Carlos Diogo Silva Mendes

Licenciado em Ciências de Engenharia Eletrotécnica e de Computadores

Planeamento Colaborativo de Serviços Customizados

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

Orientador: Doutora Ana Inês da Silva Oliveira, Professora Auxiliar Convidada, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Júri:

Presidente: Doutor André Teixeira Bento Damas Mora - FCT/UNL

Arguente: Doutor Tiago Oliveira Machado de Figueiredo Cardoso - FCT/UNL

Vogal: Doutora Ana Inês da Silva Oliveira - FCT/UNL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Março 2019

Planeamento Colaborativo de Serviços Customizados

Para os meus Avós

Agradecimentos

Nesta secção gostaria de agradecer a todos aqueles que contribuíram para a realização desta dissertação.

Em primeiro lugar, queria agradecer à minha orientadora Professora Doutora Ana Inês Oliveira por me ter dado a oportunidade de conhecer este tema de dissertação e de poder desenvolvê-lo. Pelo seu enorme apoio e disponibilidade em todo o período de realização desta dissertação, sem nunca me ter deixado desamparado ou desmotivado. Gostaria de agradecer também à Doutora Filipa Ferrada pela sua preciosa ajuda e enorme simpatia manifestada.

Expresso a minha gratidão para com os meus colegas de curso João Ramos, Marco Albuquerque, Manuel Faustino, João Bento, Nuno Miguel, Ricardo Ramos, Ricardo Guerreiro e Victor Fernandes, pois sem eles todo o meu percurso académico seria muito mais complicado. Demonstraram-se excelentes amigos, com um grande espirito de companheirismo e sempre presentes para os bons e maus momentos.

Também gostaria de agradecer a todos os meus amigos do "Gaaaaazzzz": João Tiago, João Maia, Ricardo Ramos, Ricardo Fragoso, Gonçalo Santos, Bruno Macedo, Filipe Viseu, Rodrigo Freitas e Eduardo Pedroso, pelo apoio que me deram durante o período de desenvolvimento desta dissertação e por nunca me deixarem desistir dos meus objetivos.

Um agradecimento especial à minha namorada Emanuela Teixeira por todo o apoio, motivação e paciência nesta fase importante da minha vida. Sem ela, este percurso teria sido bastante mais difícil.

Por fim, um grande obrigado à minha família por todo o apoio incondicional, especialmente aos meus pais e irmã pelos ensinamentos e educação que me proporcionaram, que sem eles toda a minha carreira de estudante teria sido impossível de realizar.

Resumo

A Quarta Revolução Industrial, conhecida como Indústria 4.0, tem resultado num considerável aumento do número de dispositivos conectados entre si. Desta forma, devido ao elevado número de conexões geradas, as empresas de produção têm-se deparado com ambientes de manufatura cada vez mais complexos. Esta é considerada uma adversidade, dado que os sistemas de produção podem tornaram-se menos eficientes, menos ágeis e mais dispendiosos.

Atualmente as empresas esforçam-se constantemente por oferecer melhores produtos e serviços, estando em contínua inovação. Um dos fatores para que tal seja possível, sendo que os métodos tradicionais não se mostram suficientes para cumprir certos requisitos, é a definição de novas estratégias de manufatura. Assim, conceitos como "Personalização" e "Customização" são de importância relevante na atual competitividade de mercado.

Nesta dissertação, é proposto um sistema para o apoio à colaboração entre dispositivos visando diferenciar os sistemas tradicionais e abordar o contexto de um ambiente de produção descentralizado e que possibilita a customização de produtos.

Foi desenvolvido um protótipo de colaboração multiagente tendo por base conceitos das Redes Colaborativas. O protótipo destina-se, particularmente, a ambientes com um tipo de produção discreta, permitindo a abstração de cada equipamento/máquina presente no ambiente, através de agentes que juntos constituem um sistema multiagente. Este sistema possibilita a colaboração entre agentes baseada em conceitos de Redes Colaborativas, suportando também a realização de pedidos customizados.

Palavras-chave: Sistema Multiagente, *Smart Factory*, Redes Colaborativas, Colaboração, Manufatura*, Smart Connectivity.*

Abstract

The Fourth Industrial Revolution, known as Industry 4.0, has resulted in an enormous increasing of devices connected to each other. Therefore, due to the high number of generated connections, production companies have been facing more complex production environments. This can be considered an adversity, since production systems became less efficient, less agile and more expensive.

Nowadays, companies are constantly striving to offer better products and services, being in constant innovation. Since the traditional methods are not sufficient to accomplish certain requirements, one of the factors that make constant innovation possible is the definition of new manufacturing strategies. Moreover, concepts such as "Personalization" and "Customization" are relevant to consider in the current market competitiveness.

In this dissertation, a system to support collaboration among devices is proposed with the aim to contrast with traditional systems and address the context of decentralized production environments that allow product customization.

A multi-agent collaboration prototype was developed based on Collaborative Networks concepts. The prototype is particularly intended to discrete production type environments, allowing an abstraction of each equipment/machine present in the environment, through a set of agents that constitute a multi-agent system. The system enables collaboration between agents based on Collaborative Networks concepts, also supporting customized requests.

Keywords: Multi-agent Systems, Smart Factory, Collaborative Networks, Collaboration, Manufactory, Smart Connectivity.

Índice de Matérias

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

ACL - Agent Communication Language

AI - Agente Inteligente

BDI - Belief-Desire-Intention

CAs - Communicative Acts

COM-MTDP - COMmunicative Multiagent Team Decision Problem

CPS - Cyber-Physical Systems

FIPA - Foundation for IntelligentPhysical Agents

IDE - Integrated Development Environment

IoT - Internet of Things

JADE - Java Agent Development Framework

KIF - Knowledge Interchange Format

KQML - Knowledge and Query Manipulation Language

MDP - Markov Decision Process

PVC - Professional Virtual Communities

R&D - Research and Development

ROs - Research Organisations

SMA - Sistema Multiagente

TDSM - Tomada de Decisão num Sistema Multiagente

VBE - Virtual organizations Breeding Environments

VE - Virtual Enterprise

VL - Virtual Laboratory

VO - Virtual Organization

Capítulo 1

1. Introdução

1.1 Domínio do Problema e Motivação

Desde a Quarta revolução Industrial, conhecida como Indústria 4.0, o número de dispositivos conectados entre si tem aumentado significativamente provocando, consequentemente, um elevado número de conexões. Com isto, as empresas de produção enfrentam ambientes que ficam cada vez mais complexos, sendo necessário recorrer a métodos que sejam capazes de realizar uma "*Smart Connectivity*" e uma distribuição da carga computacional para controladores locais. É então que surgem os Sistemas Multiagente (SMA) de forma a realizar um controlo distribuído. Estes, no âmbito de "*Smart Factories*", trazem uma maior flexibilidade, rapidez, produtividade e qualidade no processo de produção (Rüßmann et al., 2015; Xie & Liu, 2017; Gubbi et al., 2013).

As "*Smart Factories*" representam processos de produção altamente reconfiguráveis e adaptativos baseados na interação de entidades autónomas, estas que são conhecidas como *Cyber-Physical Systems* (CPS). Os Sistemas Multiagente podem ter um papel chave neste contexto uma vez que, usando as características de dado ambiente, podem cumprir novos requisitos de produção e resolver falhas inesperadas do sistema sem ter que ser alterada a arquitetura do mesmo (Kannengiesser & Müller, 2013).

Um dos fatores principais num Sistema Multiagente é a comunicação entre agentes e a forma como eles colaboram para resolver uma tarefa em conjunto. Este, entre outros fatores, é realmente vantajoso porque através de uma colaboração é possível melhorar o funcionamento de um sistema, como, por exemplo, o tempo de execução de uma tarefa. Esta colaboração entre agentes, no fundo, pode ser comparada com as redes colaborativas, onde diferentes entidades (por exemplo empresas) diferentes e autónomas criam alianças de forma a partilharem recursos e colaborarem para um mesmo objetivo, influenciando o seu sucesso individual. Neste caso, pretende-se aplicar esse conceito a um SMA para que sejam realizadas tarefas de forma colaborativa que contribuam para uma melhoria no sistema (Tošić & Ordonez, 2012; Bititci et al., 2004; Camarinha-Matos & Afsarmanesh, 2008).

