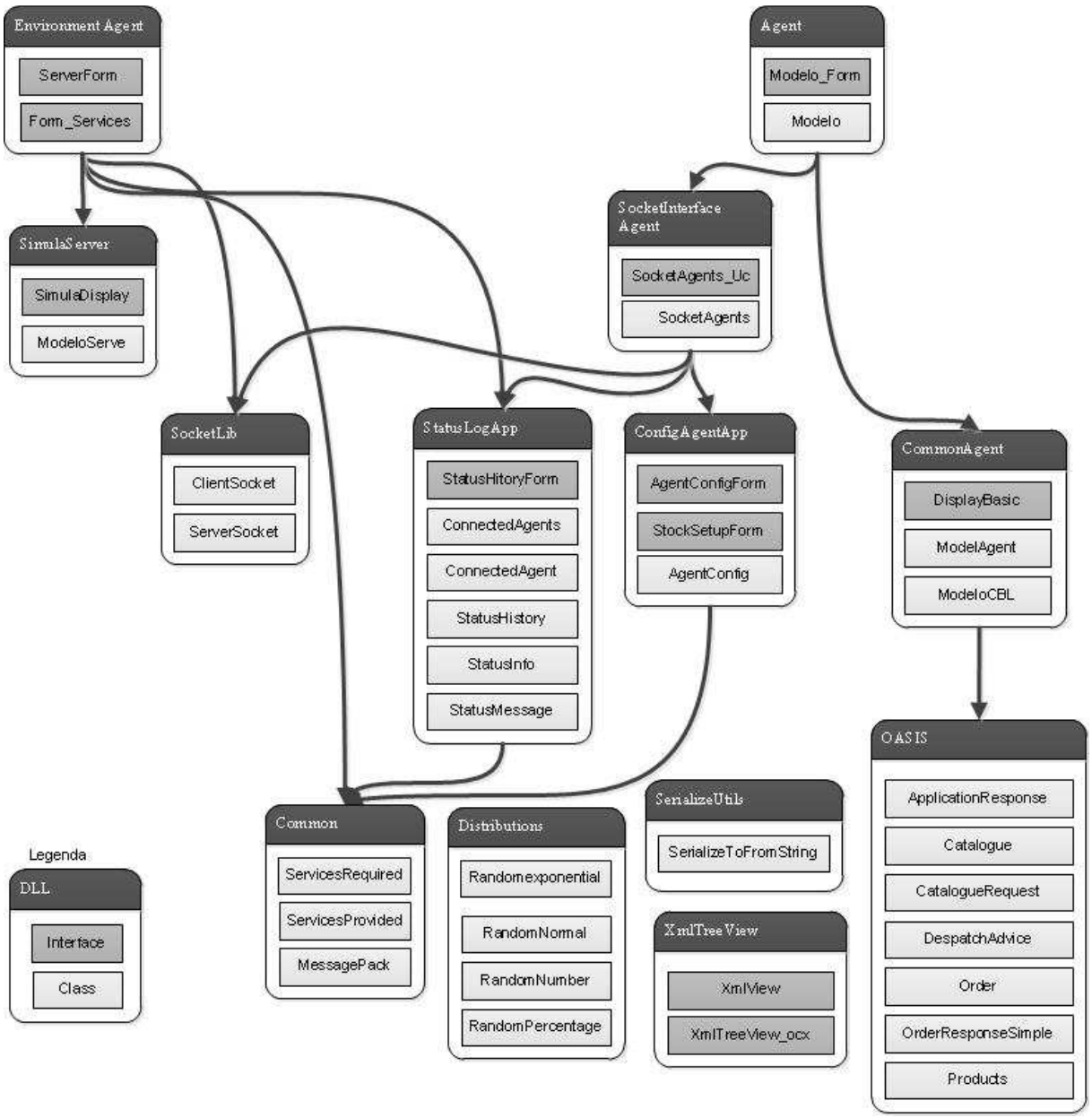


Figura 20 - Bibliotecas (DLLs) da *Framework*

5.1.1 Bibliotecas

As bibliotecas desenvolvidas, conforme a sua utilização, podem ser classificadas como:

- Bibliotecas genéricas - bibliotecas que são de utilização indiferente;
- Bibliotecas EA - utilizadas exclusivamente pelo módulo EA;
- Bibliotecas SA - utilizadas exclusivamente pelo módulo SA;

- Bibliotecas mistas – são partilhadas pelos módulos EA e SA.

Tabela 12 - Funcionalidades das principais bibliotecas

Biblioteca	Domínio	Funcionalidades
SocketLib.dll	EA,SA	Responsável pelas comunicações entre os Agentes e o *Environment Agent* (EA)
Common.dll	EA,SA	Lista dos serviços disponibilizados pelos Agentes, bem como a lista dos serviços que os Agentes necessitam.
Distributions.dll	EA,SA	Contém um conjunto de funções para a geração de distribuições estatísticas.
SerializeToFromString.dll	EA,SA	Possuí um conjunto de ferramentas para serializar classes.
XmlTreeView.dll	EA,SA	Contém um conjunto de interfaces para visualizar ficheiros ou *strings* XML.
SimulaServer.dll	EA	Modelo, executivo e interface do *Environment Agent* (EA)
SocketInterfaceAgent.dll	SA	Interface entre o Agente e o *SocketLib*
ConfigAgentApp.dll	SA	Módulo de configuração do Agente
CommonAgent.dll	SA	Modelo do Agente e da interface para a linguagem de negócio
OASIS.dll	SA	Classes de acordo com o *standard* OASIS *Universal Business Language*

Em seguida, são descritas sinteticamente as funcionalidades das principais classes existentes em cada uma das bibliotecas.

5.1.1.1 Bibliotecas Genéricas

As bibliotecas (Dll) classificadas como genéricas, são bibliotecas que tanto podem ser utilizadas na presente *Framework* como em qualquer outro projeto.

- ***Distributions.dll*** - Conjunto de classes com métodos para a geração de distribuições estatísticas.

Tabela 13 - Dll *Distributions*

Classe	Propriedades	Funcionalidade
Randomexponential	*Lambda, Seed*	Gera uma sequência de valores de acordo com uma distribuição exponencial
RandomNormal	*Average, StDeviation, Seed*	Gera uma sequência de valores de acordo com uma distribuição normal
RandomNumber	*minValue, maxValue, Seed*	Gera valores aleatórios entre dois limites
RandomPercentage	*PercOK, Seed,*	Gera uma sequência de varáveis booleanas com percentagem de verdadeiro pré-definida

- ***SerializeToFromString.dll*** - Conjunto de classes com métodos para serializar objetos em forma de sequência de caracteres ou ficheiro XML.

Tabela 14 - Dll *SerializeToFromString*

Classe	Propriedades	Funcionalidade
ToXml	*Obj, ObjType, xmlSerializerNamespaces*	Serializa um objeto numa sequência de caracteres
ToXmlFile	*filename, Obj, ObjType, xmlSerializerNamespaces*	Serializa um objeto e grava num ficheiro XML

- ***XmlTreeView.dll*** - Formulário e controle *ActiveX* (ocx) para visualizar ficheiros XML.

Tabela 15 - Dll *XmlTreeView*

Classe	Propriedades	Funcionalidade
XmlTreeView_ocx	*XMLString*	Representa em forma de árvore uma sequência de caracteres ou ficheiro XML
XmlView	*XMLString(), Ncolumn, NRow*	Representa N objetos tipo *XmlTreeView_ocx*

5.1.1.2 Bibliotecas EA

As bibliotecas (Dll) classificadas como EA contêm um conjunto de classes relacionadas com o simulador EA. As classes de Simulação destas bibliotecas foram separadas da interface com o utilizador para permitir criar interfaces com o utilizador noutras linguagens de programação.

- ***SimulaServer.dll*** – Simulador e módulo de visualização

Tabela 16 - Dll *SimulaServer*

Classe	Propriedades	Funcionalidade
ModeloServer	Modelo	Módulo de simulação
SimulaDisplay	Modelo	Interface de visualização da simulação

5.1.1.3 Bibliotecas SA

- ***SocketInterfaceAgent.dll*** – Interface gráfica entre o *Socket* e os Agentes. A Dll tem um conjunto de processos responsáveis pelas comunicações entre Agentes que facilitam o desenvolvimento de novos Agentes.

Tabela 17 - Dll *SocketInterfaceAgent*

Classe	Propriedades	Funcionalidade
SocketAgents	*agentConfig*	Conjunto de módulos para gerar e enviar mensagens para o EA ou outros Agentes. A classe cria o envelope da mensagem e envia ao destinatário As mensagens recebidas são descodificadas e enviadas ao Agente com recurso a *Events*
SocketAgents_Uc	*SocketAgents*	Interface de visualização das comunicações

- ***ConfigAgentApp.dll*** – Classe com informação sobre a configuração do Agente e suas interfaces gráficas, para permitir a sua manipulação.

Tabela 18 - Dll *ConfigAgentApp*

Classe	Propriedades	Funcionalidade
AgentConfig	*FileName*	Classe com toda a informação sobre o Agente
AgentConfigForm	*AgentConfig*	Interface gráfica para manipular a informação da classe *AgentConfig*
StockSetupForm	*FileName*	Interface gráfica para manipular a informação do *stock* inicial de cada Agente

- ***CommonAgent.dll*** – Biblioteca com um conjunto de classes comuns a vários Agentes

Tabela 19 - Dll *CommonAgent*

Classe	Propriedades	Funcionalidade
ModelAgent	Modelo	Motor de simulação dos Agentes
ModeloCBL	Modelo	Contém um conjunto de classes de interface entre o Agente e a biblioteca OASIS. Transforma a informação do Agente de acordo com o *standard* OASIS
DisplayBasic	Modelo	Interface gráfica para visualização simplificada do Modelo durante a Simulação

- ***OASIS.dll*** – Biblioteca com um conjunto de classes criadas de acordo com a especificação OASIS UBL-2.1

Tabela 20 - Dll OASIS

Classe	Propriedades	Funcionalidade
Classes UBL		Classes para criar XML de acordo com a especificação OASIS UBL-2.1 (OASIS UBL 2011)
Classes GC		Classes para criar Lista de Códigos (*Genericode*) de acordo com a especificação (OASIS 2007)

5.1.1.4 Bibliotecas EA/SA

- ***SocketLib.dll*** – Responsável pelas comunicações entre os Agentes e o *Environment Agent* (EA)

Tabela 21 - Dll *SocketLib*

Classe	Propriedades	Funcionalidade
ClientSocket	*ServerhostNameOrAddress, ServerPort*	Conjunto de classes responsáveis por estabelecer a ligação com o *socket* do EA bem como por enviar e receber mensagens
ServerSocket	*ServerPort*	Conjunto de classes responsáveis por aceitar a ligação do *socket* dos Agentes e por receber e enviar mensagens

- ***Common.dll*** – Conjunto de classes utilizadas para serializar e desserializar tanto as mensagens como a lista de serviços disponibilizados e necessários aos Agentes.

Tabela 22 - Dll *Common*

Classe	Propriedades	Funcionalidade
MessagePack	Mensagem	Classes para serializar e desserializar as mensagens em função do seu tipo
Services	Seviços	Lista dos serviços necessários e disponibilizados pelos Agentes

5.2 *Environment Agent*

O *Environment Agent* (EA) é o módulo da *Framework* responsável pelo controlo do Tempo de Simulação e pela gestão das comunicações entre os Agentes. Sendo assim, o EA tem permanentemente atualizada a informação dos Agentes envolvidos na Simulação, isto é, conhece sempre:

- qual a identificação de cada Agente;
- o endereço de cada Agente para o envio das mensagens;
- os serviços que disponibiliza;
- o Tempo de Simulação de cada Agente;
- o tempo da próxima sincronização dos Agentes;
- as mensagens trocadas entre os Agentes.

O EA tem duas interfaces gráficas distintas (Figura 21): uma para controlo e visualização da Simulação, e outra para visualizar os serviços disponibilizados pelos Agentes envolvidos.

O primeiro programa da *Framework* a iniciar a execução tem de ser obrigatoriamente o EA, pois é este o programa responsável pelas comunicações entre os Agentes. Depois de iniciado, a interface gráfica indica o endereço para o qual os Agentes terão de efetuar a ligação.

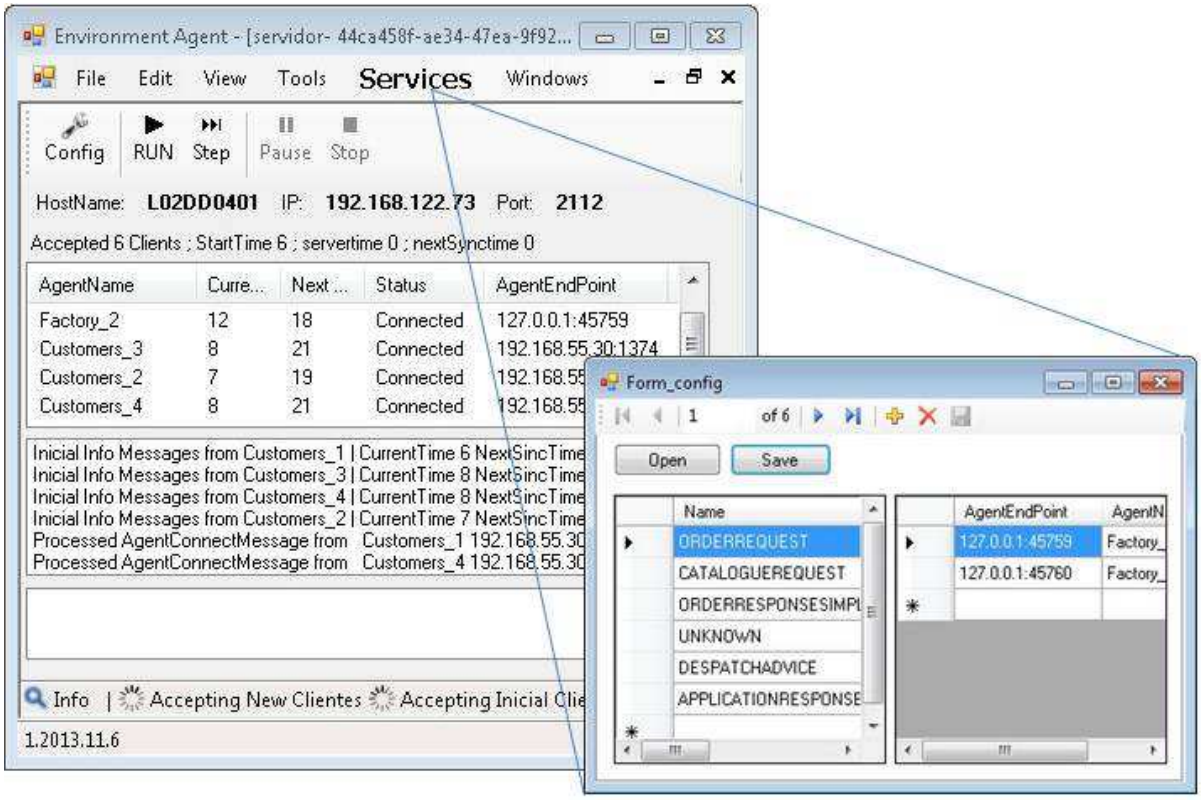


Figura 21 – EA interfaces gráficas

Depois do arranque, o EA fica em modo de espera até que os Agentes façam os pedidos de ligação, estabelecendo então um canal de comunicações independente com cada um dos Agentes. Depois de esta ser aceite, os Agentes enviam uma informação inicial que inclui os serviços que disponibilizam.

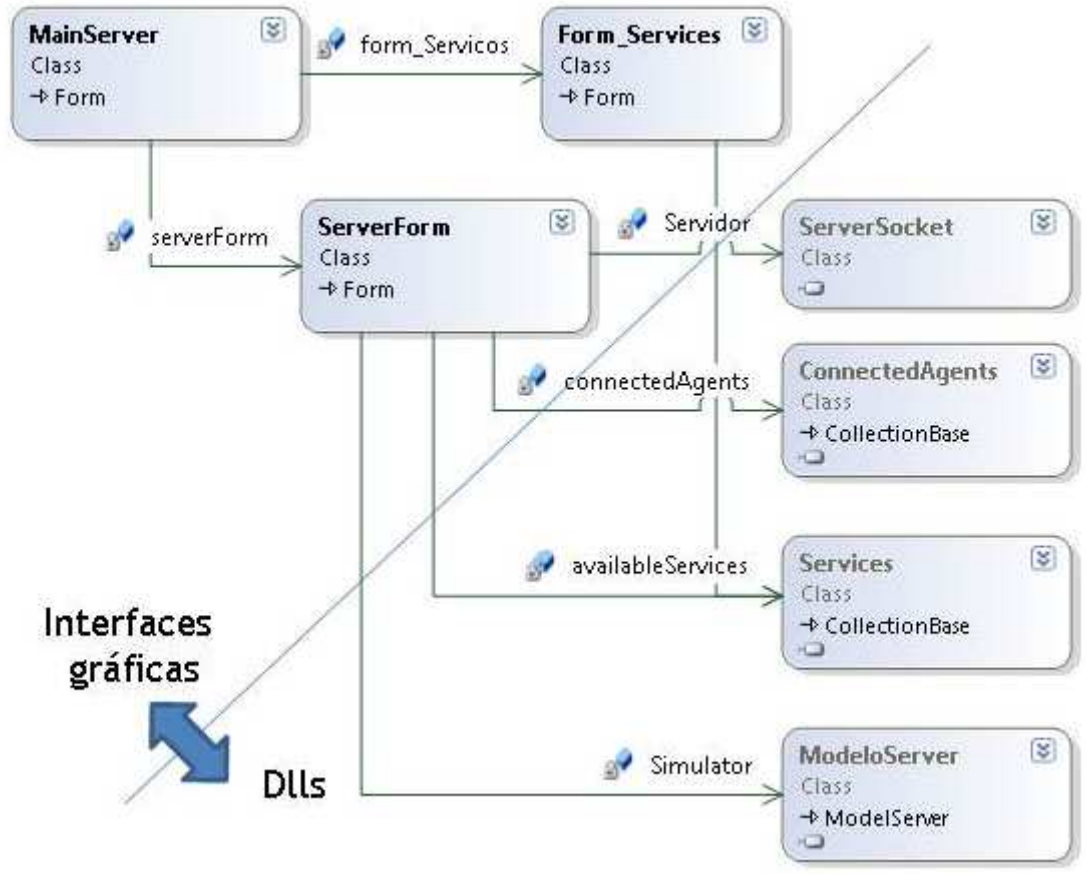


Figura 22 - Diagrama do *Environment Agent*

As interfaces gráficas estão separadas do motor de simulação bem como das bibliotecas referidas no ponto 5.1.1. Assim, caso seja necessário, é possível o desenvolvimento da interface numa outra linguagem de programação compatível ou mesmo a elaboração de uma interface mais elaborada de uma forma independente.