1.2 Objetivos e Contribuições

Existirá uma maior dificuldade em controlar eficientemente um sistema que tenha um maior número de dispositivos. De forma a garantir a alta eficiência e agilidade de um sistema, os SMA poderão ser usados para esse objetivo. Estes, que têm origem na inteligência artificial distribuída, têm obtido muita atenção devido às grandes vantagens que trazem para um sistema, não só devido às semelhanças de raciocínio de um agente comparativamente ao cérebro de um ser humano mas, também, à inteligência social que um agente pode possuir (Brandão et al., 2013; Jiang et al., 2016).

O principal objetivo deste trabalho assenta no apoio à colaboração entre dispositivos, utilizando para tal um Sistema Multiagente colaborativo. Será então necessário, para atingir este objetivo, a escolha do sistema mais adequado ao ambiente de aplicação, onde estão presentes os tipos de agente e a arquitetura do sistema. Também será necessário o uso de uma linguagem de comunicação entre agentes e métodos de coordenação e cooperação baseados em conceitos estudados relativamente às Redes Colaborativas. Finalmente, depois do desenvolvimento do SMA, foi criado um cenário onde este será validado.

1.3 Estrutura da Dissertação

Esta dissertação é composta por sete capítulos: Introdução, Estado da Arte, Contexto de Desenvolvimento, Arquitetura do Sistema, Implementação do Sistema, Validação e Conclusões e Futuros Desenvolvimentos.

O primeiro e presente capítulo, Introdução, dá uma breve introdução aos objetivos e contribuições desta dissertação, à contextualização ao tema e às principais motivações para a realização desta dissertação.

No Estado da Arte são abordados conceitos fundamentais relacionados com os sistemas multiagente, com as redes colaborativas e, também, com os tipos de produção de manufatura.

No capítulo do Contexto de Desenvolvimento, é realizado um esclarecimento da correlação entre os diferentes tópicos abordados no capitulo do Estado de Arte e quais as funcionalidades pretendidas no desenvolvimento deste trabalho.

O quarto capítulo, Arquitetura do Sistema, apresenta o modelo conceitual do sistema, incluindo os seus conceitos básicos e interações existentes entre as diferentes entidades inseridas no mesmo. São apresentados, também, os requisitos funcionais e não funcionais do sistema.

O capítulo Implementação do Sistema, ilustra a implementação da arquitetura caracterizada no capítulo antecedente a este. Evidencia todos os passos realizados nesta etapa, desde a e escolha das plataformas e ferramentas de desenvolvimento, até à descrição de todas as funcionalidades principais e secundárias do sistema.

No capítulo Validação são apresentados dois casos de teste e realizada a simulação destes no sistema implementado.

Por fim, o capitulo Conclusões e Trabalhos Futuros apresenta uma visão geral de todo o trabalho realizado e a sugestão de futuros desenvolvimentos.

Capítulo 2

2. Estado de Arte

Este capítulo tenta dar uma visão geral acerca de conceitos relacionados com redes colaborativas, sistemas multiagente e tipos de manufatura. Estes são considerados os principais temas a considerar neste trabalho, uma vez que é pretendida a implementação de um sistema multiagente, em que a ideologia de colaboração seja baseada em redes colaborativas aplicada a num contexto de manufatura, onde seja possível incluir conceitos de reconfigurabilidade e flexibilidade, com a possibilidade de customização, de forma a ultrapassar o conceito de padronização na criação de um produto. Um sistema de manufatura que permita a personalização, nos dias de hoje, tornou-se algo imprescindível para manter os níveis de competitividade do mercado global.

No final deste capítulo, são apresentadas possíveis contribuições que um sistema multiagente com estas características pode ter num sistema de manufatura.

2.1 Redes Colaborativas

Com o crescimento exponencial do número de dispositivos utilizados nos diversos contextos, tem-se observado uma complexidade cada vez maior em relação à conectividade entre eles. Consequentemente, isto torna os ambientes colaborativos cada vez mais complexos (Camarinha-Matos & Afsarmanesh, 2018).

Uma rede colaborativa é definida como uma rede em que os membros, geograficamente distribuídos ou não, cooperam entre si para alcançar certos objetivos comuns ou compatíveis através de redes de computadores (Islam et al., 2016; Camarinha-Matos & Afsarmanesh, 2005).

No âmbito deste trabalho, pretende-se um planeamento colaborativo entre dispositivos interligados num ambiente de manufatura. Como tal, é fundamental abordar o conceito de redes colaborativas para que se perceba qual o tipo de colaboração entre dispositivos que terá que ser integrado neste trabalho.

A adição do conceito de uma rede colaborativa a um SMA, foi feita para que os agentes presentes nesse sistema tenham uma maior sensibilidade relativamente ao oportunismo face ao

bem-estar do sistema. Com isto, espera-se um aumento da eficiência num ambiente de manufatura onde, conjuntamente, estão presentes estes dois conceitos.

2.1.1 Tipos de Colaboração

Foi realizado um estudo (Nieto & Santamaría, 2007) a partir de uma variedade de organizações na área da manufatura Espanhola que operam em todos os setores industriais. Este estudo tem como objetivo perceber que tipo de parceiro apresenta melhores benefícios para uma organização. Primeiro, foram criadas variáveis para os diferentes tipos de colaboração, diferenciada de cada tipo de parceiro: colaboração exclusivamente com ROs (*Research Organisations)*, com clientes, com fornecedores, com competidores e colaboração com mais do que um destes parceiros. De seguida, foram criadas variáveis de controlo para as vendas, para a intensidade de R&D (Research and Development), atividade de exportação, setores dominantes dos fornecedores, setores com fornecedores especializados e para os setores baseados em ciência.

Os resultados deste estudo mostram que a colaboração com os fornecedores é a que revela mais benefícios relativamente à inovação de produto e que ajuda a trazer os produtos para o mercado mais rapidamente. A colaboração com competidores é a menos produtiva, apesar de a colaboração com um competidor parecer bastante apelativa pela possibilidade de troca de informação acerca dos mesmos problemas e a resolução desses a partir da colaboração de ambos, acontece que o medo e a falta de confiança, dado o alto risco de comportamentos oportunistas, faz com que os competidores sejam vistos como um mau parceiro para colaboração.

Por outro lado, num outro estudo (Chen et al., 2016) são discutidas diferentes formas de colaboração e qual a melhor para a sua viabilidade. Foi realizada uma pesquisa baseada na experiência de diversas organizações em colaborações para perceber qual o melhor tipo de colaboração. Dentro destes tipos de colaboração estão, por exemplo, divisão de partes, consórcio de R&D, contrato de licença, etc. Os principais fatores para a escolha do tipo de colaboração são o custo de transação, visão baseada em recursos e os fatores sociológicos. Neste artigo são também estabelecidas três hipóteses para a escolha de um tipo colaboração:

- 1 - Características de diferença do recurso agrupado para a colaboração levará a preferências diferentes no tipo de colaboração;
- 2 - A relação inicial entre parceiros que vai influenciar o tipo de colaboração escolhida; e
- 3 - Diferentes tipos de tarefas de colaboração podem levar a uma escolha do tipo de colaboração.

Parte da análise desta pesquisa, concluiu que quanto mais concentrada uma tarefa de colaboração é, maior é a probabilidade de parceiros quererem colaborar com um consórcio de R&D. Conclui-se, também, que quando a divisão de trabalho pelos diferentes parceiros é menor, a probabilidade de colaborar com um consórcio é, mais uma vez, maior.

2.1.2 Motivos e Benefícios da Colaboração

A colaboração tem sido uma metodologia adotada por diferentes identidades que têm um objetivo em comum em determinado ambiente. Tem sido relatado por diferentes identidades que a colaboração expõe benefícios que se podem considerar significantes. Alguns motivos e benefícios da colaboração interempresarial que são considerandos relevantes (Bititci et al., 2004; Camarinha-Matos et al., 2007; Oliveira & Camarinha-Matos, 2015)**:**

- Partilha e redução do custo do desenvolvimento do produto;
- Redução do risco de falha no desenvolvimento do produto;
- Aumento da qualidade do produto;
- Aumento do conhecimento e competências;
- Rápido acesso aos mercados; e
- Aumento da quota de mercado.

2.1.3 Classes de Redes Colaborativas

Como ilustrado na Figura 2.1, as redes colaborativas podem ser distinguidas em duas classes principais: redes estratégicas de longo termo e redes orientadas por objetivos (Camarinha-Matos & Afsarmanesh, 2008).

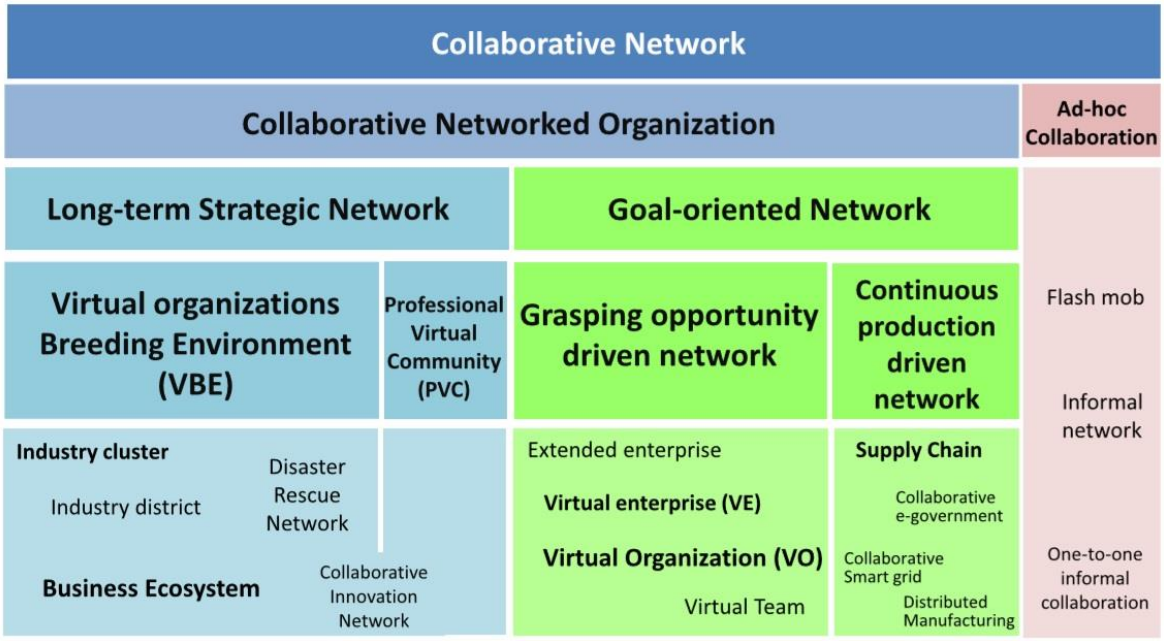


Figura 2.1 - Taxonomia das redes colaborativas (Camarinha-Matos & Afsarmanesh, 2018).

Redes Estratégicas de Longo Termo

Nesta classe estão presentes as *Virtual organizations Breeding Environments* (VBE), que são definidas como ambientes gerais onde estão presentes todas as organizações com potencial e vontade de colaboração e cooperação, para estabelecimento de acordos, quando surge uma oportunidade de negócio. Está incluído numa organização deste tipo o caso de ecossistemas de negócios, indústrias de Cluster, a indústria Distrital, a Rede de resgate a desastres e redes de inovação colaborativa (Camarinha-Matos & Afsarmanesh, 2018; Camarinha-Matos & Afsarmanesh, 2005).

Também fazem parte desta classe as *Professional Virtual Communities* (PVC) que são definidas como sistemas sociais que usam tecnologias mediadoras para troca de informação, ideias e problemas entre diferentes indivíduos. Estão incluídas nestas comunidades, tal como numa VBE, as redes de inovação colaborativa (Camarinha-Matos & Afsarmanesh, 2018).

Esta classe de redes tem o objetivo de criar fortes ligações e desenvolver fortes capacidades de consórcio entre membros para que, mais tarde, quando surge uma oportunidade de negócio já se saibam quais as ligações mais corretas a fazer na criação de uma organização colaborativa, e tudo se possa processar de uma forma rápida e consistente (Camarinha-Matos & Afsarmanesh, 2018).

Redes Orientadas por Objetivos

Nestas redes estão incluídas as *Virtual Organizations* (VO) e as *Supply Chain.* As VO's são definidas como uma organização onde os diferentes membros partilham recursos e competências de forma a chegarem a um objetivo em comum. Os diferentes membros podem ter características diferentes, desde: empresas, organizações não-governamentais, profissionais independentes, etc. Estas são redes colaborativas que são criadas de modo a aproveitar da melhor maneira oportunidades de negócio (Camarinha-Matos & Afsarmanesh, 2005; Camarinha-Matos et al., 2008).

Existem também outros tipos de Redes Colaborativas, de seguida destacam-se algumas:

Virtual Enterprise (VE) – Consiste numa aliança temporária em que são trocados recursos e competências entre si de forma a prevalecer perante uma oportunidade de negócio. Esta é um caso particular de uma VO, pois são apenas consideradas empresas (Camarinha-Matos & Afsarmanesh, 2005).

Virtual Laboratory (VL) – É um ambiente com identidades de diferentes tipos e que resolvem problemas de diferentes maneiras. Neste ambiente trabalham juntos diferentes investigadores que se encontram geograficamente distribuídos (Camarinha-Matos & Afsarmanesh, 2005).

2.1.4 Ciclo de vida de uma VO no contexto de VBE

Uma VO tem um ciclo de vida que passa pelas seguintes etapas principais: criação, operação, evolução e dissolução (Camarinha-Matos et al., 2005, Camarinha-Matos et al., 2007).

A criação de uma VO, primeiro passo no seu ciclo de vida, passa pela identificação de uma oportunidade de colaboração por parte de uma ou mais entidades do ambiente onde estão presentes diferentes entidades com potenciais de colaboração entre si. Esta identificação é o processo que desencadeia a formação de uma nova VO, onde estas entidades comunicam entre si de modo a estabelecer um consenso de quais são os objetivos a realizar e o que cada entidade pode disponibilizar e beneficiar dela (Camarinha-Matos et al., 2007).

A fase de operação de uma VO passa pela colaboração entre os diferentes membros da VO a fim de corresponder aos objetivos que têm em comum, ou seja, da execução do plano através da realização de tarefas entre os diferentes membros.

A etapa "dissolução da VO" acontece quando os objetivos em comum estão todos cumpridos ou, por outro lado, se a VO já não tem recursos suficientes para corresponder a mais objetivos (Camarinha-matos & DEE Group Author, 2011).

Por fim, a etapa de evolução de uma VO poderá acontecer caso todos os objetivos impostos inicialmente sejam realizados e a organização pretenda continuar ativa de modo a corresponder a novos objetivos.

Na Figura 2.2 pode-se observar a criação de uma VO em diferentes contextos.