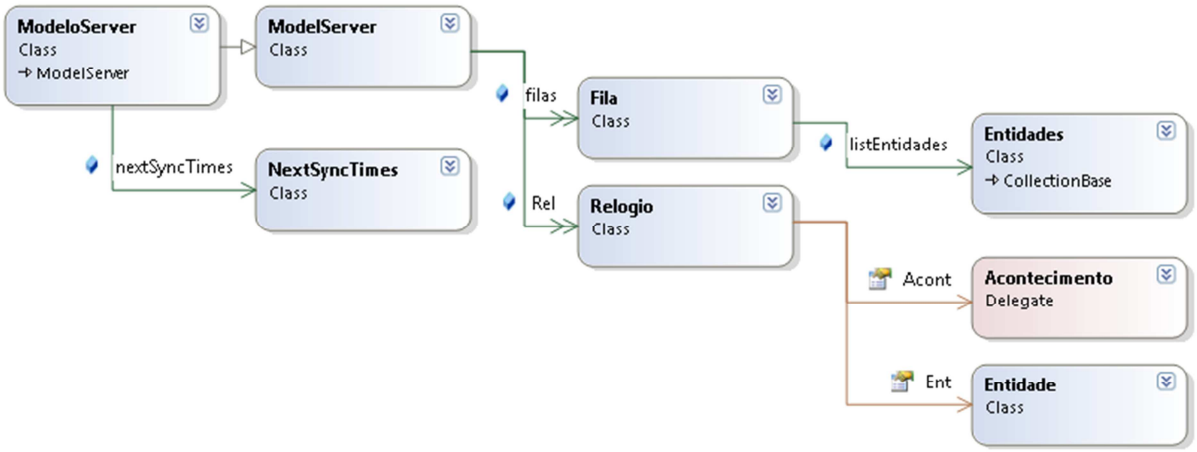


Figura 23 – Diagrama do simulador do *Environment Agent*

O EA tem um simulador *Next-event* tradicional (Figura 23), em que as entidades definem o momento de envio de mensagens de sincronização com os Agentes.

5.3 Arquitetura do Agente (SA)

Para facilitar o desenvolvimento de diferentes Agentes, cada um com características particulares, foi desenvolvido um Agente SA base (Figura 24), de uma forma modular, que recorre a um conjunto de módulos elementares comuns a todos os Agentes (ver ponto 5.1.1).

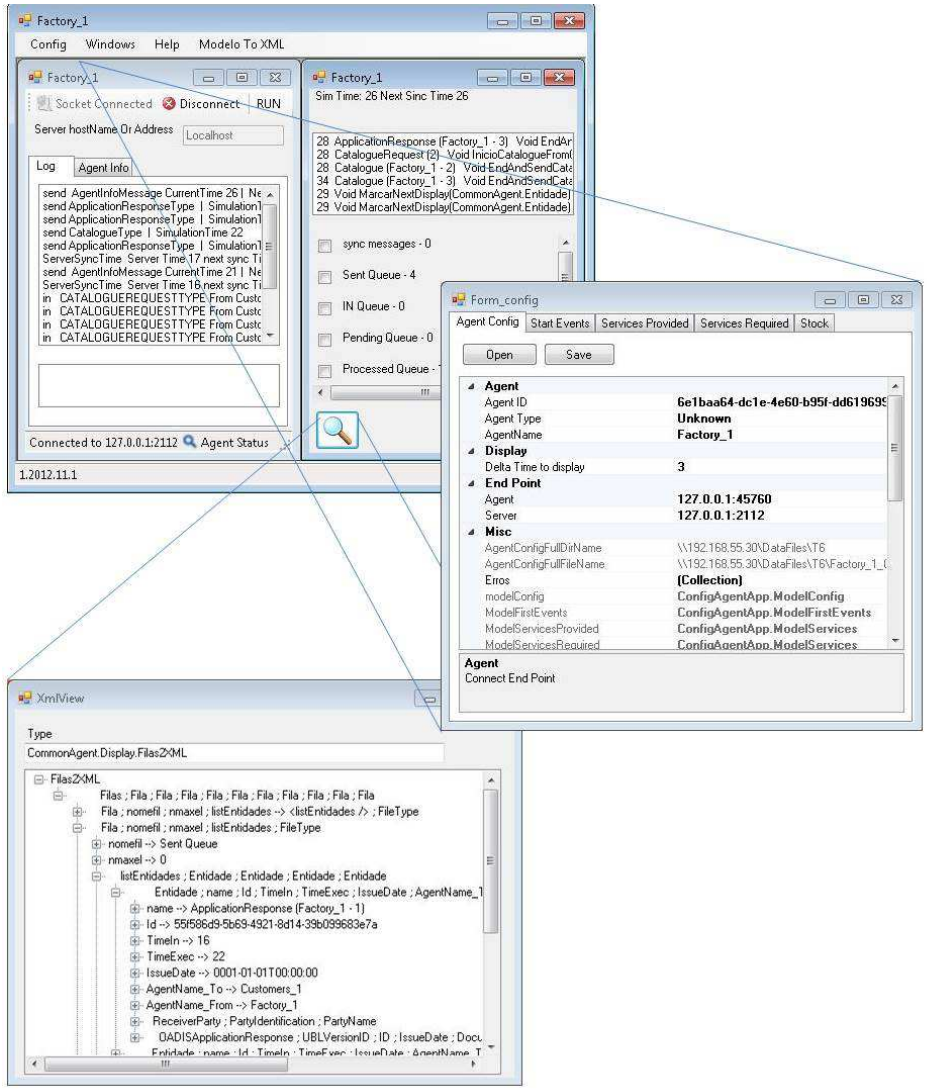


Figura 24 - Interface do *Simulation Agent*

Tal como o EA, no Agente SA Base desenvolvido, as interfaces gráficas estão separadas das restantes bibliotecas. Esta separação da interface gráfica do Modelo de Simulação permite criar novos Agentes com Modelos diferentes, utilizando a mesma interface gráfica. Por outro lado, qualquer melhoria ou alteração da interface gráfica é refletida automaticamente em todos os Agentes já desenvolvidos.

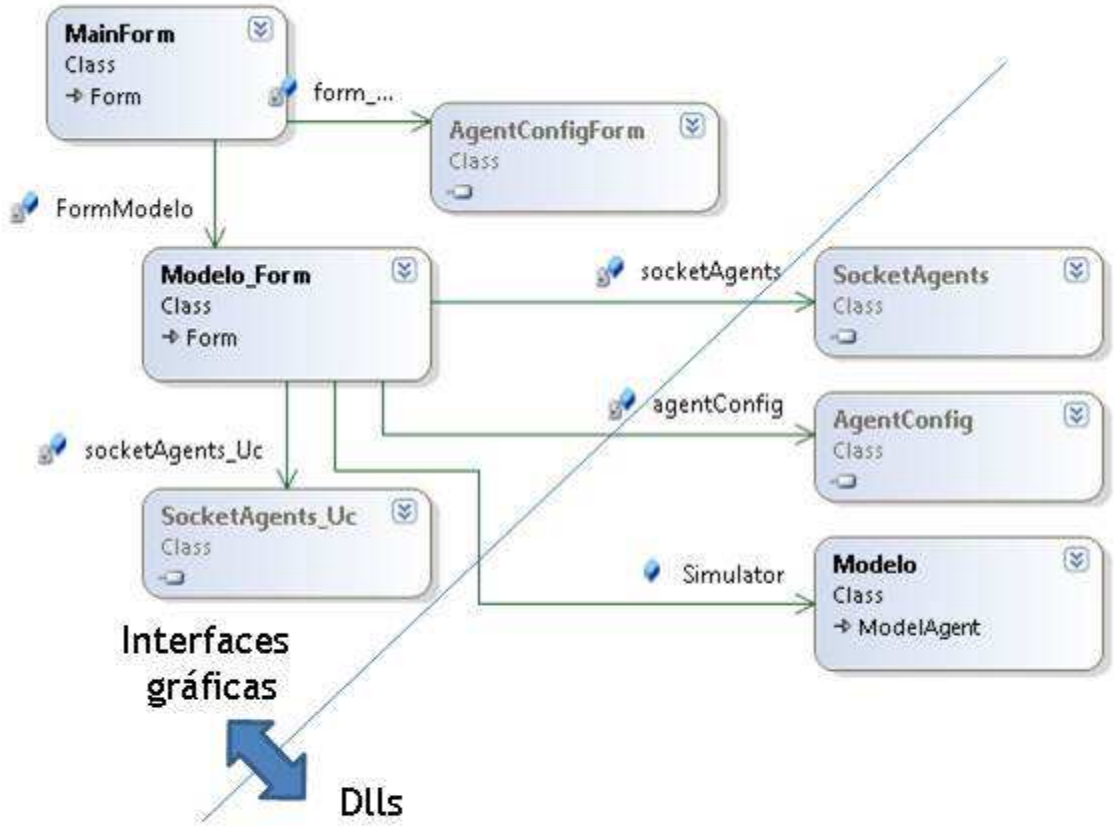


Figura 25 - Diagrama do *Simulation Agent*

O Agente SA Base desenvolvido pode desempenhar qualquer papel dentro da cadeia de abastecimento, dependendo do ficheiro de configuração associado. Note-se que o ficheiro de configuração do Agente é um ficheiro XML, gerado pelo módulo de configuração.

5.4 Comunicações

A estrutura das mensagens trocadas entre Agentes bem como entre estes e o EA é uma estrutura de dois níveis em formato XML (Tabela 23), ou seja, tem uma camada exterior ou envelope e o corpo da mensagem. O envelope da mensagem contém, por um lado, informação sobre a origem e destino da mensagem e, por outro lado, um conjunto de outras informações genéricas.

O corpo da mensagem pode ter qualquer tipo de informação, desde informação sobre o Agente, o Tempo de Simulação ou pode mesmo conter uma mensagem OASIS UBL. A Tabela 23 apresenta os principais parâmetros da estrutura das mensagens.

Tabela 23 - *MessagePackType*

```
<MessagePack ….>
   <messagePackType>...…</messagePackType>  - Tipo de mensagem
   <AgentName_From>……</AgentName_From>  - Agente que envia a mensagem
   <AgentName_To>……</AgentName_To> - Agente destino da mensagem
   <AgentCurrentTime>……</AgentCurrentTime> - Tempo corrente do simulador
   <MessageObs>……</MessageObs>   - Campo para observações
   <MessageBodyType>……</MessageBodyType>   - Tipo do corpo da mensagem
   <MessageBody>- Corpo da mensagem
      …….
   </MessageBody>
</MessagePack>
```

Todos os Agentes necessitam de estabelecer um canal de comunicação com o EA, de forma a enviar e receber mensagens. Aliás, mesmo as trocas de mensagens entre Agentes são efetuadas através do EA. Se no envelope da mensagem recebida pelo EA está um Agente que se encontra ligado ao EA, este reencaminha a mensagem para o Agente destino. Porém, se o EA desconhece o Agente destino, então envia uma mensagem para o remetente indicando que não conhece o destinatário.

O processo de troca de mensagens (ver exemplo de mensagens no Anexo C) segue os seguintes passos:

A. Ligação dos Agentes
    1. O EA inicia a execução e aguarda mensagens de pedido de ligação dos Agentes;
    2. Recebido o pedido de ligação, o EA estabelece um túnel de comunicações independente com cada Agente;
    3. Os Agentes enviam mensagens do tipo *AgentConnectMessage* com informação do seu estado e quais os serviços que disponibilizam;
    4. O EA armazena a informação da mensagem *AgentConnectMessage* e envia a mensagem *ServerConfirmConnect* de confirmação da ligação;
    5. O Agente envia a mensagem *AgentInicialInfoMessage* com informação do seu Tempo de Simulação corrente.

B. Início da Simulação
    1. Quando o utilizador inicia a Simulação, o EA envia uma mensagem *ServerStart* a todos os Agentes.

C. Serviços
    1. Quando um Agente tem de enviar um pedido a outro Agente, se não tiver a informação de quais os Agentes que disponibilizam o serviço que necessita, então envia a mensagem *AgentFindService* para o EA. Este responde com uma mensagem que contém a lista dos Agentes que fornecem o serviço pedido.
    2. Por sua vez, quando o EA recebe uma mensagem *AgentFindService*, gera a mensagem *ServerServiceProvider* com informação de quais os Agentes que disponibilizam o serviço pedido.

D. Sincronização
    1. A sincronização do Tempo de Simulação entre os Agentes é conseguida pela troca de mensagem *AgentInfoMessage* - enviada pelos Agentes - e a mensagem *ServerSyncTime* - enviada pelo EA a todos os Agentes.

E. Mensagens OASIS UBL
    1. A sequência e o número de mensagens de negócio do tipo OASIS UBL variam de acordo com o processo. No exemplo de um pedido de catálogo, o Agente envia uma mensagem *CatalogueRequestType* aos Agentes que disponibilizam o serviço e aguarda as mensagens *ApplicationResponseType* e *CatalogueType*. Depois de receber o catálogo, inicia o processo de pedido de encomenda de acordo com o Modelo definido.

As mensagens dos grupos A e B, são trocadas no início da Simulação. Os pedidos de serviços - grupo C - são efetuados quando necessário. No início da Simulação, cada Agente tem, por um lado, uma lista de serviços que necessita para o seu funcionamento, por outro, o conjunto de serviços que precisará de saber quem são os fornecedores que os facultam. Sendo assim, quando a Simulação começa, cada Agente envia todos os seus pedidos. Contudo, durante a Simulação pode necessitar de outros serviços, e neste caso efetua o(s) novo(s) pedido(s) já com a Simulação em andamento. As mensagens do tipo D e E, são trocadas durante todo o período da Simulação.

5.5 Linguagem de Negócio

As mensagens de negócio trocadas entre os Agentes respeitam as especificações OASIS UBL. A Figura 26 apresenta de forma simplificada o diagrama de classes do Modelo do Agente SA base desenvolvido. O **Modelo** herda a classe ***ModelAgent,*** que inclui as classes ***AgentConfig*** e **ModeloCBL**. A classe ***AgentConfig*** contém toda a informação sobre a configuração do Agente, incluindo os serviços que disponibiliza e necessita, informação sobre *stocks*, etc. A classe **ModeloCBL** contém um conjunto de variáveis e procedimentos tanto para criar os acontecimentos a adicionar na lista do simulador como para tratar as mensagens recebidas de outros Agentes.

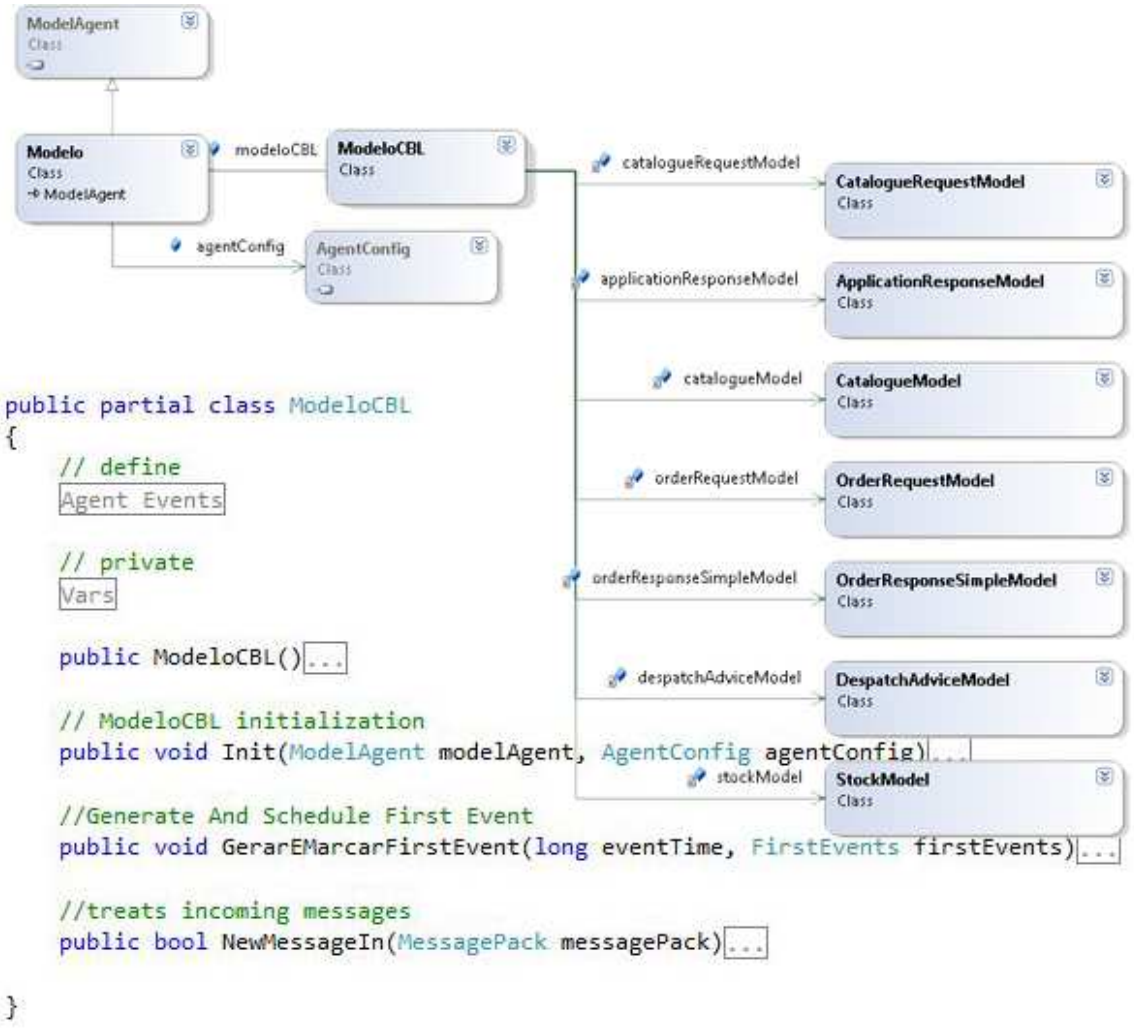


Figura 26 - Diagrama de classes do ModeloCBL

O método **Init,** da classe **ModeloCBL,** inicializa o Modelo em função da informação contida no ficheiro de configuração.

Os métodos para criar ou tratar cada uma das mensagens de negócio foram agrupados em classes, de acordo com o tipo de mensagem. A passagem de informação entre as classes é efetuada com recurso ao envio de *Events* que são tratados de acordo com o que está definido no método **Init** da classe **ModeloCBL.** Os principais métodos e respetivas funcionalidades estão resumidos da Tabela 24 à Tabela 29.

Tabela 24 - *CatalogueRequestModel*

Método	**Funcionalidade**
NewCataloguesRequests2AllFornec	Se tem informação dos Agentes que aceitam pedidos de catálogos: - Marca um acontecimento "Criar pedido de catálogo" para cada fornecedor do serviço Se não tem: - Envia pedido ao EA - Adiciona pedido de catálogo à lista de pendentes
MarcarCreateCatalogueRequest	Marca o acontecimento "Criar pedido de catálogo"
MarcarSendNewCatalogueRequest	Marca o acontecimento "Enviar pedido de catálogo"
SendCatalogueRequest	Adiciona pedido de catálogo à fila *AgentOUTQueue* e envia pedido de catálogo
MarcarTreat ApplicationResponse To CatalogueRequest	Marca o processamento da *ApplicationResponse*
InicioTreat ApplicationResponse To CatalogueRequest	Se a resposta for aceite: - Move pedido de catálogo para lista de aceites Caso contrário: - Move pedido de catálogo para lista de recusados
INCatalogueRequest	Recebe pedido de catálogo e marca início da resposta

Tabela 25 - *ApplicationResponseModel*

Método	**Funcionalidade**
Marcar *Start ApplicationResponse To CatalogueRequest*	Marca o acontecimento "*Criar ApplicationResponse To CatalogueRequest"*
Start ApplicationResponse To CatalogueRequest	Envia *ApplicationResponse* Se for aceite: - Marca início para criar catálogo
INApplicationResponse	Marca o início do processamento da *ApplicationResponse*

Tabela 26 - *CatalogueModel*

Método	Funcionalidade
MarcarStartCatalog2CatalogueRequest	Marca o acontecimento "Criar *InicioCatalogueFromCatalogueRequest”*
InicioCatalogueFromCatalogueRequest	Inicia criação do catálogo e marca envio
INCatalogue	Marca o acontecimento "Processar Catálogo"
StartTreatINCatalogue	Trata a informação do catálogo e marca o acontecimento *NextCataloguesRequests2Fornec*

Tabela 27 - *OrderRequestModel*

Método	Funcionalidade
MarcarCreateOrderRequest	Marca o acontecimento *CreateOrderRequest*
CreateOrderRequest	Gera encomenda e Marca o acontecimento *SendOrderRequest*
InicioTreatOrderResponseSimple	Move encomenda para a fila aceite ou recusada
InicioTreatINDespatchAdvice	Atualiza *stocks*
CreateOrderByStockMin	Gera encomendas para repor *stock*
INOrderRequest	*MarcarStartOrderResponseSimpleToOrderRequest*

Tabela 28 - *OrderResponseSimpleModel*

Método	Funcionalidade
StartOrderResponseSimpleToOrderRequest	*GerarOrderResponseSimple* Marca acontecimento de envio Se aceite: - Marca acontecimento *MarcarStartDespatchAdviceToOrderRequest*
INOrderResponseSimple	*MarcarTreatOrderResponseSimple*

Tabela 29 - *DespatchAdviceModel*

Método	Funcionalidade
StartDespatchAdviceToOrderRequest	*GerarDespatchAdvice* aceite ou recusada Marca acontecimento de envio

Capítulo 6.

Validação e Avaliação da *Framework*

Neste capítulo apresentam-se alguns dos testes efetuados com a *Framework* desenvolvida neste trabalho. Os testes realizados têm como objetivo avaliar a *Framework* e analisar o seu comportamento em diferentes configurações.

6.1 Introdução

Para a validação da *Framework* foram efetuados um conjunto de testes com vista a validar o mecanismo de comunicação e a consistência dos resultados em diferentes configurações. Em todos os testes foram comparados os resultados das simulações monoposto e multiposto.

Nas simulações multiposto foram utilizados até dez computadores com as mesmas características e comparados os resultados com um número variado de computadores. Como seria de esperar, foram obtidos sempre os mesmos resultados em tempos de execução diferentes de acordo com o número de computadores utilizados.

Para validar o comportamento da *Framework* em ambientes heterogéneos, foram introduzidos nas diferentes configurações, computadores com características inferiores. Como seria de esperar, a eficiência *Framework* é afetada pela existência de computadores de menor capacidade de processamento, sendo no entanto possível reduzir o impacto desses computadores, se lhes forem atribuídas tarefas de elementos menos exigentes em termos de capacidade de processamento.

Para validar o mecanismo de comunicação, foram geradas configurações com mensagens de dimensão variável em tamanho e em periodicidade. Os resultados obtidos mostraram que em situações normais, raramente se verificavam perda de mensagens. Foram ainda efetuados testes em que alguns dos computadores estavam ligados remotamente com recurso a uma VPN e os resultados não se alteraram.

6.2 Monoposto Versus Multiposto

Neste teste são comparados os Tempos de Execução de configurações com N Agentes, simulados com recurso a um ou N+1 computadores. Os resultados do teste confirmam que o recurso a uma simulação distribuída conduz a menores tempos de execução.

A Tabela 30 apresenta alguns dos parâmetros de configuração dos Agentes utilizados no teste. O Tempo de Simulação inicial não é o mesmo para todos os Agentes. No exemplo, o Agente *Customers_1_2K* inicia a simulação na unidade de tempo 6, no entanto, o Agente *Factory_4_2K* inicia na unidade de tempo 12. Neste caso, os agentes com tempo simulação inicial mais elevado, aguardam que os restantes Agentes iniciem a simulação e atinjam um Tempo de Simulação igual ou superior ao seu Tempo de Simulação inicial, e só depois iniciam a simulação. Para efeitos de teste foram também configurados Agentes com incrementos de Tempo de Simulação a variar entre 5 e 13 unidades de tempo. Todos os Agentes foram configurados com 2000 artigos em *Stock* num universo de 4000 artigos diferentes.

Tabela 30 - Agentes utilizados

Agente	**Tempo de Simulação**		**Nº artigos em Stock**
	Início	**Incremento**	
Customers_1_2K	6	11	2000
Customers_2_2K	7	12	2000
Customers_3_2K	8	13	2000
Customers_4_2K	8	13	2000
Customers_5_2K	8	13	2000
Customers_6_2K	8	13	2000
Factory_1_2K	9	5	2000
Factory_2_2K	12	6	2000
Factory_3_2K	10	6	2000
Factory_4_2K	12	6	2000

Os Agentes tipo *Customers* efetuam pedidos de catálogos, enviando mensagens do tipo *CatalogueRequest* aos Agentes que disponibilizem o serviço *CatalogueRequest*, e, em função da resposta, efetuam ou não encomendas, enviando mensagens do tipo *OrderRequest*. Os Agentes tipo *Factory* têm configurado os serviços da Tabela 31, necessários para poder responder aos pedidos de catálogos, aceitar encomendas e enviar os produtos aos clientes.

Tabela 31 - Serviços dos Agentes

Tipo de Agente	**Serviços que disponibiliza**	**Serviços que necessita**
Customers		*CatalogueRequest* *OrderRequest*
Factory	*ApplicationResponseToCatalogueRequest* *CatalogueRequest* *DespatchAdvice* *OrderRequest* *OrderResponseSimple*	

Uma das vantagens da Simulação Distribuída é poder utilizar a capacidade de processamento de vários computadores em simultâneo. Por outro lado, as comunicações entre programas a correrem em computadores diferentes, serão mais lentas do que as comunicações entre programas a correrem no mesmo computadores , o que pode ser uma desvantagem. Com este teste pretende-se avaliar o impacto das comunicações na Simulação Distribuída em comparação com a simulação num só compudador.

A Tabela 32 apresenta os Tempos de Execução das simulações realizadas, comparando o Tempo de Execução da configuração com N Agentes quando executada num computador e em N+1 computadores. Note-se que, na configuração num só computador, o EA e os N Agentes encontram-se no mesmo computador, sendo que na configuração com N+1 computadores, o EA encontra-se num computador e cada um dos N Agentes encontram-se num computador diferente. Os testes foram repetidos para configurações a variar entre dois e dez Agentes.

Tabela 32 - Comparação de Tempos de Execução

Número de Agentes (Na)	**Número de PCs**	**Tempo de Execução (ms)**	**Número de PCs**	**Tempo de Execução (ms)**	**Variação percentual**
2	1	42 291	3	52 666	125%
3	1	60 107	4	76 050	127%
4	1	62 775	5	72 696	116%
5	1	75 625	6	80 793	107%
6	1	75 521	7	76 487	101%
7	1	72 322	8	75 348	104%
8	1	112 469	9	102 166	91%
9	1	158 342	10	139 512	88%
10	1	193 373	11	156 886	81%

Como seria de esperar, quanto maior for o número de Agentes no modelo de Simulação, maior será o Tempo de Execução necessário para simular o mesmo Tempo de Simulação (Tabela 32 e Figura 27). Para uma configuração com dois Agentes, o Tempo de Execução em três computadores é cerca de 25% mais elevado do que a mesma configuração executada num só computador. Por outro lado, quando o número de Agentes do modelo aumenta, quando são utilizados vários computadores o Tempo de Execução é inferior ao da mesma configuração num só computador. No exemplo, a simulação com dez Agentes é cerca de 20% mais rápida do que na simulação num só computador, Os resultados obtidos mostram que com poucos Agentes, a Simulação num só computador é mais rápida do que se realizada em vários. Todavia, quando o número de Agentes aumenta consideravelmente, a distribuição do processamento por vários computadores, leva a que a Simulação Distribuída seja mais rápida. Os resultados estão assim de acordo com o esperado, pois quando as necessidades de processamento aumentam, o ganho do processamento distribuído compensa a perda com as comunicações.

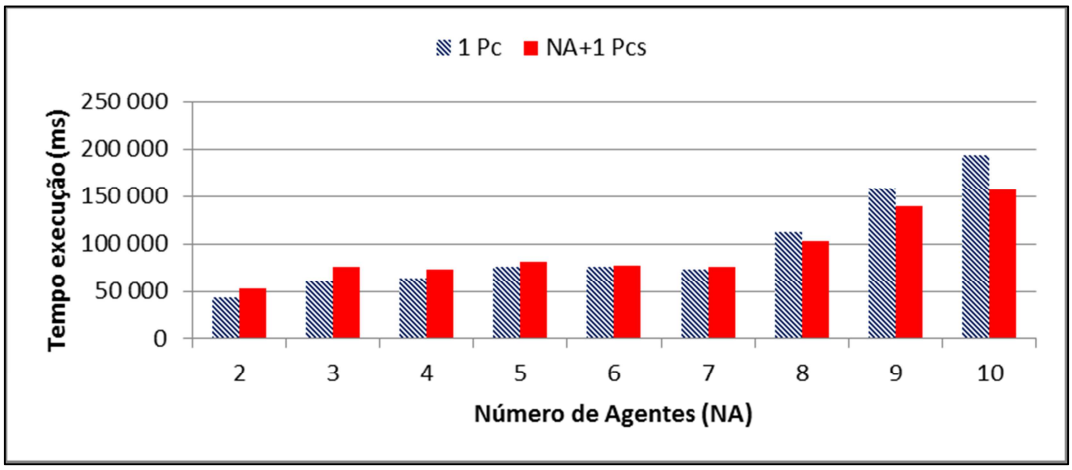


Figura 27 - Tempo de Execução (ms)

Tendo como referência, para cada configuração, o Tempo de Execução num só computador, os resultados do teste mostram que com o aumento do número de Agentes a simulação distribuída é mais eficiente. No caso de serem utilizados dez Agentes a simulação distribuída é 20% mais rápida do que a simulação num só computador (Figura 28).

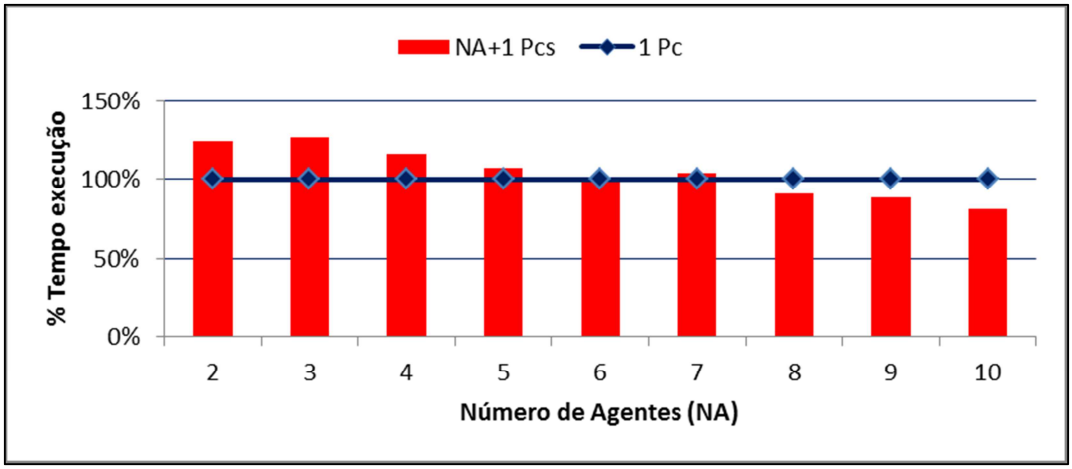


Figura 28 – Variação percentual do Tempo de Execução

6.3 Stock com 1k e 2k artigos

Neste teste são comparados os Tempos de Execução de um modelo com dez Agentes (Tabela 33) em configurações de um, três, cinco e onze computadores. Os testes foram repetidos para uma quantidade de 1000 e 2000 artigos em *Stock*. Este teste procura por um lado, estudar o

impacto do número de computadores utilizados na simulação, e por outro, comparar este impacto em função da dimensão de cada Agente.

Os serviços disponibilizados pelos Agentes tipo *Factory*, bem como serviços necessários pelos Agentes tipo *Customers*, são os mesmos do teste anterior (Tabela 31). Tal como no teste anterior a unidade de tempo inicial, bem como o incremento de Tempo de Simulação entre sincronizações, não é a mesma para todos os Agentes (Tabela 33).

Tabela 33 - Início e incremento do Tempo de Simulação dos Agentes

Agente	**Tempo de Simulação**	
	Início	**Incremento**
Customers_1	6	11
Customers_2	7	12
Customers_3	8	13
Customers_4	8	13
Customers_5	8	13
Customers_6	8	13
Factory_1	9	5
Factory_2	12	6
Factory_3	10	6
Factory_4	12	6

Para permitir a comparação entre todos os tempos de execução, nas diferentes combinações de números de computadores, e quantidade em stock, o Tempo de Execução foi normalizado, tendo como base o Tempo de Execução com todos os Agentes e o EA a correr num só computador.

A Figura 29 apresenta a variação do Tempo de Execução das diferentes combinações de número de computadores, para a simulação com 1000 e com 2000 artigos em *Stock*, tendo como base o Tempo de Execução só com um computador. Apesar de a simulação com 2000 artigos

em stock gerar mensagens com o dobro do tamanho, a variação das necessidades de processamento não é significativa. A análise dos resultados (Figura 29) permite concluir que globalmente a simulação distribuída consegue uma redução da ordem dos 10% a 20% relativamente à simulação realizada com um só computador.

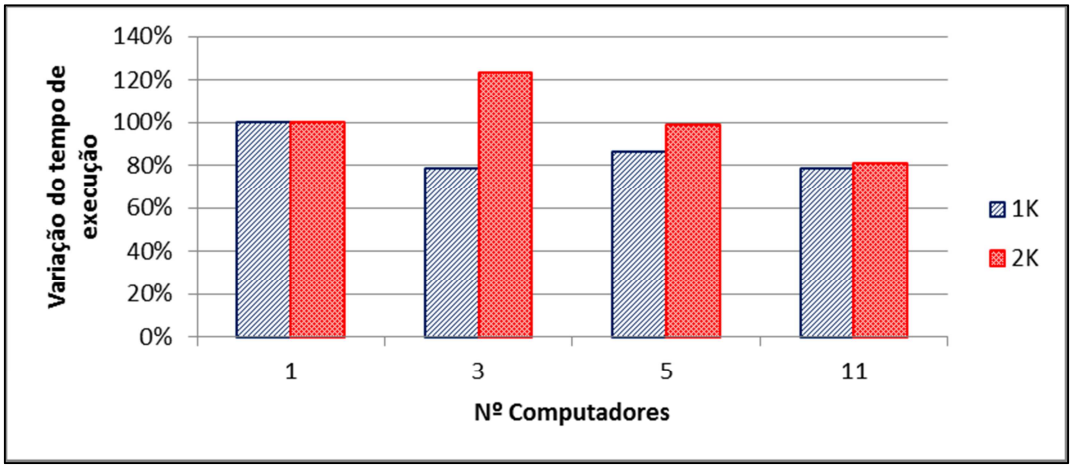


Figura 29 - Variação percentual do Tempo de Execução

Em ambas as simulações, Agentes com 1000 e 2000 artigos em stock, há um pico a partir do qual o Tempo de Execução diminui com o amento do número de computadores utilizados. A posição deste pico depende das necessidades de processamento de cada Agente e da dimensão das mensagens trocadas entre os Agentes.

Em ambos os testes, com 1000 ou 2000 artigos em stock, a utilização de um computador por Agente consegue reduções da ordem dos 20% no Tempo de Execução.

6.4 Stock com 1k, 2k e 5k artigos

A dimensão das mensagens de negócio varia de forma linear com a dimensão do Stock, mas a necessidade de processamento aumenta de forma significativa com o número de artigos em Stock (Figura 30). Numa configuração com Agentes contendo 5000 artigos em Stock, o Tempo de Execução chegou a atingir valores quatro a cinco vezes superiores ao da configuração com 2000 artigos.

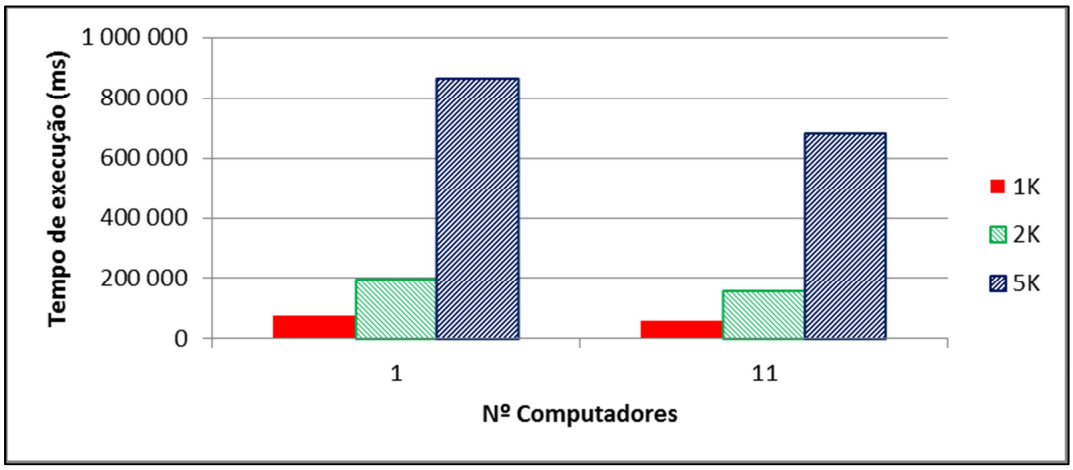


Figura 30 – Tempo de Execução (ms)

A configuração dos Agentes utilizados neste teste, em comparação com os Agentes do teste anterior, variou apenas na dimensão do Stock. Foram criados três modelos com dez agentes, com 1000, 2000 e 5000 artigos em Stock.