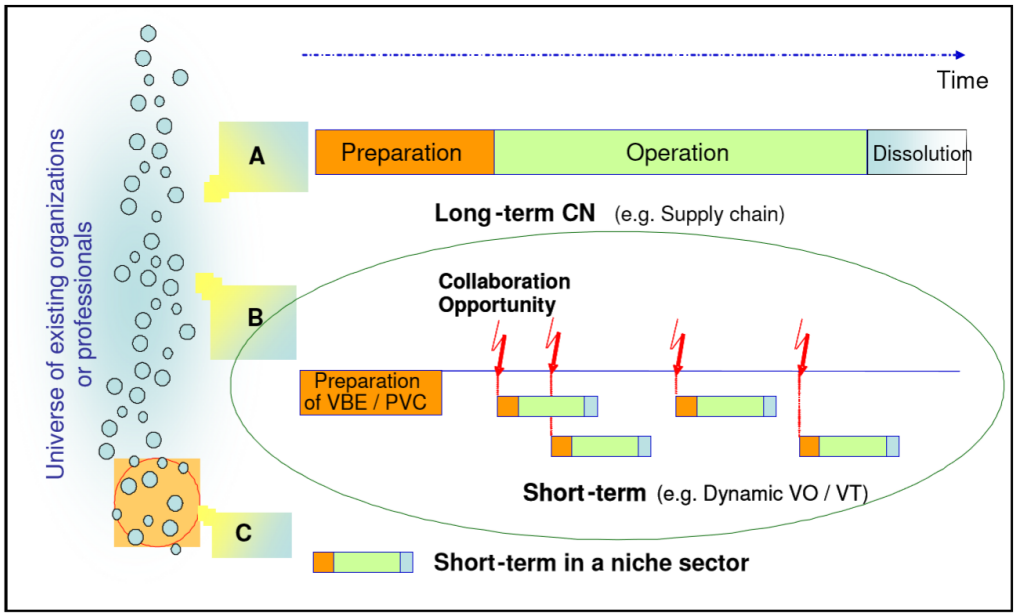


Figura 2.2 - Criação de VO em diferentes contextos (Camarinha-Matos et al., 2008).

O caso A da Figura acima representa o ciclo de vida de uma colaboração a longo prazo, onde se podem observar as diferentes etapas descritas anteriormente (Camarinha-Matos et al., 2008).

No caso C, observa-se um ciclo de vida mais curto. Este refere-se ao ciclo de vida de colaborações com um curto prazo de duração, que consequentemente têm uma curta janela de tempo para tomar uma decisão em relação à oportunidade que surge. Este tipo de colaborações visa criar consórcios de modo a corresponder rapidamente a uma oportunidade (Camarinha-Matos et al., 2008).

Um tipo de VO que se adequa ao caso C é, por exemplo, uma *"Dynamic Virtual Organization"* que é descrita como uma VO, com um curto ciclo de vida, que é criada de maneira a corresponder a oportunidades competitivas, ou seja, que requerem uma rápida resposta, e a sua etapa de dissolução acontece quando o propósito para a sua criação estiver completo (Camarinha-Matos et al., 2008; Oliveira & Camarinha-Matos, 2008; Wang et al., 2006). Este é o tipo de colaboração idealizado para a implementação do sistema pretendido nesta dissertação.

2.1.5 Consórcios Colaborativos

Quando existe uma possível formação de VO, passa-se sempre por uma fase de negociação que é bastante importante para a criação dessa VO (Oliveira & Camarinha-Matos, 2010). Esta negociação pode assegurar o correto funcionamento da VO no seu processo de operação (Oliveira & Camarinha-Matos, 2012). Esta fase de criação é maioritariamente representada por um acordo de consórcio para a VO, onde são pactuados todos os direitos e deveres de todos os parceiros envolvidos na mesma (Oliveira & Camarinha-Matos, 2015; Oliveira et al., 2010). Um consórcio, entre possíveis parceiros, pode ser determinante na tomada de decisão para a colaboração, uma vez que é decidido o que cada parte está disposta a oferecer e aceitar (Chen et al., 2016; Camarinha-Matos, 2013).

Um dos objetivos deste trabalho será a criação de consórcios para o SMA, onde dois ou mais agentes podem criar uma VO através de um contrato de consórcio. Este serve para os agentes negociarem a colaboração entre eles, sendo a VO estabelecida quando todas as propostas, de todos os envolvidos na colaboração, for consensual.

2.2 Sistema Multiagente (SMA)

Um SMA é composto por um grupo de agentes inteligentes que comunicam e colaboram entre si num dado ambiente, de forma a completarem tarefas conforme a sua perceção ao ambiente (zona de influência) e da maneira mais eficiente possível (Figura 2.3) (Reis, 2003).

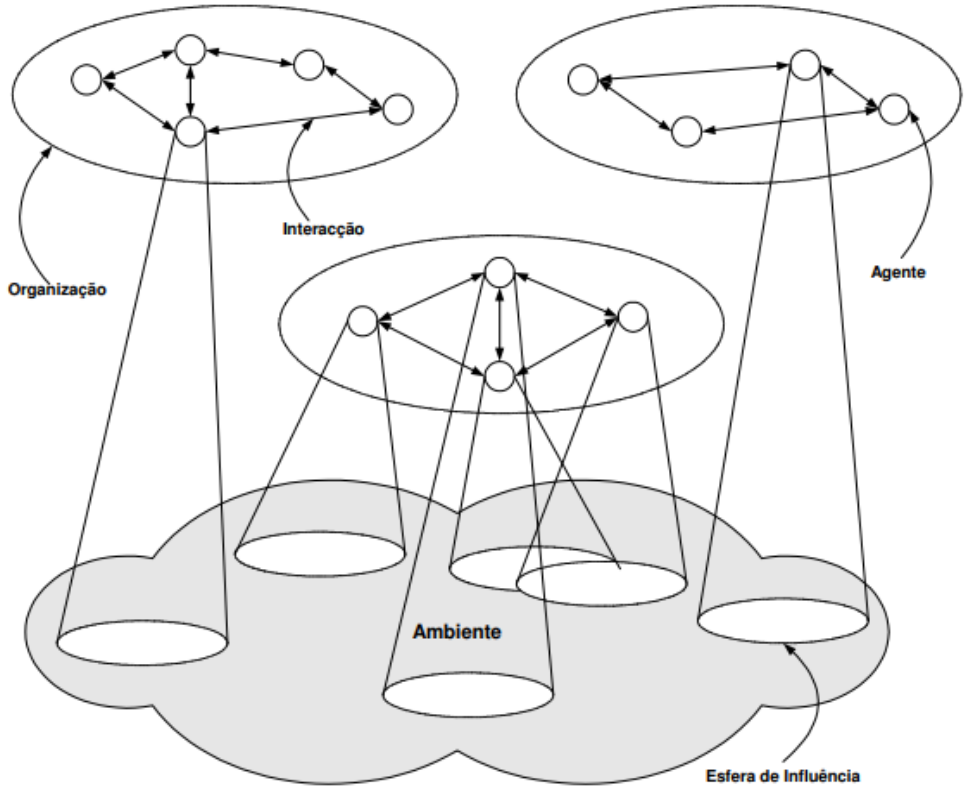


Figura 2.3 - Estrutura de um SMA (Reis, 2003).

2.2.1 Agente Inteligente (AI)

Um agente é um mecanismo computacional *(software)* que é autónomo o adotar as suas decisões, a partir da informação que lhe é fornecida nesse dado ambiente. Esta informação pode ser fornecida a partir de dispositivos (sensores de aquisição de dados) ou, também, de agentes vizinhos presentes no seu sistema (Panait & Luke, 2005).

Estas são algumas características que distinguem um AI de qualquer outro sistema de *software* (McArthur & Davidson, 2005):

1. **Reatividade:** Um agente é capaz de lidar com mudanças de estado num ambiente. Toma decisões baseado nas alterações que ocorrem no ambiente e na tarefa que tem a desempenhar no seu sistema.

2. **Proatividade:** Um agente, sendo autónomo, assume responsabilidades pelas próprias ações que toma em função da sua reação ao ambiente. Esta, é uma característica chave num agente pois, devido às mudanças no seu ambiente, ele é capaz de tomar decisões por eles para um correto funcionamento do sistema ou até para instituir-lhe melhorias.