Como esperado, o Tempo de Execução e a dimensão das mensagens de negócio trocadas entre os Agentes, é superior no modelo com 5000. A análise dos resultados (Figura 30) mostra que mesmo com um aumento da dimensão das mensagens de negócio, independentemente do número de artigos em Stock, a simulação distribuída tem um ganho de cerca de 20% relativamente à simulação num único computador (Figura 31).

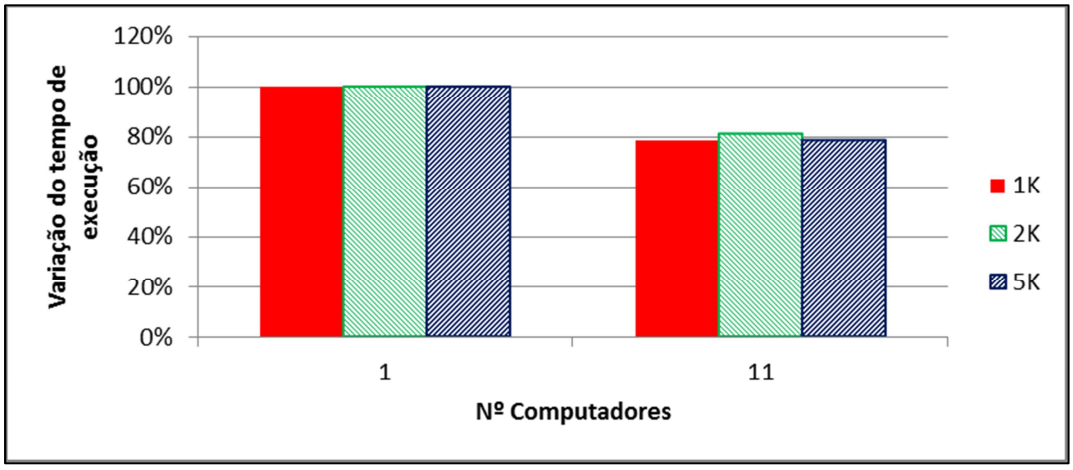


Figura 31 – Variação percentual do Tempo de Execução

À semelhança do teste anterior, podemos concluir que o aumento da dimensão das mensagens trocadas entre os Agentes é compensado com o ganho de capacidade de processamento na simulação distribuída.

Capítulo 7.

Conclusões

7.1 Síntese do Trabalho Desenvolvido

O objetivo principal deste trabalho foi a criação de um modelo (*framework*) de integração de simuladores distribuídos baseados em agentes para analisar problemas complexos dos negócios entre empresas, tendo sido selecionada a técnica ABMS por se entender que esta é a mais adequada pelas seguintes razões:

- A dimensão dos problemas a analisar, quando se pretende simular uma cadeia de abastecimentos, pode ser demasiado complexo modelar com um só simulador. A técnica ABMS tem a capacidade de modelar um sistema com componentes, com um diferente nível de detalhe, permitindo uma análise mais flexível e consistente.
- O desenvolvimento de modelos de simulação de cenários alternativos, recorrendo à técnica ABMS é um processo incomparavelmente mais fácil e flexível, do que a solução com um só simulador. O desenvolvimento de Agentes independentes para modelar uma empresa ou um sector da empresa, simplifica o processo de modelação e validação de novos simuladores. Depois de validado, o Agente pode ser replicado em diferentes cenários e criar diferentes Modelos com um menor esforço de desenvolvimento e validação.
- A técnica ABMS permite a criação de simuladores que tirem partido das capacidades computacionais de uma rede de computadores distribuídos. A *Framework* proposta permite a criação de Modelos com Agentes distribuídos por uma rede de computadores. A possibilidade de o Modelo global ser composto por diferentes Agentes que podem ser desenvolvidos e executados de forma distribuída permite o desenvolvimento descentralizado, sendo assim um incentivo ao trabalho colaborativo.

- A técnica ABMS quando aplicada numa rede de computadores, permite
    - Uma solução natural para problemas geográfica e/ou funcionalmente distribuídos.
    - Envolver os recursos humanos locais de setores específicos, no desenvolvimento do Modelo definindo as especificações particulares do seu setor independentemente da sua distribuição geográfica
- Sendo o modelo global, constituído por diferentes Agentes independentes, é possível incluir no modelo soluções (*softwares*) desenvolvidas em diferentes sistemas operativos e/ou desenvolvidos em diferentes linguagens, sendo mesmo possível a inclusão de simuladores de gerações anteriores.

Uma *framework* baseada na técnica ABMS necessita, de mecanismos de registo dos Agentes envolvidos, bem como de comunicação e sincronização temporal. Um dos objetivos iniciais consistia na possibilidade de incluir na *Framework* um mecanismo para tratar a independência relativa entre as empresas a simular. Este objetivo, levou a que fosse desenvolvido um mecanismo de controlo do avanço do Tempo de Simulação de cada um dos Agentes com base num conjunto de mensagens de sincronização trocadas entre os Agentes e o Agente desenvolvido para o efeito (*Environment Agent*). Com este mecanismo, cada Agente pode avançar no Tempo de Simulação, se não necessitar de comunicar com o ambiente que o rodeia, até ao tempo de Simulação negociado com o *Environment Agent.*

A normalização das mensagens de negócio trocadas entre os elementos de uma cadeia de abastecimentos é uma realidade dos nossos dias. Os vários projetos incluídos na Agenda Digital para a Europa da União Europeia e os vários standards recomendados pelo OASIS (Organization for the Advancement of Structured Information Standards) são a demonstração da necessidade de normalização. Para tirar partido desta normalização, foram criadas um conjunto de bibliotecas para geração de mensagens de negócio que respeitam as especificações OASIS. Estas bibliotecas agrupadas numa dll, podem ser utilizadas de forma independente por diferentes Agentes, facilitando assim a compreensão das mensagens trocadas entre os Agentes. Com esta dll, a adaptação de simuladores existentes à *Framework* desenvolvida fica

simplificada pois normaliza as comunicações. Foi dada prioridade ao desenvolvimento de mensagens relacionadas com encomendas das quais resultam a troca de produtos ou serviços numa cadeia de abastecimentos. Mensagens financeiras como faturas e recibos ficaram para desenvolvimentos futuros sendo a sua implementação semelhante às mensagens já desenvolvidas.

7.2 Contribuições da Tese

As contribuições mais importantes deste trabalho estão incluídas na *Framework* em desenvolvimento, da qual se podem salientar:

- A *Framework* adaptada à modelação de uma cadeia de abastecimentos com um conjunto de Agentes clones do Agente Base parametrizável.
- A possibilidade de gerar Agentes específicos com base no Agente Base desenvolvido, alterando o modelo ao nível de programação.
- O módulo da *Framework* responsável pelo controlo do Tempo de Simulação e pela gestão das comunicações entre os Agentes e outros simuladores.
- Bibliotecas de comunicação entre os Agentes e simuladores.
- Bibliotecas para gerar as mensagens de negócio de acordo com os recomendações da OASIS *Universal Business Language* (UBL).
- A possibilidade de adaptação e integração na *Framework* de simuladores permite a construção de modelos mais complexos, rentabilizando trabalhos anteriormente desenvolvidos, com reduzido esforço de adaptação.

7.3 Limitações e Perspetivas de Desenvolvimento Futuro

O protótipo da *Framework* em desenvolvimento, no seu estado atual, tem algumas limitações, mas é uma importante base de trabalho para desenvolvimentos futuros. Nos parágrafos seguintes, são assinalados, sem nenhuma classificação de prioridade ou importância, alguns dos possíveis trabalhos futuros.

- Completar o Modelo do Agente SA com as restantes mensagens de negócio UBL, nomeadamente mensagens financeiras.

- Desenvolver um serviço de ontologia integrado com mensagens de negócio.
- Criar mecanismo de controlo das perdas de comunicação e redes instáveis.
- Criar um mecanismo que permita a comunicação entre os Agentes com *Web Services*, de modo a permitir comunicações sem as limitações provocadas pelas Firewall dos sistemas.
- Melhorar o módulo de configuração de modo a flexibilizar a modelação de configurações alternativas.
- Melhorar o módulo de visualização do estado e resultado da simulação, nomeadamente a inclusão de interfaces que permitam uma visualização geográfica da posição dos Agentes das entidades envolvidas na simulação.
- Incluir no Agente Base heurísticas que permitam simular de forma mais realista o comportamento dos Agentes de acordo com o elemento da cadeia de abastecimento que pretende simular. A possibilidade de escolha de um modelo de gestão de stocks de entre uma lista de opções, é um exemplo de uma destas heurísticas a implementar.

Certamente que muitas questões de investigação continuam sem resposta, mas espera-se que este trabalho possa motivar a realização de desenvolvimentos futuros, dando assim continuidade ao trabalho aqui iniciado.

Bibliografia

Barros, Fernando J. 2007. “Modeling and Simulation of Parallel Adaptive Divide-and-conquer Algorithms.” *The Journal of Supercomputing* 43 (3) (June 28): 241–255. doi:10.1007/s11227-007-0143-3.

Barros, Fernando J. 2011. “Modeling and Simulation of Dynamic Toplogy Models: An Overview of Applications.” *Modeling and Simulation Magazine* 2: 103–111.

Bellifemine, FL, G Caire, and D Greenwood. 2007. *Developing Multi-agent Systems with JADE*.

Berryman, Matthew. 2008. “Review of Software Platforms for Agent Based Models.” *Science And Technology*.

Bosch, Peter C., and Majdi Rajab. 2004. “Autonomous Predictive-adaptive Simulation for Operations Support.” In *Simulation Conference, 2004.*, 2:2018–2024. IEEE.

Brailsford, SC, and NA Hilton. 2001. “A Comparison of Discrete Event Simulation and System Dynamics for Modelling Health Care Systems”: 1–17.

Brito, AESC. 1992. “Configuring Simulation Models Using CAD Techniques: a New Approach to Warehouse Design”. Cranfield Institute of Technology.

Brito, António Carvalho, Carlos Bragança de Oliveira, and Paulo Sá Marques. 2011. “A Simulation Study of a Multi-Site Production Plant Using ARENA.” In *ESM’2011*. Guimarães,Portugal.

Brito, António E S Carvalho. 1999. “Introduction to a Warehouse Visual Simulator.” *Management*: 14–16.

Carvalho, JÁ, and MP Morais. 2001. “Sistemas Informáticos e Conhecimento Organizacional: Uma Reinterpretação Dos Papeis Desempenhados Pelos Sistemas Informáticos Nas Organizações.” In *CONFERÊNCIA DA ASSOCIAÇÃO PORTUGUESA DE SISTEMAS DE INFORMAÇÃO*. Évora.

Castle, CJE, and A.T. Crooks. 2006. “Principles and Concepts of Agent-based Modelling for Developing Geospatial Simulations.” *UCL Discovery* 44 (0).

DAE. 2013. “Digital Agenda for Europe (Scoreboard) - European Commission.” http://ec.europa.eu/digital-agenda/en/scoreboard.

Dias, L, G Pereira, and G Rodrigues. 2007. “A Shortlist of the Most Popular Discrete Simulation Tools.” *Simulation News Europe*.

Dias, L, AJ Rodrigues, and G Pereira. 2005. “An Activity Oriented Visual Modelling Language with Automatic Translation to Different Paradigms”: 1–10.

ebXML. 2013. “Electronic Business Using XML (ebXML) Standards.” http://ebxml.xml.org/.

e-CODEX. “e-CODEX: Home.” http://www.e-codex.eu/home.html.

epSOS. “epSOS: Home.” http://www.epsos.eu/.

Feliz-Teixeira, J Manuel, and António E S Carvalho Brito. 2005. “Using Simulation to Analyse the Procurement of Additives for Lubricants in the Oil Refinery of Porto , Portugal.” *Industrial Simulation Conference*.

Feliz-teixeira, J Manuel, and António E S Carvalho Brito. 2006. “Holistic Metrics , a Trial on Interpreting Complex Systems.” In *Simulation*.

Ferreira, Isabel, Cândida Silva, Sílvia Ferreira, and João Álvaro Carvalho. 2012. “Dilemas Iniciais Na Investigação Em TSI.” In *7ª Conferência Ibérica De Sistemas e Tecnologias De Informação*.

Fujimoto, Richard M. 2000. *Parallel and Distributed Simulation Systems*. Ed. ALBERT Y. ZOMAYA. John Wiley & Sons, Inc.

Fujimoto, RM. 2001. “Parallel and Distributed Simulation.” In *Proceedings of the 2001 Winter Simulation Conference*, 1:147–157. IEEE.

Geraldes, CAS, S Carvalho, and G Pereira. 2011. “An Integrated Approach for Warehouse Design and Planning.”

Gilbert, Nigel, and Steven Bankes. 2002. “Platforms and Methods for Agent-based Modeling.” *Proceedings of the National Academy of Sciences of the United States of America* 99 Suppl 3 (May 14): 7197–8. doi:10.1073/pnas.072079499.

Hevner, Alan R. 2007. “A Three Cycle View of Design Science Research.” *Scandinavian Journal of Information Systems* 19 (2): 87–92.

Hevner, AR, ST March, Jinsoo Park, and Sudha Ram. 2004. “Design Science in Information Systems Research.” *MIS Quarterly* 28 (1): 75–105.

IEEE Std. 2000. “IEEE Standard for Modeling and Simulation (M Amp;S) High Level Architecture (HLA) - Framework and Rules.” *IEEE Std 1516-2000*. doi:10.1109/IEEESTD.2000.92296.

ISA. “Interoperability Solutions for European Public Administrations - ISA - European Commission.” http://ec.europa.eu/isa/index_en.htm.

Jennings, N.R., P. Faratin, T.J. Norman, P. O’Brien, B. Odgers, and J.L. Alty. 2000. “Implementing a Business Process Management System Using ADEPT: A Real-world Case Study.” *Applied Artificial Intelligence* 14 (5) (June): 421–463. doi:10.1080/088395100403379.

Jennings, N.R., T.J. Norman, and P Faratin. 1998. “ADEPT: An Agent-based Approach to Business Process Management.” *ACM Sigmod Record* 27 (4): 32–39.

Jennings, Nicholas R. 2000. “ScienceDirect - Artificial Intelligence : On Agent-based Software Engineering*1.” *Artificial Intelligence* 117 (2) (March): 277–296. doi:10.1016/S0004-3702(99)00107-1.

LSPs. “eGovernment Large Scale Pilot Projects.” https://ec.europa.eu/digital-agenda/en/egovernment-large-scale-pilot-projects.

Macal, C M, and M J North. 2005. “Tutorial on Agent-based Modeling and Simulation.” In *Proceedings of the 2005 Winter Simulation Conference*, 2–15. Orlando, Florida: Winter Simulation Conference.

Macal, C M, and M J North. 2009. “Agent-based Modeling and Simulation.” In *Proceedings of the 2009 Winter Simulation Conference*, 86–98. IEEE. doi:10.1109/WSC.2009.5429318.

Macal, C M, and M J North. 2010. “Tutorial on Agent-based Modelling and Simulation.” *Journal of Simulation* 4 (3) (September): 151–162. doi:10.1057/jos.2010.3.

Macedo, Ana Paula Cunha da Rocha. 2001. “Metodologias De Negociação Em Sistemas Multi-agentes Para Empresas Virtuais”. Porto:: FEUP.

Malucelli, A, D Palzer, and E Oliveira. 2004. “B2B Transactions Enhanced with Ontology-Based Services.” *ICETE’04--1St International …*: 10–17.

Markus, ML, A Majchrzak, and L Gasser. 2002. “A Design Theory for Systems That Support Emergent Knowledge Processes.” *Mis Quarterly*.

Marzouk, Mohamed, Ibrahim Bakry, and Moheeb El-Said. 2010. “Application Of Lean Principles In Construction Consultancy Firms.” In *Proceedings of the 2010 Winter Simulation Conference*.

Minar, N, R Burkhart, C Langton, and M Askenazi. 1996. “The Swarm Simulation System: A Toolkit for Building Multi-agent Simulations.”

Navy, U.s. 2004. “Modeling and Simulation Verification , Validation , and Accreditation Implementation Handbook.” *DEPARTMENT OF THE NAVY -USA*.

Nfaoui, El Habib, Yacine Ouzrout, Omar El Beqqali, and Abdelaziz Bouras. 2006. “An Approach of Agent-based Distributed Simulation for Supply Chains: Negotiation Protocols Between Collaborative Agents.” *European Simulation and Modelling Conference* 01.

Nicol, David M, and Jason Liu. 2002. “Composite Synchronization in Parallel Discrete-event Simulation.” *IEEE Transactions on Parallel and Distributed Systems* 13 (5) (May): 433–446. doi:10.1109/TPDS.2002.1003854.

Niewiadomska-Szynkiewicz, E., M. Zmuda, and K. Malinowski. 2003. “Application of a Java-based Framework to Parallel Simulation of Large-scale Systems.” *International Journal of Applied Mathematics and Computer Science* 13 (4): 537–548.

North, M.J., and C.M. Macal. 2007. *Managing Business Complexity: Discovering Strategic Solutions with Agent-based Modeling and Simulation*. Oxford university press, USA.

OASIS. 2007. “Code List Representation (Genericode) Version 1.0.” http://docs.oasis-open.org/codelist/genericode/doc/oasis-code-list-representation-genericode.pdf.

OASIS. 2013. “About Us | OASIS.” https://www.oasis-open.org/org.

OASIS CCTS. 2003. “Core Components | EBXML.” http://ebxml.xml.org/core-components.

OASIS CPPA. 2002. “ebXML Collaboration Protocol Profile and Agreement (CPPA).” http://ebxml.xml.org/cppa.

OASIS DITA. 2010. “Darwin Information Typing Architecture OASIS Standard.” http://dita.xml.org/.

OASIS ebBP. 2002. “ebXML Business Process (ebBP) OASIS Standard.” http://ebxml.xml.org/bp.

OASIS ebMS. 2002. “Messaging Services | EBXML.” http://ebxml.xml.org/messaging.

OASIS ODF. 2006. “OpenDocument Format (ODF) OASIS Standard.” http://opendocument.xml.org/.

OASIS RIM. 2002. “Registry/Repository | EBXML.” http://ebxml.xml.org/regrep.

OASIS SAML. 2005. “Security Assertion Markup Language (SAML) OASIS Standard.” http://saml.xml.org/.

OASIS UBL. 2011. “Universal Business Language Version 2.1.” http://docs.oasis-open.org/ubl/prd2-UBL-2.1/UBL-2.1.html.