3. **Sociabilidade:** Esta é a característica que destaca um agente relativamente à sua inteligência artificial. Este, não só tem semelhanças na maneira de raciocinar de um ser humano, como também, nas suas capacidades sociais. É capaz de interagir, negociar e resolver problemas em conjunto com outros agentes, com o intuito de realizar tarefas com sucesso, de maneira eficiente e que traga benefícios para ambos os agentes. Com isto, poderá dizer-se que a comunicação é um fator chave num SMA.

Assim, dependendo da forma como é construída a arquitetura de um agente e a forma como os diferentes componentes se interligam, podem diferenciar-se tipos de agentes.

Um agente pode ser do tipo reativo em que toma as suas decisões, unicamente, a partir da informação que recebe do ambiente no momento e não no seu historial de acontecimentos do ambiente, portanto atua no ambiente apenas em função da última informação que recebeu do mesmo. Existem também os agentes lógicos que, tal como o nome indica, as suas decisões são tomadas através de cálculos lógicos. Por fim, existem os agentes BDI (Belief-Desire-Intention) que baseiam as suas decisões no seu estado atual de informação, atuando no ambiente segundo aquilo que mais acreditam que seja a decisão certa (Xie & Liu, 2017).

2.2.2 Ambientes

Um agente recebe informações e atua num determinado ambiente. As características do ambiente em que se encontra são importantes para projetar o tipo de agente a utilizar e os seus diferentes modos de operação. Assim, é preciso um estudo prévio do tipo de ambiente para perceber qual o tipo de papel o agente pode ter nesse ambiente, sendo que quanto mais acessível um ambiente for, mais fácil será a escolha de uma arquitetura e a forma de atuação e tomada de decisões dos agentes.

Cada ambiente é diferenciado pelas seguintes características (Reis, 2003):

- **Acessibilidade:** Um agente estando presente num ambiente acessível consegue, de uma forma precisa, obter facilmente informação através de sensores relativos a todo o ambiente;

- **Determinação:** As ações de um ambiente determinístico são sempre bastante explícitas, onde cada ação vai ter um resultado que, à partida, já é esperado nesse ambiente;

➔ **Dinâmica:** Num ambiente com esta característica, os agentes presentes nele agem sincronicamente, realizando tarefas, de modo a atingirem objetivos em comum; e

➔ **Discreto:** Um ambiente discreto é aquele que apenas tem um número de estados limitados e distinguíveis uns dos outros. Deste modo é facilitada a perceção dos agentes ao ambiente.

2.2.3 Comunicação e Negociação entre agentes

Para que os agentes num SMA possam negociar entre si são necessários protocolos de comunicação e protocolos de negociação. Estes protocolos permitem que os agentes colaborem entre si de forma a completarem tarefas do sistema. Uma vez que diferentes agentes têm diferentes perspetivas do estado do mundo, a negociação é um fator chave uma vez que pode ser utilizada para chegar a um consenso de qual é a melhor decisão a tomar para o bom funcionamento do sistema (Tošić & Ordonez, 2012).

A comunicação entre agentes ocorre através de protocolos de transporte de mensagens, sendo que existem diversos tipos destes protocolos, onde o HTTP é um exemplo (Bravo et al., 2015).

Cada plataforma de modelação de agentes tem a sua linguagem de comunicação entre agentes denominada como *"Agent Communication Language (ACL)".* Uma ACL deve ser inequívoca, ou seja não deve ter mais que uma interpretação para que a comunicação entre agentes seja feita de forma clara. Também deve ter uma semântica concisa onde existe uma troca de mensagens breve mas de uma forma altamente compreensiva. A primeira linguagem a ser padronizada foi a KQML *(*Knowledge and Query Manipulation Language*)*, que inclui a compreensão do conteúdo presente numa mensagem. Existem outras ACL's como a *KIF* (*Knowledge Interchange Format*) e a FIPA (*Foundation for IntelligentPhysical Agents)*, onde se destaca a última pela sua simplicidade e por ter apenas 20 tipos de interpretação de uma mensagem. Um dos tipos de interpretação de uma mensagem é, por exemplo, um "*request"* que consiste num pedido a um agente para realizar uma determinada ação(Reis, 2003; Bravo et al., 2015; Yu et al., 2010).

Existem assim diferentes formas de comunicar num SMA:

- **Comunicação Direta:** A comunicação é feita através de trocas de mensagens diretamente de um agente para outro. Ou seja, um agente comunica com outro diretamente sem existir qualquer intermediário. São ligações ponto-a-ponto, como exemplificado na figura 2.4 (Reis, 2003; Silva, 2003).

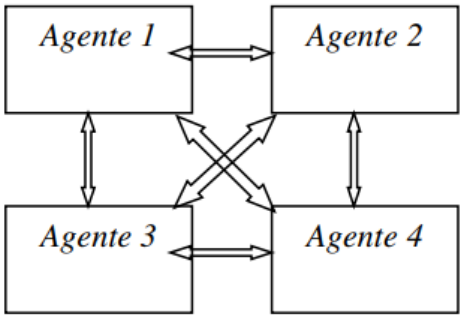


Figura 2.4-Comunicação Direta (Reis, 2003).

- **Comunicação Assistida:** Neste tipo de comunicação a troca de mensagens é feita por um agente intermediário denominado agente "facilitador". Este tipo de comunicação é utilizada quando o número de agentes num SMA é muito elevado, de modo a reduzir a complexidade da troca de mensagens no sistema, bem como trocas de mensagens cruzadas. No exemplo da figura 2.5 em que os agentes 4 e 5 são, respetivamente, os agentes facilitadores do grupo dos agentes 1, 2 e 3 e do grupo dos agentes 6, 7 e 8) (Reis, 2003; Silva, 2003).

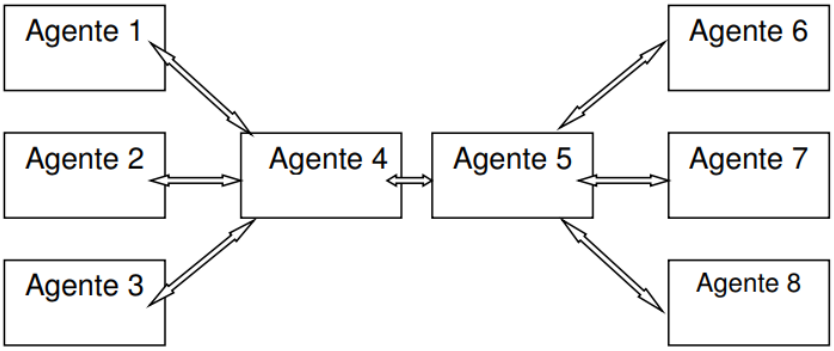


Figura 2.5-Comunicação Assistida (Reis, 2003).

- **Comunicação Broadcast:** Quando, a partir de um agente, uma mensagem é enviada para todos os agentes vizinhos com o intuito de enviar a todos os agentes à sua volta ou quando não sabe quem é o agente destinatário (Silva, 2003).

Depois de implementados métodos de comunicação entre agentes, a negociação entre eles é um aspeto chave para um bom funcionamento de um sistema em geral. A negociação é um fator importante num sistema porque, uma vez que estes são autónomos e conseguem tomar decisões sozinhos, podem ser agentes que se preocupam mais com os próprios objetivos do que um bom funcionamento do sistema. Deste modo, existem protocolos de negociação entre agentes para que seja resolvida uma adversidade de modo satisfatório para ambos. Para que uma negociação seja feita com sucesso, ambas as partes devem entender devidamente o protocolo de

modo a chegar a um consenso para resolver uma tarefa da melhor forma. Para se chegar a uma solução com sucesso, normalmente uma tarefa é dividida em subtarefas e este processo pode ser feito no ato de negociação entre agentes (Beer et al., 1999; Fu et al., 2015).

Foi proposto um protocolo de negociação em que o objetivo é que os agentes cheguem a um consenso daquilo que podem oferecer em função dos benefícios poderão ter, que é designado como "*Equilibrium"*. Neste protocolo assegura-se também que os objetivos, as preferências e as políticas dos agentes são satisfeitas (Fu et al. 2015).

2.2.4 Tomada de Decisão num Sistema Multiagente

Os agentes presentes num SMA interagem uns com os outros e, em certas ocasiões, com os humanos que estão presentes no mesmo ambiente. A Tomada de Decisão num Sistema Multiagente (TDSM), tal como o nome indica, preocupa-se com a tomada de decisão num Sistema Multiagente através da comunicação, cooperação e negociação entre os agentes presentes neste sistema.

Existem diversos modelos de TDSM. Existem modelos com objetivos qualitativos e modelos com objetivos quantitativos. Nos modelos com objetivos quantitativos, os agentes focam-se em escolher a melhor opção dentro de muitas opções. Os modelos com objetivos qualitativos são utilizados em casos mais realísticos e complicados, onde existem muitos fatores a ter em conta, ocasionando resultados que não são fáceis de quantificar (Bulling, 2014).

2.2.4.1 Markov Decision Process (MDP)

Um MDP é um processo de controlo estocástico (que o estado é indeterminado baseado em eventos aleatórios) em tempo discreto e é definido por uma sequência finita de objetos denominada "tupla". Este é um processo de tomada de decisão individual, onde o agente escolhe a sua decisão independentemente de outros agentes. A figura 2.6 mostra a estrutura de um MDP, onde estão presentes todos os objetos da tupla (Bulling, 2014; Kallenberg, 2011; Littman, 1994; Rizk et al., 2018):

- "s" representa um dos estados possíveis do ambiente.
- "a" é a ação que pode ser tomada pelo agente depois de observado o estado do ambiente.
- "r" representa a função "*reward"* que determinada qual a recompensa para a ação tomada naquele estado do ambiente.
- Esta presente também uma função probabilística que calcula qual a probabilidade da transação de um estado do ambiente para outro, baseado no estado atual.

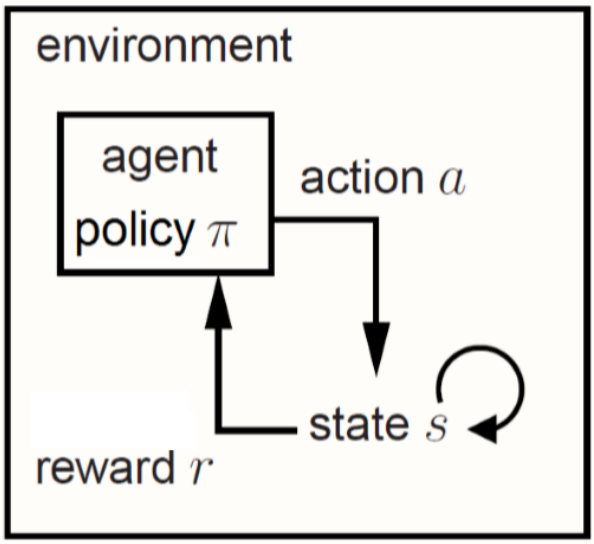


Figura 2.6-Arquitetura MDP (Kallenberg, 2011).

2.2.4.2 Game Theory

Game Theory é considerado uma forma racional de tomada de decisão. Esta já é uma tomada de decisão multiagente e, ao contrário do que acontece num MDP, as decisões dos agentes são interdependentes umas das outras, mesmo os objetivos destes sendo incompatíveis (Bulling, 2014).

2.2.4.3 Stochastic Games

A tomada de decisão realizada pelos jogos estocásticos é uma generalização de todos os processos de tomada de decisão de Markov. São tomadas de decisão realizadas para o interesse próprio de um agente e, apesar de poderem ser modeladas configurações para que sejam feitas de forma cooperativa, são sempre focadas na melhor resposta singular de um agente. Os jogos estocásticos, inserem-se no tipo de modelos com probabilidades quantitativas e recompensas como já foi discutido anteriormente (Rizk et al., 2018).

2.2.4.4 COMmunicative Multiagent Team Decision Problem (COM-MTDP)

Este é um modelo em que os agentes agem em equipa e sempre com objetivos em comum, não tomando decisões egoístas para atingir esses objetivos. Os agentes não têm objetivos privados. É assumido que, neste modelo, todos os membros têm as mesmas preferências (Pynadath & Tambe, 2002).

Baseado no que foi dito no último parágrafo, é assumido que o modelo é comum para todos os membros da equipa e que todos eles acreditam que para os outros é igualmente comum.

Posto isto, também este modelo é representado como uma tupla em que os diferentes componentes são (Pynadath & Tambe, 2002):

Estado do mundo (S): O estado do mundo aqui representa o estado do ambiente da equipa. O "S" representa todas as combinações das perceções individuais de cada membro;

Ações a nível do Domínio (A_α): Representa um conjunto de ações, de cada agente, que serão realizadas de maneira a mudar o estado do ambiente;

Observações do agente (Ω_α): É um conjunto de observações que cada agente tem do seu mundo. Estas observações podem ser indiretamente obtidas a partir de elementos do ambiente, como por exemplo os outros agentes;

Função de Observação (O_α): Esta função representa erros dos sensores;

Extensão para um espaço de estados de crenças melhor (B_α): Esta função cria uma série de crenças baseadas nas observações de um agente. Depois, estas crenças são transformadas em ações; e

Função Recompensa (R): Esta função representa as preferências para a equipa de agentes ao longo dos estados, onde cada agente tem como preferência o que seja melhor para a equipa.

Na Tabela 2.1 são resumidos os diferentes modelos TDSM apresentados, sendo esta baseada na principal característica que os define.

Tabela 2.1-Descrição dos modelos TDSM

Modelo TDSM	Característica
Markov Decision Process	Decisão individual independentemente da decisão dos outros agentes presentes no sistema
Game Theory	Decisão interdependente dos outros agentes presentes no sistema (decisão multiagente)
Stochastic Games	Decisão individual com foco na melhor resposta singular de um agente
COM-MTDP	Decisão em equipa para atingir os objetivos do sistema

2.3 Projetos relacionados com este trabalho

Nesta secção são apresentados projetos em que o conceito está diretamente relacionado com o que é pretendido atingir neste trabalho.

-Futebol robótico (Reis, 2003)

Esta tese de doutoramento foca a criação de uma simulação de futebol robótico, tendo sido feita com o intuito de participar numa competição internacional associada à investigação do futebol robótico. Esta competição atraiu a atenção de algumas das melhores empresas e universidades mundiais, como a Sony, a Philips, a universidade de Cornell.

Nesta tese, o autor começa por estudar todos os conceitos associados a agentes e sistemas multiagente, depois passa a definir linguagens que permitam a comunicação e coordenação entre agentes autónomos, implementa também estratégias de coordenação para as equipas de agentes e finalmente implementa a equipa de futebol robótico, formada pelo conjunto de agentes, com condições pré-especificadas para participar na competição anteriormente falada (Reis, 2003).

-Multi-agent manufacturing scheduling system for dynamic environment (Jiang et al., 2016)

Neste projeto, o objetivo era a simulação de um sistema de manufatura real e para tal foi usado um modelo genérico de uma fábrica. Neste modelo, foi equipado um sistema multiagente, em que os agentes estão todos interconectados e têm a capacidade de tomada de decisão independente.

Neste sistema multiagente foram implementados 4 tipos de agente: "*Job Agent", "Machine Agent", "Machine Group Agent"* e *"Shop Agent".* Um *"Job Agent"* trata da informação inerente a um certo trabalho. O *"Machine Agent"* é responsável pelas tomadas de decisão em relação a uma certa máquina, uma vez que cada máquina tem as suas próprias especificidades e lida com as diferentes operações da fábrica de maneira diferente. O "*Machine Group Agent"* tem como função manter dados estatísticos de um grupo de máquinas com as mesmas capacidades. Finalmente, o "*Shop Agent"* trata da gestão da fábrica, contendo alguma informação geral e da topologia dos agentes.