OASIS UDDI. 2005. “Universal Description, Discovery, and Integration.” http://uddi.xml.org/.

Oliveira, Carlos Bragança de, and António Carvalho Brito. 2011. “Multi-Hierarchical Agent Based Modelling Simulation Framework.” In *ESM’2011*. Guimarães,Portugal.

Oliveira, Carlos Bragança de, and António Carvalho Brito. 2012. “A Business Language for a Distributed Simulation Framework.” In *ESM’2012*. Essen, Germany.

Peffers, K, T Tuunanen, and CE Gengler. 2006. “The Design Science Research Process: a Model for Producing and Presenting Information Systems Research.” In *DESRIST*. Claremont.

Peffers, Ken, Tuure Tuunanen, Marcus a. Rothenberger, and Samir Chatterjee. 2007. “A Design Science Research Methodology for Information Systems Research.” *Journal of Management Information Systems* 24 (3) (December 1): 45–77. doi:10.2753/MIS0742-1222240302.

Peito, F, G Pereira, and A Leitão. 2011. “Simulation as a Decision Support Tool in Maintenance Float Systems.” *17th European Concurrent Engineering …*.

Peito, F, G Pereira, A Leitão, and L Dias. 2011. “Simulation as a Decision Support Tool in Maintenance Float Systems: The Automatic Generation of Simulation Programs.” In .

PEPPOL. 2013. “PEPPOL | Pan-European Public Procurement Online.” http://www.peppol.eu/.

Pereira, AMC. 2010. “Intelligent Simulation of Coastal Ecosystems”. University of Porto.

Pereira, G, L Dias, P Vik, and JA Oliveira. 2011. “Discrete Simulation Tools Ranking: a Commercial Software Packages Comparison Based on Popularity.”

Pereira, G, F Peito, A Leitão, and L Dias. 2011. “Simulation as a Decision Support Tool in Maintenance Float Systems: System Availability Versus Total Maintenance Cost.” In .

Pidd, M, and A Carvalho. 2006. “Simulation Software: Not the Same Yesterday, Today or Forever.” *Journal of Simulation* 1 (1) (December): 7–20. doi:10.1057/palgrave.jos.4250004.

Railsback, S. F., S. L. Lytinen, and S. K. Jackson. 2006. “Agent-based Simulation Platforms: Review and Development Recommendations.” *SIMULATION* 82 (9) (September 1): 609–623. doi:10.1177/0037549706073695.

Robinson, Stewart. 2004. “Discrete-event Simulation: From the Pioneers to the Present, What Next?” *Journal of the Operational Research Society*.

Sargent, Robert G. 2011. “Verification and Validation of Simulation Models.” *Proceedings of the 2011 Winter Simulation Conference*: 183–198.

Schlegel, Sacha. 2005. “The ebXML Collaboration Protocol Agreement Formation Process”. Curtin University of Technology.

Shannon, RE. 1998. “Introduction to the Art and Science of Simulation.” In *Proceedings of the 30th Conference on Winter Simulation*, 7–14.

Silva, DAGC. 2013. “Cooperative Multi-robot Missions: Development of a Platform and a Specification Language”. Faculdade de Engenharia da Universidade do Porto.

SPOCS. “SPOCS (Simple Procedures Online for Cross- Border Services).” http://www.eu-spocs.eu/.

Srbljinović, A, and Ognjen Škunca. 2003. “An Introduction to Agent Based Modelling and Simulation of Social Processes.” *Interdisciplinary Description of Complex …* 1: 1–8.

Stone, Peter, and M Veloso. 2000. “Multiagent Systems: A Survey from a Machine Learning Perspective.” *Autonomous Robots*.

STORK 2.0. “Stork 2.0: Home.” https://www.eid-stork2.eu/.

Syriani, E., H. Vangheluwe, and A. Al-Mallah. 2011. “Modelling and Simulation-based Design of a Distributed DEVS Simulator.” In *Proceedings of the 2011 Winter Simulation Conference*, 3007–3021.

Tako, AA, and Stewart Robinson. 2009. “Comparing Model Development in Discrete Event Simulation and System Dynamics.” *Simulation Conference (WSC), …* (Sweetser 1999): 979–991.

Teixeira, J Manuel Feliz, and António E S Carvalho Brito. 2004. “On Measuring the Supply Chain Flexibility.” *Information Systems* (February).

Teixeira, J Manuel Feliz, António E S Carvalho Brito, and Richard Saw. 2004. “Distributed Application for Supply Chain Management Training.” In *Industrial Simulation Conference (ISC)*. Malaga.

Teixeira, J.M. 2006. “Flexible Supply Chain Simulation (thesis)”. Faculdade de Engenharia da Universidade do Porto.

Teixeira, JMF, and AESC Brito. 1999. “Visual C++ Software for Warehouse Simulation (an Overview).” *11th European Simulation Symposium, …* (October): 24–27.

Teixeira, JMF, and AESC Brito. 2003. “An Approach for Dynamic Supply Chain Modelling.” *ESM’2003* 5 (2) (December): 55–63.

Tobias, R., and C. Hofmann. 2004. “Evaluation of Free Java-libraries for Social-scientific Agent Based Simulation.” *Journal of Artificial Societies and Social Simulation* 7 (1) (January 31).

Vesiluoma, Sari. 2009. “Understanding and Supporting Knowledge Sharing in Software Engineering.” *Tampere University of Technology, Publication* (843).

Vik, P, L Dias, and G Pereira. 2010. “Using Simio for the Specification of an Integrated Automated Weighing Solution in a Cement Plant.” In *Proceedings of the Winter …*, 1534–1546.

Wallace, Peter Dungan. 2009. “SMT Goes ABMS: Developing Strategic Management Theory Using Agent-Based Modelling and Simulation.” *Management*. Durham University.

Wang, Wei, Yingguang Li, Weiming Shen, Xiaoping Li, and Wenping Mou. 2012. “An Industrial Case Study of Feature-based In-process Workpiece Modeling.” *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (October): 814–819. doi:10.1109/ICSMC.2012.6377828.

White, K. Preston, and Ricki G. Ingalls. 2009. “Introduction to Simulation.” In *Proceedings of the 2009 Winter Simulation Conference (WSC)*, 12–23. IEEE. doi:10.1109/WSC.2009.5429315.

Wooldridge, Michael. 2008. *An Introduction to Multiagent Systems*.

WS-BPEL. 2007. “Web Services Business Process Execution Language.” http://bpel.xml.org/.

Anexo A. Tipos de documentos UBL 2.1

UBL 2.1 define vários tipos de documentos de suporte aos processos de negócios. A tabela a seguir mostra todos os tipos de documentos UBL 2,1 os processos de negócio associados e as partes envolvidas na troca de mensagens.

Tabela A 1 - Tipos de documentos UBL 2.1(OASIS UBL 2011)

DOCUMENT NAME	DESCRIPTION	PROCESSES INVOLVED	SUBMITTER ROLE	RECEIVER ROLE
Application Response	A document to indicate the application s response to a transaction. This may be a business response and/or a technical response, sent automatically by an application or initiated by a user.	Any	Sender	Receiver
Attached Document	A UBL wrapper that allows a document of any kind to be packaged with the UBL document that references it.	Any	Sender	Receiver
Awarded Notification	The document used to communicate the contract award to the winner	Tendering		
Bill Of Lading	The Bill of Lading is issued by the party who acts as an agent for the carrier or other agents to the party who gives instructions for the transportation services (shipper, consignor, etc.) stating the details of the transportation, charges, and terms and conditions under which the transportation service is provided. The party issuing this document does not necessarily provide the physical transportation service. It corresponds to the information on the Forwarding Instruction. It is used for any mode of transport. A Bill of Lading can serve as a contractual document between the parties for the transportation service. The document evidences a contract of carriage by sea and the acceptance of responsibility for the goods by	Freight Management	Freight Forwarder, Carrier	Consignor (or Consignee), Freight Forwarder

DOCUMENT NAME	DESCRIPTION	PROCESSES INVOLVED	SUBMITTER ROLE	RECEIVER ROLE
	the carrier, and by which the carrier undertakes to deliver the goods against surrender of the document. A provision in the document that the goods are to be delivered to the order of a named person, or to order, or to bearer, constitutes such an undertaking.			
Call For Tenders	The document used for a Contracting Party to define the procurement project to buy goods, services or works during an specified period.	Tendering	Contracting Authority	Tenderer
Catalogue	The document that describes items, prices, and price validity.	Catalogue	Seller	Contracting Party
Catalogue Deletion	The document used to cancel an entire Catalogue.	Catalogue	Seller	Contracting Party
Catalogue Item Specification Update	The document used to update information about Items (e.g., technical descriptions and properties) on an existing Catalogue.	Catalogue	Seller	Contracting Party
Catalogue Pricing Update	The document used to update information about prices on an existing Catalogue.	Catalogue	Seller	Contracting Party
Catalogue Request	The document used to request a Catalogue.	Catalogue	Contracting Party	Seller
Certificate Of Origin	A document that describes the Certificate of Origin.	Certification of Origin of Goods	Exporter, Issuer	Issuer, Importer
Contract Award Notice	The document published by a Contracting Party to announce the awarding of a procurement project.	Tendering	Contracting Authority	Tenderer
Contract Notice	The document used for a Contracting Party to announce the project to buy goods, services or works.	Tendering	Contracting Authority	Tenderer
Credit Note	The document used to specify credits due to the Debtor from the	Billing	Supplier Accounting Party	Customer Accounting

DOCUMENT NAME	DESCRIPTION	PROCESSES INVOLVED	SUBMITTER ROLE	RECEIVER ROLE
	Creditor.			Party
Debit Note	The document used to specify debits made by the Debtor.	Billing	Customer Accounting Party	Supplier Accounting Party
Despatch Advice	The document used to describe the despatch or delivery of goods and services.	Fulfilment	Despatch	Delivery
Document Status	A document used to provide information about document status.	Any collaboration	Party currently controlling Status of the collaboration	Party requesting Status on collaboration
Document Status Request	A document used to request the status of another document.	Any collaboration	Party requesting Status on collaboration	Party currently controlling Status of the collaboration
Exception Criteria	Used to specify basic information about the content of the message including version number, creation date and time.	Collaborative Planning, Forecasting and Replenishment		
Exception Notification	The document used to notify an exception	Collaborative Planning, Forecasting and Replenishment		
Forecast	The document used to specify a forecast.	Collaborative Planning, Forecasting and Replenishment		
Forecast Revision	The document used to revise a Forecast.	Collaborative Planning, Forecasting and Replenishment		

DOCUMENT NAME	DESCRIPTION	PROCESSES INVOLVED	SUBMITTER ROLE	RECEIVER ROLE
Forwarding Instructions	The document issued to a forwarder, giving instructions regarding the action to be taken for the forwarding of goods described therein. Forwarding Instructions is used by any party who gives instructions for the transportation services required for a consignment of goods to any party who is contracted to provide the transportation services. The parties who issue this document are commonly referred to as the shipper or consignor, while the parties who receive this document are forwarders, carriers, shipping agents, etc. Note that this document may also be issued by a forwarder or shipping agent in their capacity as a shipper . This document can be used to arrange for the transportation (1) of different types of goods or cargoes; (2) whether containerized or non-containerized; (3) through different modes of transport including multi-modal; and (4) from any origin to any destination.	Freight Management	Consignor (or Consignee), Freight Forwarder	Freight Forwarder, Carrier
Freight Invoice	A document stating the charges incurred for the logistics service.	Freight Billing	Freight Forwarder	Consignor or Consignee
Goods Item Itinerary	A document specifying the route and the time schedule for a transport of a Goods Item for all segments in a transport service.	Intermodal Freight Management	Transport Service Provider	Transport User
Guarantee Certificate	A document to notify the deposit of a guarantee.	Tendering	Tenderer	Contracting Authority
Instruction For Returns	This document is used to initiate a return of goods. The producer is requesting products which are badly sold either for use in other places or just to free the area from it.	Cyclic Replenishment Program		
Inventory Report	Report about the quantities of each item which are or will be on stock.	Cyclic Replenishment Program		

DOCUMENT NAME	DESCRIPTION	PROCESSES INVOLVED	SUBMITTER ROLE	RECEIVER ROLE
Invoice	The document used to request payment.	Billing	Supplier Accounting Party	Customer Accounting Party
Item Information Request	The document used to request product activity, forecast, or performance data.	Collaborative Planning, Forecasting and Replenishment		
Order	The document used to order goods and services.	Ordering	Buyer	Seller
Order Cancellation	The document used to cancel an entire Order.	Ordering, Fulfilment	Buyer	Seller
Order Change	The document used to specify changes to an existing Order.	Ordering, Fulfilment	Buyer	Seller
Order Response	The document used to indicate detailed acceptance or rejection of an Order or to make a counter-offer.	Ordering	Seller	Buyer
Order Response Simple	The document used to indicate simple acceptance or rejection of an entire Order.	Ordering	Seller	Buyer
Packing List	A document stating the detail of how goods are packed.	Freight Management	Consignor	Freight Forwarder
Performance History	Performance History represents a collection of values gathered for key performance metrics in the trading partner relationship.	Cyclic Replenishment Program		
Prior Information Notice	The document used for a Contracting Party to declare the intention to buy goods, services or works during an specified period.	Tendering		
Product Activity	Product activity represents movement of a product through a location in terms of the base unit of measure for the item.	Cyclic Replenishment Program		
Quotation	The document used to quote for the provision of goods and services.	Quotation	Seller	Originator

DOCUMENT NAME	DESCRIPTION	PROCESSES INVOLVED	SUBMITTER ROLE	RECEIVER ROLE
Receipt Advice	The document used to describe the receipt of goods and services.	Fulfilment	Delivery	Despatch
Reminder	The document used to remind the customer of payments overdue.	Billing	Supplier Accounting Party and/or Payee	Customer Accounting Party and/or Payee
Remittance Advice	The document used to specify details of an actual payment.	Payment	Supplier Accounting Party and/or Payee	Customer Accounting Party and/or Payee
Request For Quotation	The document used to request a Quotation for goods and services from a Seller.	Quotation	Originator	Seller
Retail Event	The document used to specify basic information about the content of the Retail Event Meassage message including version number, creation date and time.	Cyclic Replenishment Program		
Self Billed Credit Note	The Credit Note created by the Debtor in a Self Billing arrangement with a Creditor; Self Billed Credit Note replaces Debit Note in such arrangements.	Billing	Customer Accounting Party	Supplier Accounting Party
Self Billed Invoice	The Invoice document created by the Customer (rather than the Supplier) in a Self Billing relationship.	Billing	Customer Accounting Party	Supplier Accounting Party
Statement	The document used to specify the status of Orders, Billing, and Payment. This document is a Statement of Account and not intended as a summary Invoice	Billing	Supplier Accounting Party	Customer Accounting Party
Stock Availability Report	Report about the quantities of each item which are or will be on stock.	Cyclic Replenishment Program		
Tender	A message which a tenderer offers a tender to the tendering organization for bid.	Tendering	Tenderer	Contracting Authority

DOCUMENT NAME	DESCRIPTION	PROCESSES INVOLVED	SUBMITTER ROLE	RECEIVER ROLE
Tenderer Qualification	A document used for the Tenderer to declare things about his own condition.	Tendering	Tenderer	Contracting Authority
Tenderer Qualification Response	A message which the procurement organization sends to an economic operator in order to notify its admision or exclusion to/from the tendering process	Tendering	Contracting Authority	Tenderer
Tender Receipt	A message sent by the Contracting Party to an Economic Operator in order to notify the reception of the tendering offer	Tendering	Contracting Authority	Tenderer
Trade Item Location Profile	This document is used to send trade item attributes which are focused on replenishment policies.	Collaborative Planning, Forecasting and Replenishment		
Transportation Status	A message to report the transport status and/or change in the transport status (i.e. event) between agreed parties.	Freight Status Reporting	Freight Forwarder	Consignee, Consignor
Transport Execution Plan	A document which is used in the negotiation of a transport service between a transport user and a transport service provider	Intermodal Freight Management	Transport User, Transport Service Provider	Transport Service Provider, Transport User
Transport Execution Status	The Transport Execution Status is used to provide the transport user with timing and condition status information about the transport operation	Intermodal Freight Management	Transport Service Provider	Transport User
Transport Progress Status	A document being sent from Transportation Network Manager to Transport Service Provider giving a status on the transport means	Intermodal Freight Management	Transportation Network Manager	Transport Service Provider
Transport Service Description	A document being sent from the Transport Service Provider to the Transport User in order to announce a transport service	Intermodal Freight Management	Transport Service Provider	Transport User
Unawarded	The document used to communicate the contract has been awarded to	Tendering	Contracting	Tenderer

DOCUMENT NAME	DESCRIPTION	PROCESSES INVOLVED	SUBMITTER ROLE	RECEIVER ROLE
Notification	another tenderer		Authority	
Utility Statement	The Utility Statement contains information on the consumption of services provided by utility suppliers to private and public customers. These utilities include electricity, gas, water and telephony services. The Utility Statement is therefore a supplement to an Invoice or CreditNote.	Utility Billing	Supplier Accounting Party	Customer Accounting Party
Waybill	The Waybill is issued by the party who acts as an agent for the carrier or other agents, to the party who gives instructions for the transportation services (shipper, consignor, etc.) stating the details of the transportation, charges, and terms and conditions under which the transportation service is provided. The party issuing this document could be a party other than that providing the physical transportation. It corresponds to the information on the Forwarding Instruction. It is used for all modes of transport. It can serve as a contractual document between the parties for the transportation service. The document made out by the carrier or on behalf of the carrier evidencing the contract for the transport of cargo.	Freight Management	Freight Forwarder, Carrier	Consignor (or Consignee), Freight Forwarder

Anexo B. DLLs da *Framework*

Na Figura B 1 estão representadas as bibliotecas mais relevantes da *Framework* relacionadas com o *Environment Agent* (EA) e os *Simulation Agents* (SA). Nas Figuras seguintes estão representadas com mais detalhe os módulos mais importantes.

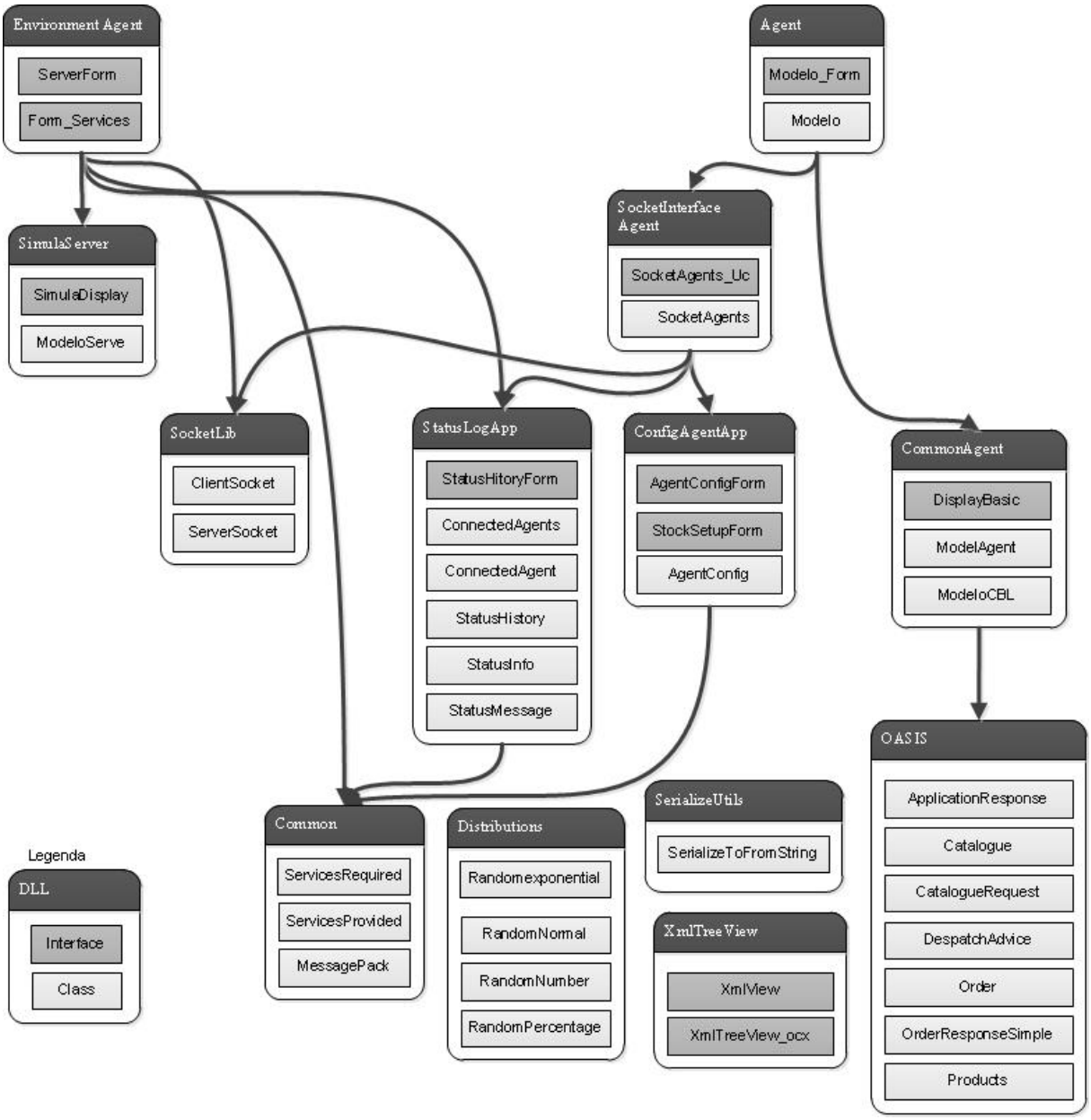


Figura B 1 - Bibliotecas da *Framework*

- *Agente Base*

A interface com o utilizado e o *Environment Agent* (EA) (Figura B 2) estão separados do Modelo (Figura B 3) de modo a facilitar o desenvolvimento de novos agentes.

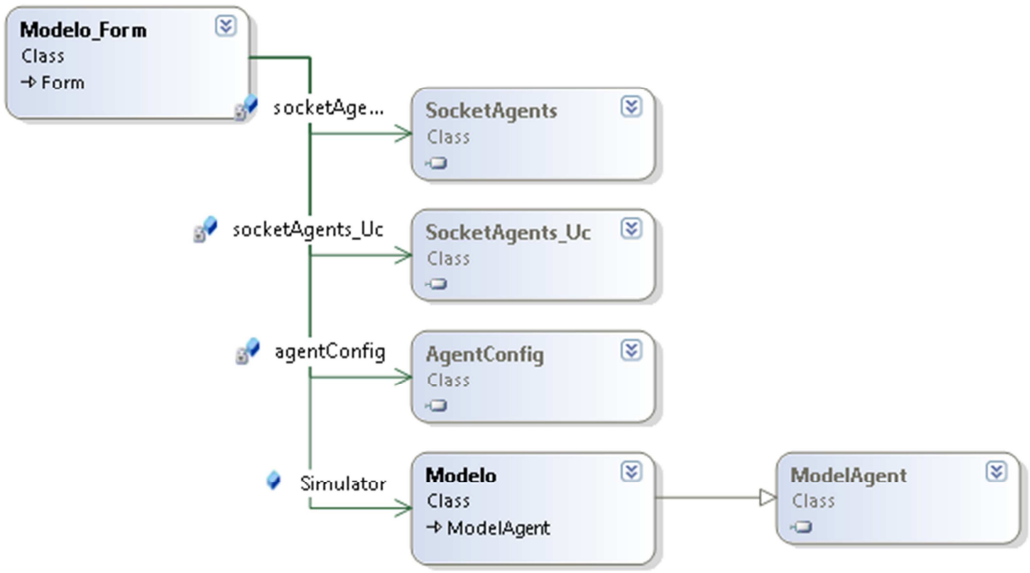


Figura B 2 – Interface

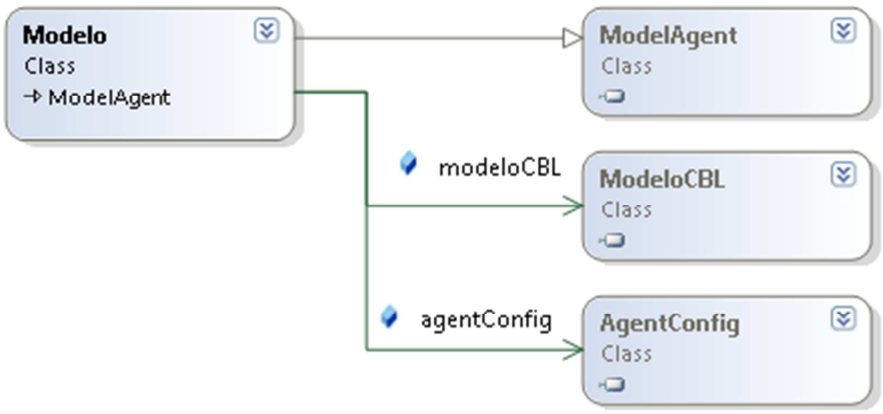


Figura B 3 - Modelo

- *ConfigAgentApp*

Módulo de configuração do Agente

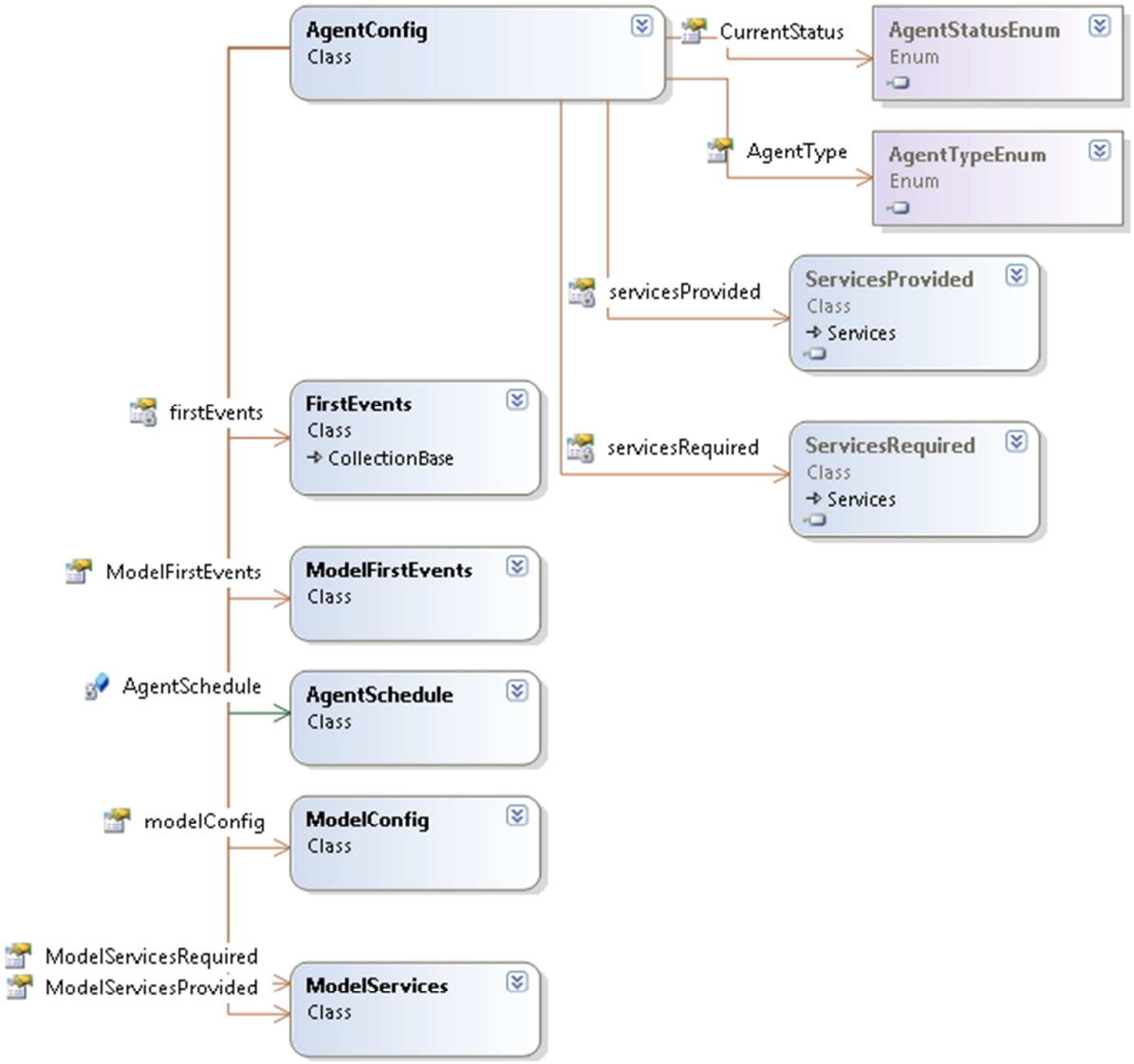


Figura B 4 – Agente Config

- *SocketInterfaceAgent*

Interface com o *Socket* responsável pelas comunicações

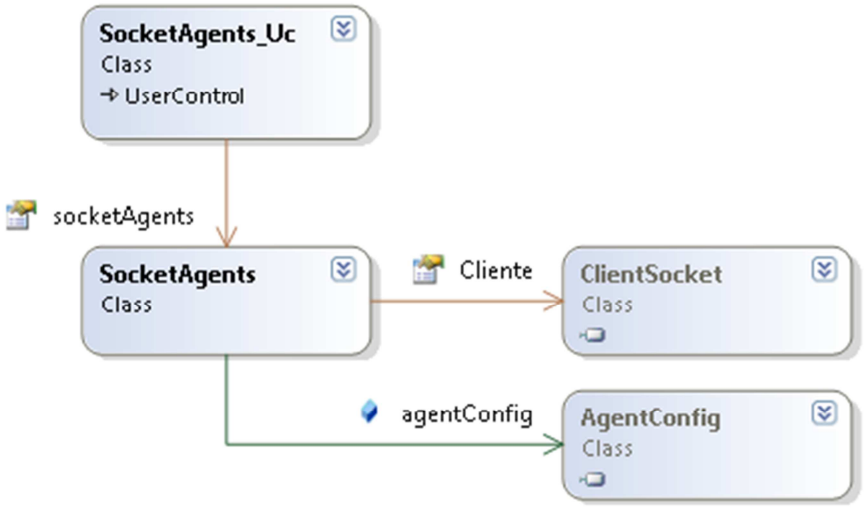
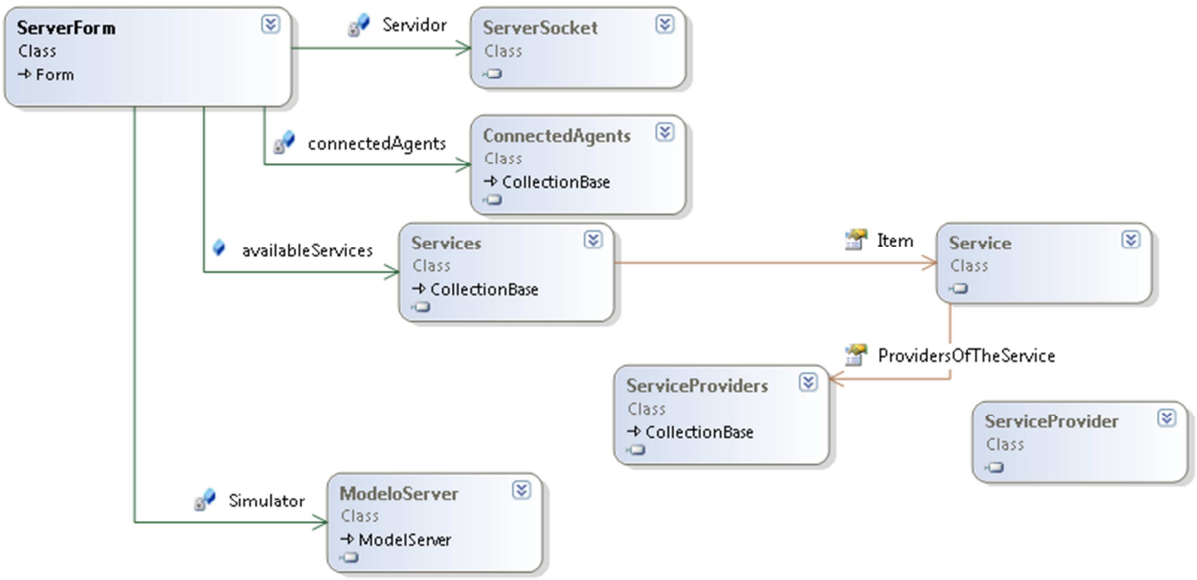


Figura B 5 – Interface com o *Socket*

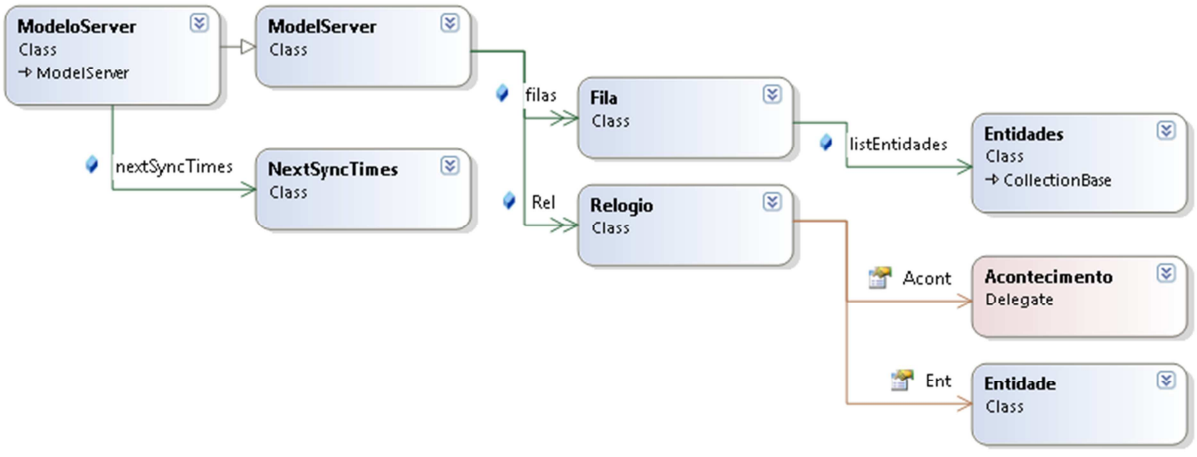
- *Environment Agent*

Interface do *Environment Agent*



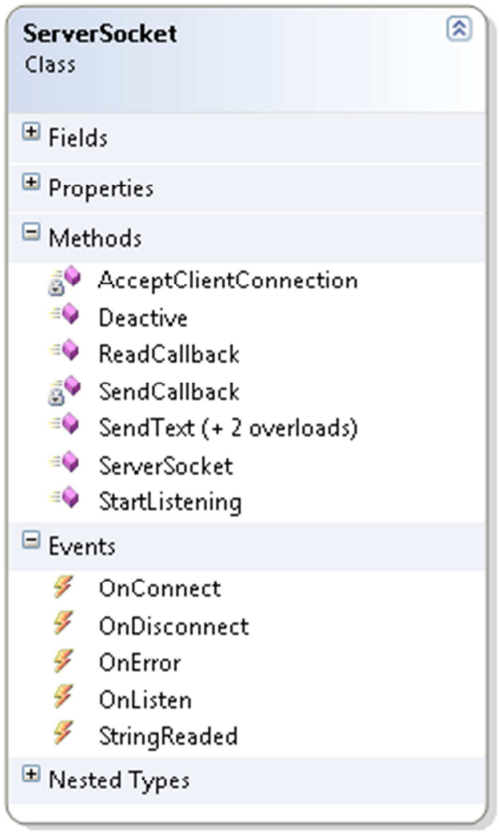
- *Environment Agent – SimulaServer*

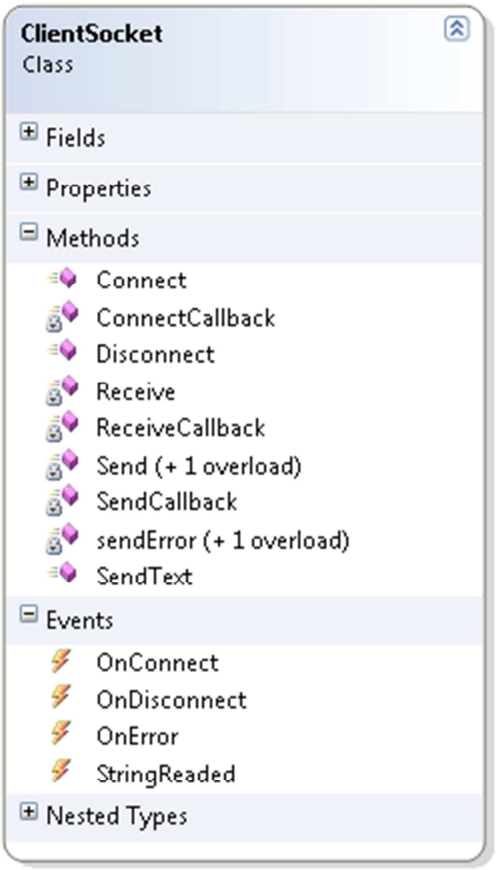
Módulo de simulação



- *SocketLib*

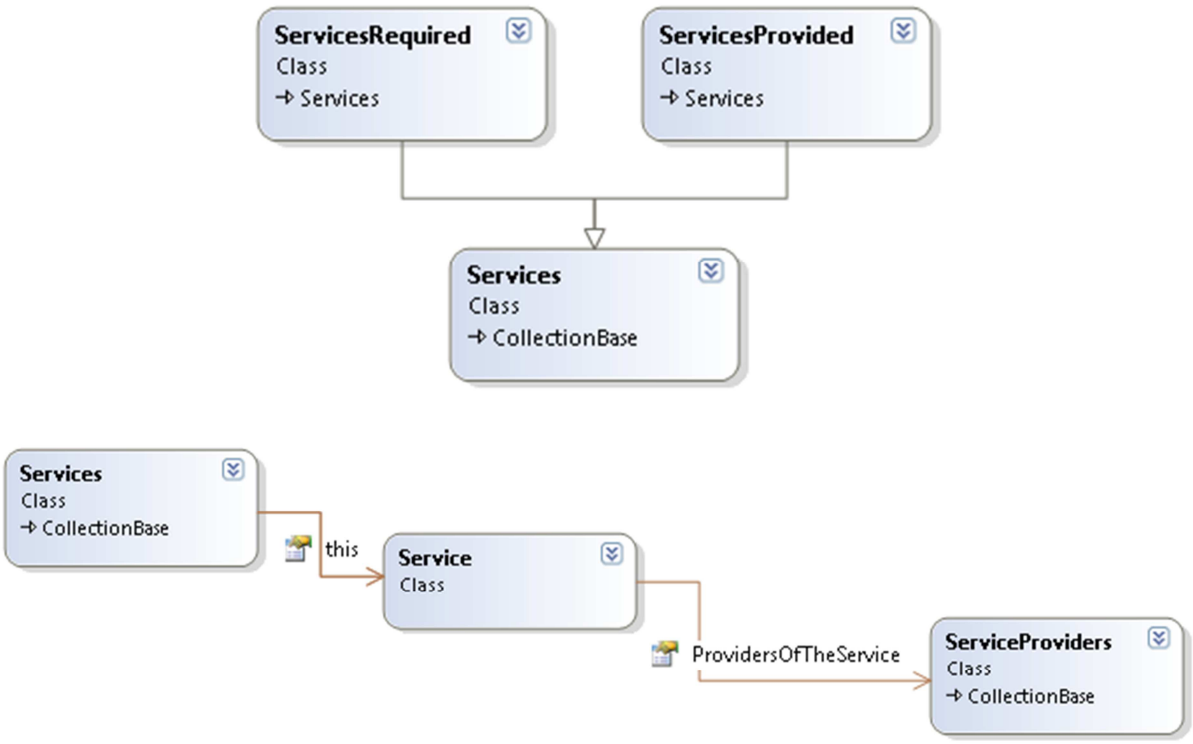
Biblioteca responsável pelas comunicações entre os Agentes e o *Environment Agent*