Neste sistema é utilizado um protocolo de negociação e um algoritmo para a escolha da máquina que mais se adequa à tarefa que existe para completar (Jiang et al., 2016).

Todos os conceitos discutidos neste capítulo de Estado de Arte serão importantes para a realização do trabalho. O objetivo será a criação de um SMA, para isso terá que ser decidido qual

a arquitetura dos agentes, qual o ambiente em que estarão inseridos e a forma como irão comunicar. Para além dos SMA, referiram-se em vários conceitos relativamente a Redes Colaborativas, que se pretendem aplicar ao SMA, como por exemplo, o conceito consórcio de colaboração de forma a criar um SMA eficiente e eficaz.

2.4 Tipos de sistemas de produção de Manufatura

Um sistema de produção tem como objetivo a criação de um produto ou a disposição de um serviço, sendo que pode-se estar a referir, por exemplo, a uma fábrica (Pessoa & Spinola, 2014).

Existem dois principais tipos de produção:

- **Produção Contínua -** Normalmente é um processo de produção em massa de produtos com pouca variedade. É também denominada produção em fluxo, sendo que é feita continuamente sem interrupções. A produção de matérias-primas, como por exemplo os derivados do petróleo, é um exemplo típico (Pessoa & Spinola, 2014).

- **Produção Discreta –** Também referida como produção descontínua, é um tipo de produção que trata de quantidades relativamente pequenas em que os produtos finais, normalmente, são diferentes e personalizados. Em fábricas, as máquinas são agrupadas por funções e a produção de um produto exige um planeamento trabalhoso. Exemplos deste tipo de produção estão na indústria automóvel e na indústria da alta tecnologia (Pessoa & Spinola, 2014; Calado, 2015).

Tabela 2.2-Comparação entre produção contínua e discreta (Calado, 2015).

Tipo de produção	**Contínua**	**Discreta**
Tipos de Ordem	Contínua	Lotes
Fluxo de Produção	Sequencial	Mal definido
Variedade	Pouca	Muita
Tipo de Mercado	Massa	Por pedido
Volume	Elevado	Médio

Na Tabela 2.2 pode-se observar uma comparação entre os dois principais tipos de produção descritos. É de notar que o tipo de mercado da produção descontínua (discreta) é “por pedido”, o que representa uma produção personalizada.

Posto isto, é importante mencionar que a implementação do sistema colaborativo, base desta dissertação, vai ser assente no contexto de produção discreta visto que é uma produção planeada dependendo do pedido, o que torna possível pedidos customizados. Também é uma produção em que, depois da criação de um produto, é possível descrever todos os componentes que o constituem.

Capítulo 3

3. Contexto de Desenvolvimento

Neste capítulo são descritos os objetivos pretendidos no desenvolvimento deste sistema de colaboração multiagente. São também discutidas funcionalidades pretendidas para o mesmo.

3.1 Enquadramento

O objetivo deste trabalho é a realização de um sistema multiagente colaborativo que de certa forma contribua para uma otimização de uma linha de manufatura. Para atingir esta otimização, pensou-se num cenário para o aumento do nível de descentralização de uma fábrica, tornando-a mais inteligente, automática e com a possibilidade de corresponder a pedidos customizados.

Um dos objetivos no desenvolvimento deste sistema é fazê-lo baseado em conceitos de redes colaborativas, tratando este sistema multiagente como um tipo de rede colaborativa. O cenário idealizado seria um com as seguintes características:

- Montagem de manufatura discreta, ou seja, onde existiria uma produção unitária separada em que cada unidade requeira diferentes tipos de componentes;
- Montagem onde existiriam diferentes tipos de equipamentos/máquinas com diferentes ou iguais tipos de recursos e/ou capacidades;
- Seria conectado a cada equipamento/máquina um agente responsável com o objetivo de integrar um sentido de perceção do estado do mundo;
- Os agentes teriam maneira de comunicar e, consequentemente, colaborar entre si de forma a ter um sistema eficiente;
- Todo o sistema multiagente seria considerado uma VBE, ou seja um ambiente onde as entidades têm potenciais de colaboração entre si quando surge uma oportunidade de negócio, que neste caso seria a solicitação de um dado produto pela parte do cliente;

- Para corresponder à solicitação de um produto, através de determinados processos e de maneira eficiente, seria formada uma equipa colaborativa composta pelos agentes de cada máquina/equipamento desta montagem. Neste caso, como o objetivo é a criação de uma equipa num curto espaço de tempo com, também, um curto ciclo de vida, esta seria considerada uma "*Dynamic Virtual Organization*" (VO), passando à fase de dissolução uma vez que a criação do produto estivesse completa.

3.2 Funcionalidades pretendidas

O principal objetivo para o desenvolvimento deste sistema é que as suas funcionalidades se enquadrem nas fases do ciclo de vida de uma VO, com o principal foco nas fases de criação e operação.

Na tabela 3.1 podem-se observar as funcionalidades desejadas para cada fase do ciclo de vida de cada VO no desenvolvimento deste sistema.

Tabela 3.1-Funcionalidades pretendidas para cada fase do ciclo de vida.

Fase do Ciclo de Vida	**Funcionalidades**
Identificação da Oportunidade de Colaboração	• Agente de cada máquina/equipamento faz uma análise à solicitação de produto para verificar se há algo com que possa colaborar. • Caso essa solicitação seja do interesse do agente, este faz uma proposta com o que a máquina/equipamento pode oferecer. • Escolha, de acordo com as preferências do cliente, da melhor equipa para colaboração na criação do produto solicitado. • Criação da equipa e distribuição de tarefas, entre os membros da mesma, para a produção do produto. • Agente cuja máquina/equipamento possui recurso sem stock, efetua um pedido de stock a um agente cuja máquina/equipamento possua o mesmo tipo de recurso. Apenas será solicitado *stock,* por parte do agente contactado, caso este esteja ocupado na colaboração para um pedido.
Operação	• Existência de um agente responsável pela coordenação da execução das tarefas necessárias para a criação de um produto. • Execução das tarefas pelos membros da equipa colaborativa para a criação de um produto.

Dissolução	• Atualização dos estados dos agentes. • Atualização do estado da equipa colaborativa, transitando para o estado inativo (estado que possibilita que, posteriormente, a mesma equipa realize, novamente, a criação do mesmo produto caso todos os membros se encontrem disponíveis). • Possibilidade do cliente dar uma opinião do serviço que lhe foi oferecido pela equipa de colaboração.

3.3 Desenho e Interface

Para efeitos de validação do *software* desenvolvido, pretende-se também implementar interfaces gráficas que permitam realizar uma simulação e monitorização do sistema, tornando as interações e funcionalidades do sistema observáveis.