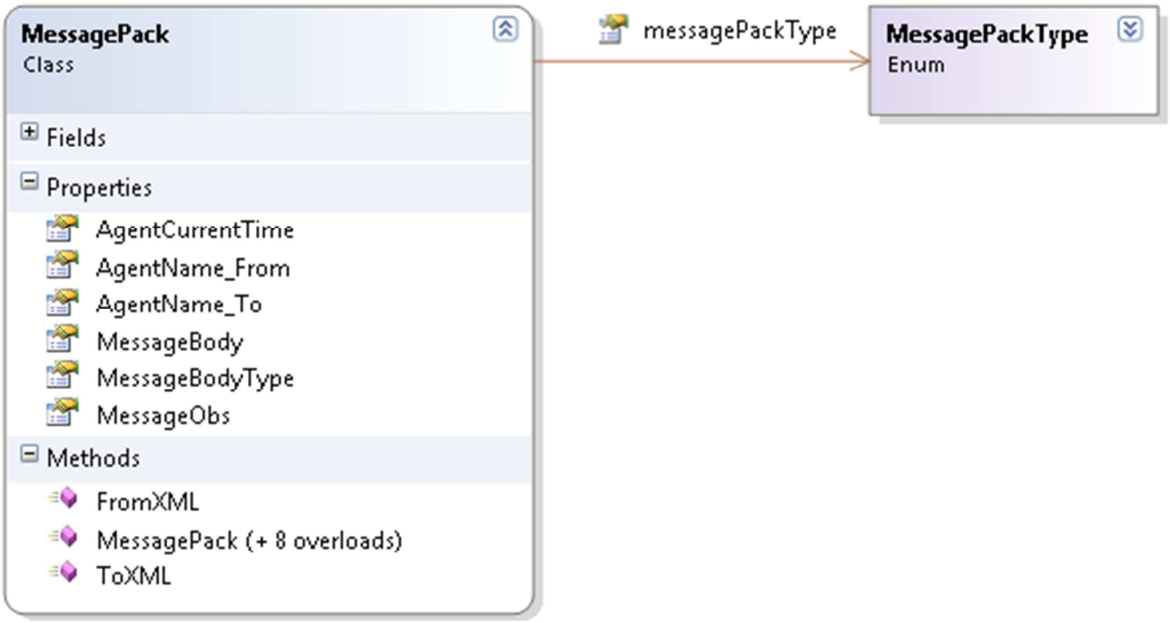
- *Common*

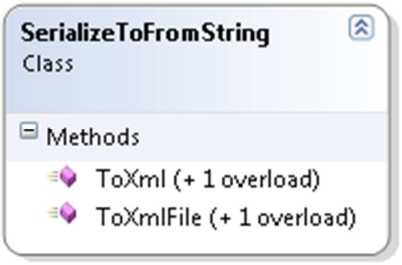
Lista dos serviços disponibilizados pelos Agentes, bem como a lista dos serviços que os Agentes necessitam



- *MessagePack*

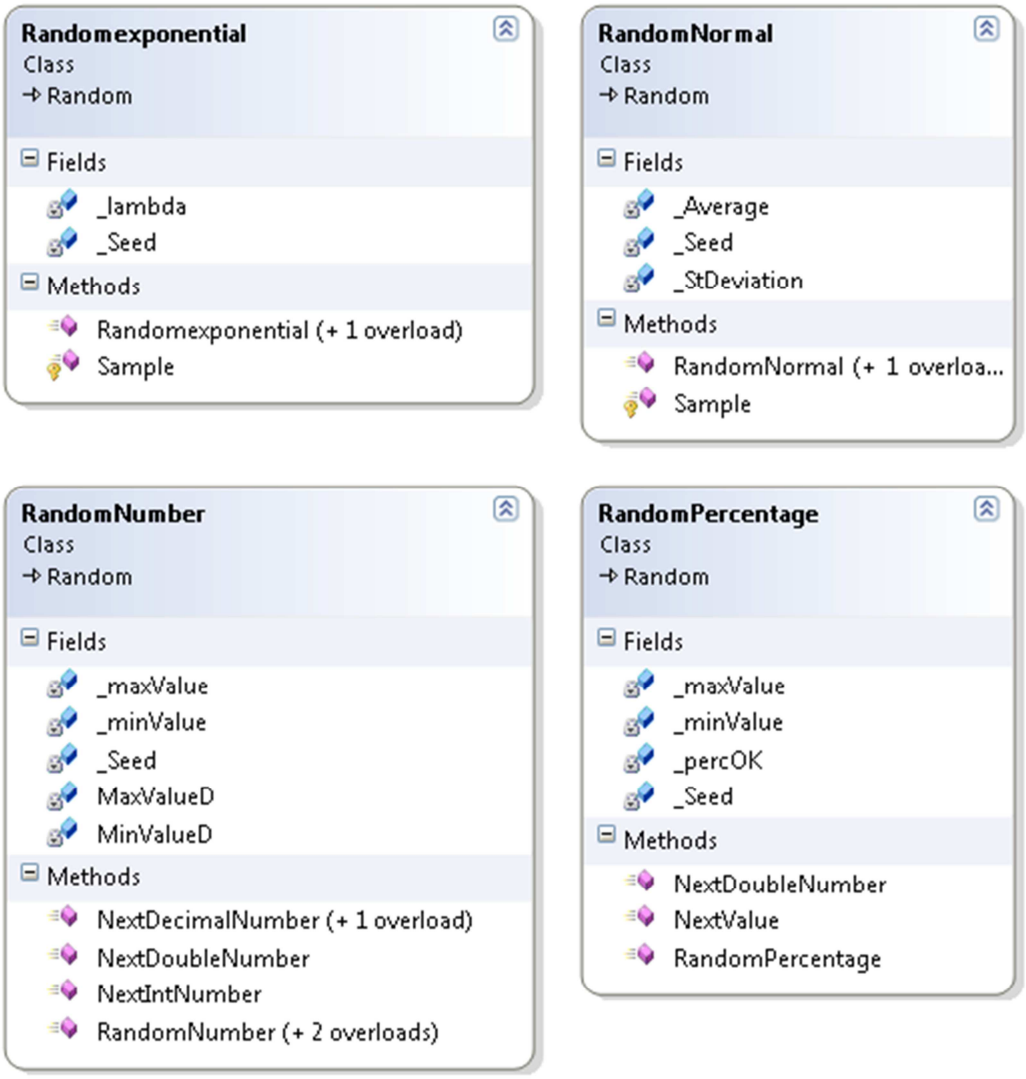
Classes para serializar e desserializar as mensagens em função do seu tipo





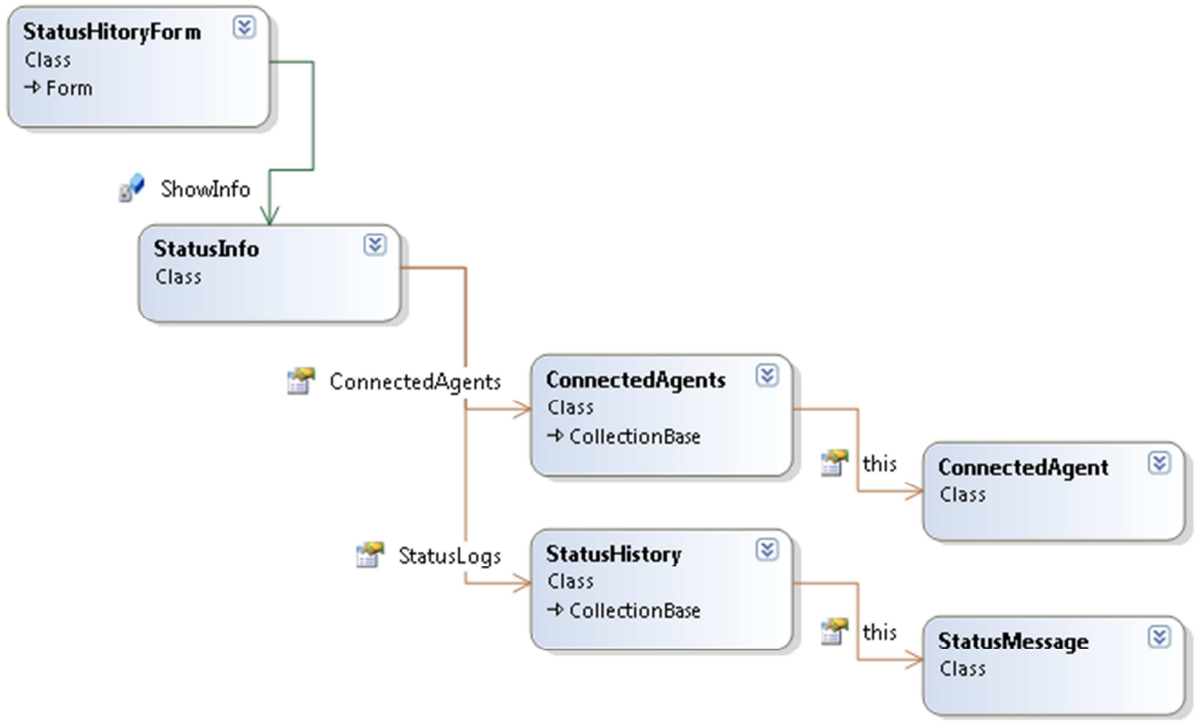
- *Distributions*

Conjunto de classes com métodos para a geração de distribuições estatísticas



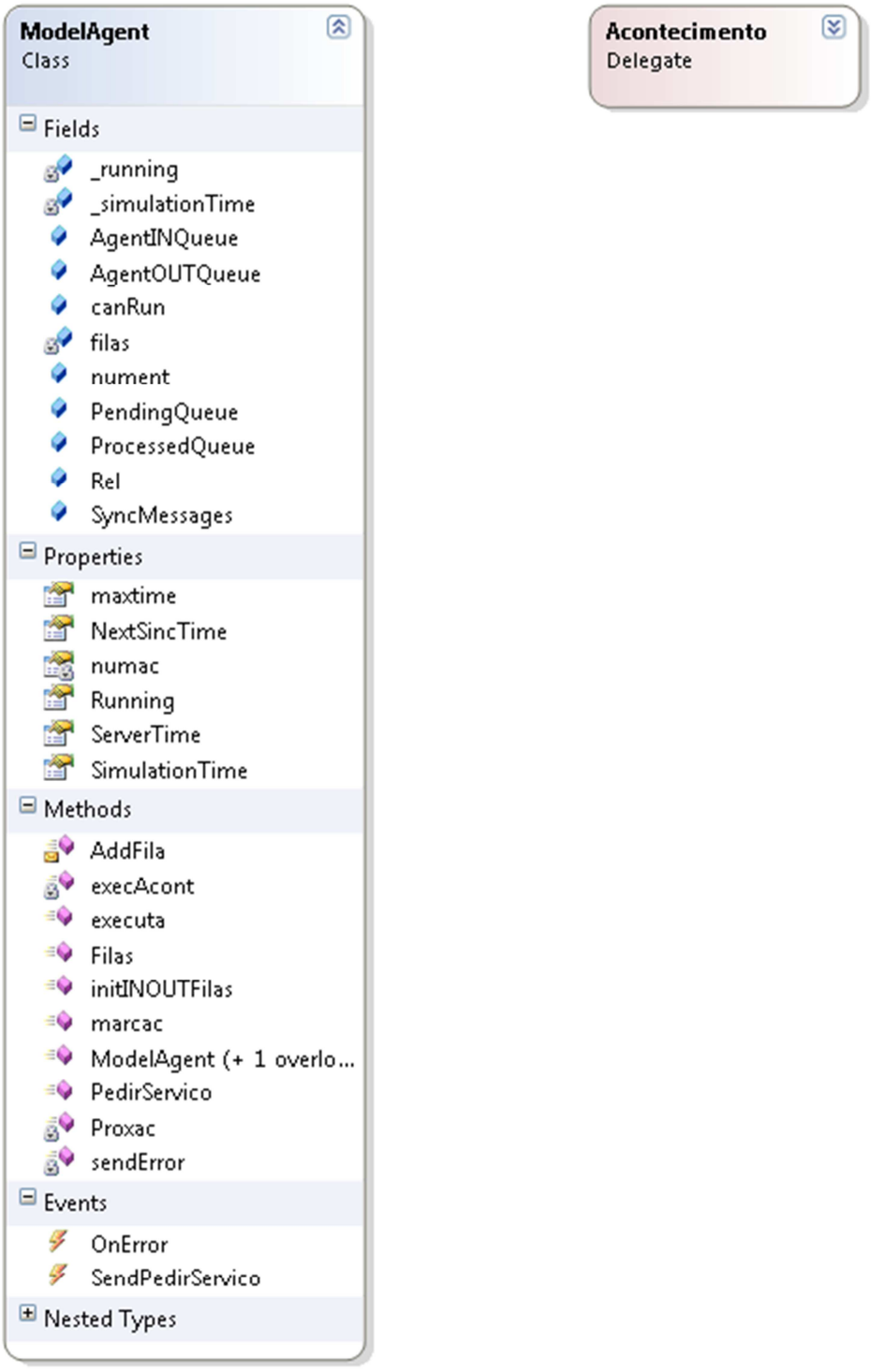
- *StatusLogApp*

Módulo para visualização e registo do estado da simulaçãp



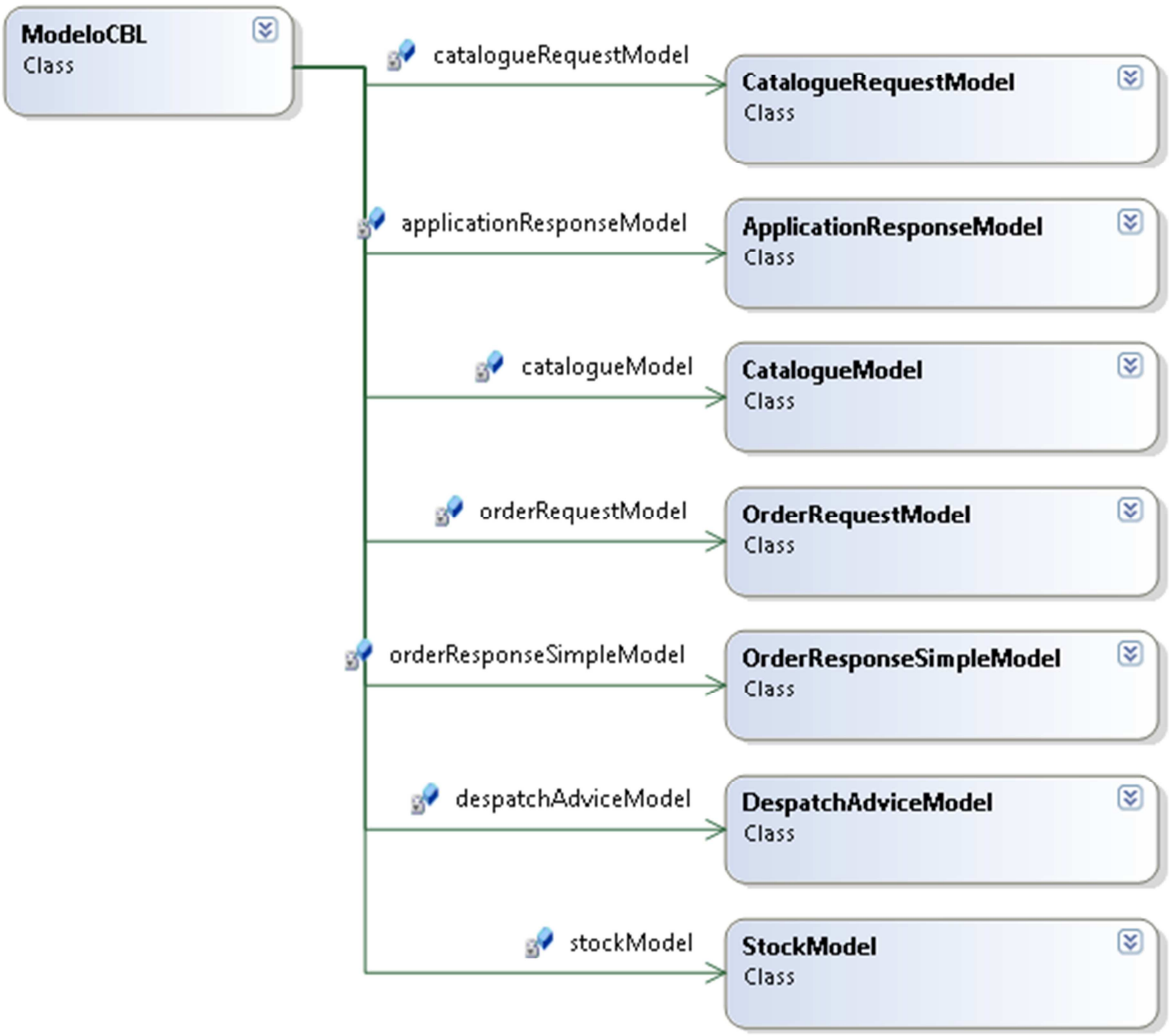
- *CommonAgent*

Biblioteca com um conjunto de classes comuns a vários Agentes



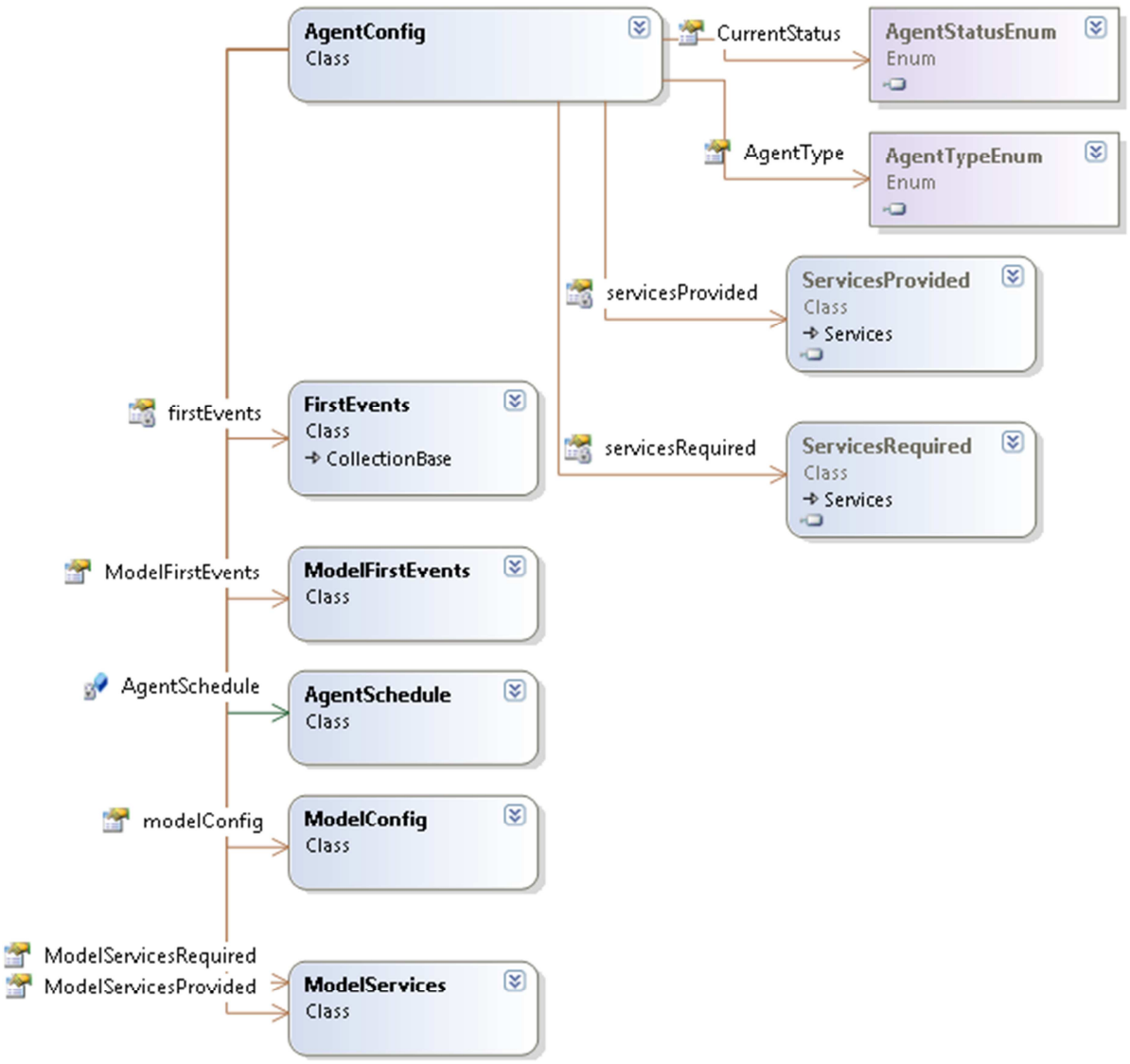
- *ModeloCBL*

Contém um conjunto de classes de interface entre o Agente e a biblioteca OASIS. Transforma a informação do Agente de acordo com o *standard* OASIS



- *ConfigAgentApp*

Classe com informação sobre a configuração do Agente e suas interfaces gráficas, para permitir a sua manipulação



Anexo C. Mensagens XML

Tabela C 1 - *AgentConnectMessage*

```
<?xml version="1.0" ?>
<MessagePack xmlns:xsi="...." xmlns:xsd="....">
   <messagePackType>AgentConnectMessage</messagePackType>
   <AgentName_From>AgentBaseFactory_1</AgentName_From>
   <AgentCurrentTime>11</AgentCurrentTime>
   <MessageObs>agentConnectingMessage</MessageObs>
   <MessageBodyType>AGENTCONNECTINGMESSAGE</MessageBodyType>
   <MessageBody>
      <?xml version="1.0"?>
      <AgentConnectingMessage xmlns:xsi="...." xmlns:xsd="....">
      <AgentType>PauseAll</AgentType>
      <AgentName>AgentBaseFactory_1</AgentName>
      <AgentEndPoint>127.0.0.1:30689</AgentEndPoint>
      <CurrentTime>11</CurrentTime>
      <NextSincTime>31</NextSincTime>
      <State>Unknown</State>
      <EndTime>400</EndTime>
      <AgentGuiId>6e1baa64-dc1e-4e60-b95f-dd619699fef3</AgentGuiId>
      <ServicesProvided>
      <Service> <Name>ORDERREQUEST</Name> </Service>
      <Service> <Name>CATALOGUEREQUEST</Name> </Service>
      <Service> <Name>ORDERRESPONSESIMPLE</Name> </Service>
      <Service> <Name>DESPATCHADVICE</Name> </Service>
      <Service> <Name>APPLICATIONRESPONSETOCATALOGUEREQUEST</Name> </Service>
      </ServicesProvided>
      </AgentConnectingMessage>
   </MessageBody>
</MessagePack>
```

Tabela C 2 - *ServerConfirmConnect*

```
<?xml version="1.0" ?>
<MessagePack xmlns:xsi="....-instance" xmlns:xsd="....">
   <messagePackType>ServerConfirmConnect</messagePackType>
   <AgentCurrentTime>0</AgentCurrentTime>
   <MessageObs>ServerConfirmConnect</MessageObs>
   <MessageBody>
      <?xml version="1.0"?>
      <MessageServerInfo2Client xmlns:xsi="....-instance" xmlns:xsd="....">
      <EndPoint_Server>0.0.0.0:2112</EndPoint_Server>
      <EndPoint_Agent>127.0.0.1:30688</EndPoint_Agent>
      <ServerCurrentTime>0</ServerCurrentTime>
      <NextSincTime>0</NextSincTime>
      </MessageServerInfo2Client>
   </MessageBody>
</MessagePack>
```

Tabela C 3 - *AgentInicialInfoMessage*

```
<?xml version="1.0" ?>
<MessagePack xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <messagePackType>AgentInicialInfoMessage</messagePackType>
   <AgentCurrentTime>0</AgentCurrentTime>
   <MessageObs>agentConnectingMessage</MessageObs>
   <MessageBodyType>AGENTCONNECTINGMESSAGE</MessageBodyType>
   <MessageBody>
      <?xml version="1.0"?>
      <AgentInicialInfoMessage xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <State>Connected</State>
      <CurrentTime>6</CurrentTime>
      <NextSincTime>17</NextSincTime>
      <EndTime>400</EndTime>
      </AgentInicialInfoMessage>
   </MessageBody>
</MessagePack>
```

Tabela C 4 - *ServerStart*

```
<?xml version="1.0" ?>
<MessagePack xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <messagePackType>ServerStart</messagePackType>
   <AgentCurrentTime>0</AgentCurrentTime>
   <MessageObs>ServerStart</MessageObs>
   <MessageBody>
      <?xml version="1.0"?>
      <MessageServerInfo2Client xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <EndPoint_Server>0.0.0.0:2112</EndPoint_Server>
      <EndPoint_Agent>127.0.0.1:30689</EndPoint_Agent>
      <ServerCurrentTime>6</ServerCurrentTime>
      <NextSincTime>17</NextSincTime>
      </MessageServerInfo2Client>
   </MessageBody>
</MessagePack>
```

Tabela C 5 - *AgentFindService*

```
<?xml version="1.0" ?>
<MessagePack xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <messagePackType>AgentFindService</messagePackType>
   <AgentName_From>127.0.0.1:30688</AgentName_From>
   <AgentCurrentTime>0</AgentCurrentTime>
   <MessageBody>
      <?xml version="1.0"?>
      <Service xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <Name>CATALOGUEREQUEST</Name>
      </Service>
   </MessageBody>
</MessagePack>
```

Tabela C 6 - *ServerServiceProviders*

```
<?xml version="1.0" ?>
<MessagePack xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <messagePackType>ServerServiceProviders</messagePackType>
   <AgentName_To>127.0.0.1:30688</AgentName_To>
   <AgentCurrentTime>0</AgentCurrentTime>
   <MessageBody>
      <?xml version="1.0"?>
      <Service xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <Name>CATALOGUEREQUEST</Name>
      <ProvidersOfTheService>
      <ServiceProvider>
      <AgentEndPoint>127.0.0.1:30689</AgentEndPoint>
      <AgentName>AgentBaseFactory_1</AgentName>
      </ServiceProvider>
      </ProvidersOfTheService>
      </Service>
   </MessageBody>
</MessagePack>
```

Tabela C 7 - *AgentInfoMessage*

```
<?xml version="1.0" ?>
<MessagePack xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <messagePackType>AgentInfoMessage</messagePackType>
   <AgentName_From>AgentBaseFactory_1</AgentName_From>
   <AgentCurrentTime>17</AgentCurrentTime>
   <MessageObs>AgentInicialInfoMessage</MessageObs>
   <MessageBodyType>MESSAGEAGENTINFO</MessageBodyType>
   <MessageBody>
      <?xml version="1.0"?>
      <AgentInfoMessage xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <AgentName>AgentBaseFactory_1</AgentName>
      <AgentType>PauseAll</AgentType>
      <ServicesOffer>
      <Service> <Name>ORDERREQUEST</Name> </Service>
      <Service> <Name>CATALOGUEREQUEST</Name>
      </Service> <Service> <Name>ORDERRESPONSESIMPLE</Name> </Service>
      <Service> <Name>DESPATCHADVICE</Name> </Service>
      <Service> <Name>APPLICATIONRESPONSETOCATALOGUEREQUEST</Name> </Service>
      </ServicesOffer>
      <GuiId>6e1baa64-dc1e-4e60-b95f-dd619699fef3</GuiId>
      <Status>Connected</Status>
      <AgentEndPoint>127.0.0.1:35809</AgentEndPoint>
      <ServerEndPoint>127.0.0.1:2112</ServerEndPoint>
      <CurrentTime>17</CurrentTime>
      <NextSincTime>31</NextSincTime>
      <EndTime>400</EndTime>
      </AgentInfoMessage>
   </MessageBody>
</MessagePack>
```

Tabela C 8 - *ServerSyncTime*

```
<?xml version="1.0" ?>
<MessagePack xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <messagePackType>ServerSyncTime</messagePackType>
   <AgentCurrentTime>0</AgentCurrentTime>
   <MessageObs>ServerSyncTime</MessageObs>
   <MessageBody>
      <?xml version="1.0"?>
      <MessageServerInfo2Client xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <EndPoint_Server>0.0.0.0:2112</EndPoint_Server>
      <EndPoint_Agent>127.0.0.1:35809</EndPoint_Agent>
      <ServerCurrentTime>17</ServerCurrentTime>
      <NextSincTime>28</NextSincTime>
      </MessageServerInfo2Client>
   </MessageBody>
</MessagePack>
```

Tabela C 9 - *CatalogueRequestType*

```
<?xml version="1.0" ?>
<MessagePack xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <messagePackType>ToAgent</messagePackType>
   <AgentName_From>Vendedor</AgentName_From>
   <AgentName_To>AgentBaseFactory_1</AgentName_To>
   <AgentCurrentTime>6</AgentCurrentTime>
   <MessageObs>Message CatalogueRequestType</MessageObs>
   <MessageBodyType>CATALOGUEREQUESTTYPE</MessageBodyType>
   <MessageBody>
      <?xml version="1.0"?>
      <CatalogueRequest
xmlns:cbc="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2"
xmlns:ccts="urn:un:unece:uncefact:documentation:2"
xmlns:ext="urn:oasis:names:specification:ubl:schema:xsd:CommonExtensionComponents-2"
xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2"
xmlns="urn:oasis:names:specification:ubl:schema:xsd:CatalogueRequest-2">
      <cbc:UBLVersionID>2.1</cbc:UBLVersionID>
      <cbc:ID>23b6231a-d808-44d0-bead-332fa4d86170</cbc:ID>
      <cbc:Name>CatalogueRequest (1) </cbc:Name>
      <cbc:IssueDate>0001-01-01</cbc:IssueDate>
      <cac:ReceiverParty>
      <cac:PartyIdentification> <cbc:ID>6e1baa64-dc1e-4e60-b95f-dd619699fef3</cbc:ID>
</cac:PartyIdentification>
      <cac:PartyName> <cbc:Name>Vendedor</cbc:Name> </cac:PartyName>
      </cac:ReceiverParty>
      </CatalogueRequest></MessageBody>
</MessagePack>
```

Tabela C 10 - *ApplicationResponseType*

```
<?xml version="1.0" ?>
<MessagePack xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <messagePackType>ToAgent</messagePackType>
   <AgentName_From>AgentBaseFactory_1</AgentName_From>
   <AgentName_To>Vendedor</AgentName_To>
   <AgentCurrentTime>23</AgentCurrentTime>
   <MessageObs>Message ApplicationResponseType</MessageObs>
   <MessageBodyType>APPLICATIONRESPONSETYPE</MessageBodyType>
   <MessageBody>
      <?xml version="1.0"?>
      <ApplicationResponse
xmlns:cbc="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2"
xmlns:ccts="urn:un:unece:uncefact:documentation:2"
xmlns:ext="urn:oasis:names:specification:ubl:schema:xsd:CommonExtensionComponents-2"
xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2"
xmlns="urn:oasis:names:specification:ubl:schema:xsd:ApplicationResponse-2">
      <cbc:UBLVersionID>2.1</cbc:UBLVersionID>
      <cbc:ID schemeID="81664380-3a99-422e-8119-4f205ee23a24">ApplicationResponse
(AgentBaseFactory_1 - 1) </cbc:ID>
      <cbc:IssueDate>2013-03-04</cbc:IssueDate>
      <cac:DocumentResponse>
      <cac:Response> <cbc:ResponseCode>Accepted</cbc:ResponseCode> </cac:Response>
      <cac:DocumentReference>
      <cbc:ID>23b6231a-d808-44d0-bead-332fa4d86170</cbc:ID>
      <cbc:DocumentType>CATALOGUEREQUESTTYPE</cbc:DocumentType>
      </cac:DocumentReference>
      </cac:DocumentResponse>
      </ApplicationResponse>
   </MessageBody>
</MessagePack>
```

Tabela C 11 - *CatalogueType*

```
<?xml version="1.0" ?>
<MessagePack xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <messagePackType>ToAgent</messagePackType>
   <AgentName_From>AgentBaseFactory_1</AgentName_From>
   <AgentName_To>Vendedor</AgentName_To>
   <AgentCurrentTime>23</AgentCurrentTime>
   <MessageObs>Message CatalogueType</MessageObs>
   <MessageBodyType>CATALOGUETYPE</MessageBodyType>
   <MessageBody>
      <?xml version="1.0"?>
      <Catalogue xmlns:cbc="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2"
xmlns:ccts="urn:un:unece:uncefact:documentation:2"
xmlns:ext="urn:oasis:names:specification:ubl:schema:xsd:CommonExtensionComponents-2"
xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2"
xmlns="urn:oasis:names:specification:ubl:schema:xsd:Catalogue-2">
      <cbc:UBLVersionID>2.1</cbc:UBLVersionID>
      <cbc:ID schemeID="07e17af6-65e9-47b8-b718-f04519f3e239">Catalogue (AgentBaseFactory_1 - 1)
</cbc:ID>
      <cbc:IssueDate>2013-03-04</cbc:IssueDate>
      <cbc:Note>catalog 1</cbc:Note>
      <cac:DocumentReference>
      <cbc:ID>23b6231a-d808-44d0-bead-332fa4d86170</cbc:ID>
<cbc:DocumentType>CATALOGUEREQUESTTYPE</cbc:DocumentType>
      </cac:DocumentReference>
      <cac:ProviderParty>
      <cac:PartyIdentification> <cbc:ID>6e1baa64-dc1e-4e60-b95f-dd619699fef3</cbc:ID>
</cac:PartyIdentification>
      <cac:PartyName> <cbc:Name>AgentBaseFactory_1</cbc:Name> </cac:PartyName>
      </cac:ProviderParty>
      <cac:CatalogueLine>
      <cbc:ID>3902585b-7400-4cf8-bcdb-8e608ff37772</cbc:ID>
      <cbc:OrderableUnit>36</cbc:OrderableUnit>
      <cbc:MinimumOrderQuantity>3</cbc:MinimumOrderQuantity>
      <cbc:MaximumOrderQuantity>1</cbc:MaximumOrderQuantity>
      <cac:RequiredItemLocationQuantity>
      <cac:Price> <cbc:PriceAmount currencyID="AFN">2.74</cbc:PriceAmount> <cbc:BaseQuantity
unitCode="36">0.25</cbc:BaseQuantity> </cac:Price>
      </cac:RequiredItemLocationQuantity>
      <cac:Item>
      <cbc:Description>Description 4</cbc:Description>
      <cbc:Name>Name 4</cbc:Name>
      <cac:SellersItemIdentification> <cbc:ID>4</cbc:ID> </cac:SellersItemIdentification>
      </cac:Item>
      </cac:CatalogueLine>
      <cac:CatalogueLine>
      .......
      </cac:CatalogueLine>
      ......
      </Catalogue>
   </MessageBody>
</MessagePack>
```