Na simulação do sistema, será importante considerar as seguintes etapas ou fases:

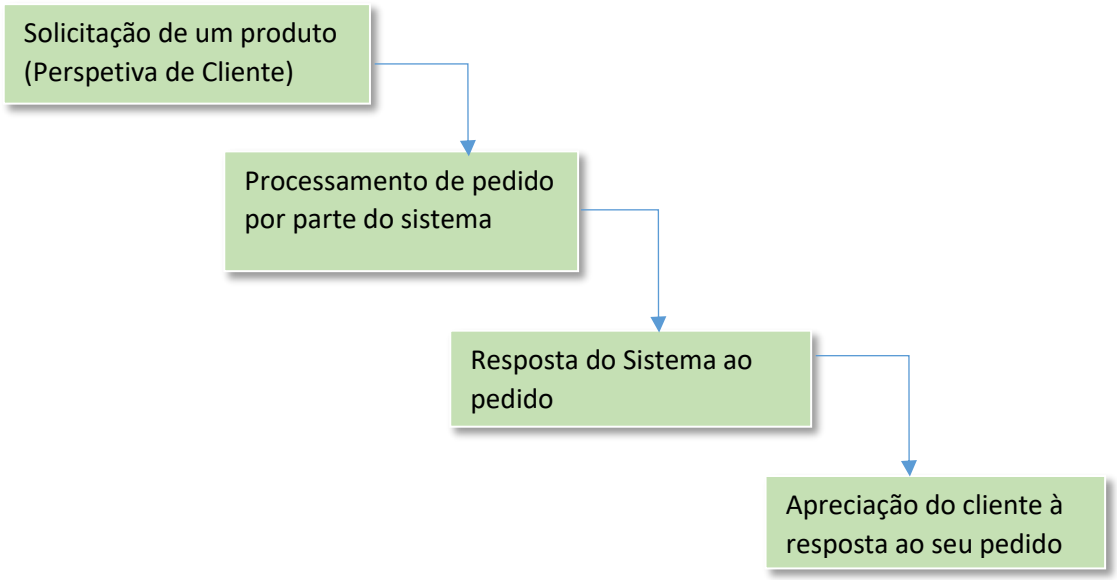


Figura 3.1-Fases do sistema a considerar para o desenvolvimento de Interfaces Gráficas.

De acordo com a Figura 3.1, para representar a primeira etapa deve existir uma interface gráfica que possibilite a solicitação de um produto, conforme as preferências do cliente.

Deve existir, também, uma interface de monitorização do *software* que torne observável o processamento do pedido e a resposta do sistema ao mesmo. Nesta interface, deverão ser

visíveis os agentes responsáveis por cada máquina, tendo uma descrição de cada uma delas. Deverá ser demonstrada uma descrição para cada uma das solicitações e, também, para cada uma das colaborações, mostrando o seu estado e os membros que a incorporam.

Para a última etapa da Figura 3.1, pretende-se realizar uma interface onde, uma vez que o produto esteja concluído, possa ser simulada uma apreciação do cliente ao pedido realizado. Este feedback tem como objetivo criar mais funcionalidades no sistema, sendo que está a ser concedido à equipa responsável pelo pedido.

Capítulo 4

4. Arquitetura

Neste capítulo é apresentada a arquitetura proposta para o sistema desenvolvido. Esta arquitetura é baseada num contexto multiagente que visa maximizar as capacidades de um ambiente de manufatura. Posto isto, no presente capítulo é apresentada a arquitetura conceitual do sistema, os seus requisitos e todos os constituintes que nele estão presentes.

4.1 Arquitetura Conceitual

O modelo conceitual do sistema apresentado, como pode ser observado na Figura 4.1, tem como base o sistema multiagente a ser implementado. Como se pode constatar, esta arquitetura permite a interação entre a ação humana e o sistema, que corresponde à solicitação de um novo produto da parte do cliente.

Neste sistema estão presentes três classes de agentes*: Console Agent*, *Customer Agent* e *Supplier Agent*, todos eles com funções distintas.

É importante referir que o sistema suporta a presença de múltiplos agentes, sendo que na arquitetura apresentada apenas existe uma abstração de cada um deles. Esta é uma particularidade do sistema que permite a existência do paralelismo de ações, o que vai ao encontro de um dos principais objetivos deste trabalho: ter um sistema eficiente.

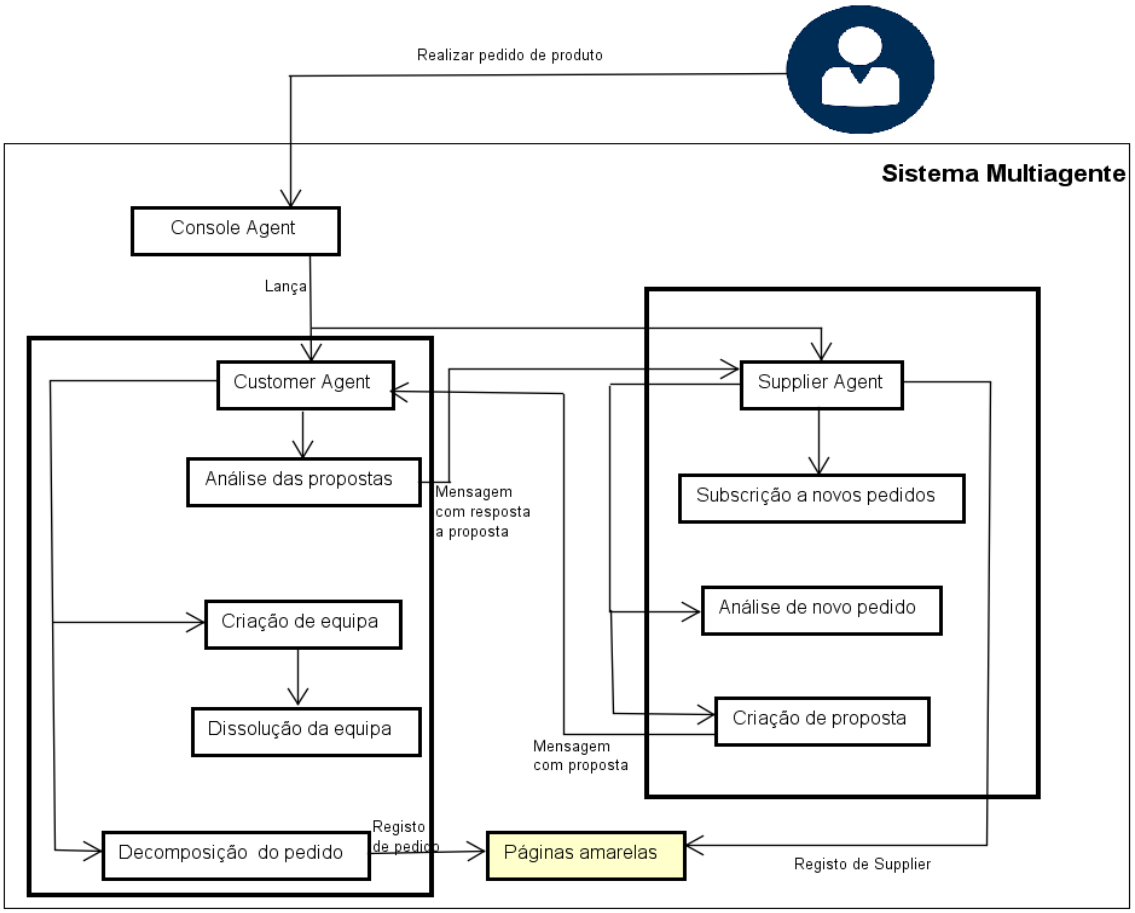


Figura 4.1-Arquitetura Conceitual do Sistema.

Depois do pedido de um novo produto, são lançados pelo *Console Agent* todos os agentes responsáveis por abstrair as entidades presentes no sistema. Cada um destes agentes tem um conjunto de comportamentos criados com o apoio da plataforma JADE (*Java Agent DEvelopment Framework*), que vai ser abordada no capítulo seguinte.

É de referir que o *Console Agent* permite a realização de novos pedidos depois do primeiro, tendo por consequência o lançamento de um novo *Customer Agent*, sem que seja necessário lançar novamente os *Supplier Agents*.

Deste modo, na Tabela 4.1 são apresentadas breves descrições, de um ponto de vista funcional, para cada uma das principais componentes apresentadas aqui nesta arquitetura conceitual.

Tabela 4.1-Descrição dos Agentes e as suas principais funcionalidades.

Grupo	Classes	Descrição
Agentes	Console Agent	Agente responsável por iniciar o sistema, lançando os agentes necessários para a abstração de todas as máquinas/equipamentos presentes no ambiente e do cliente.
	Customer Agent	Agente responsável por abstrair um cliente, cujas principais características são o pedido com as respetivas preferências e os dados do cliente.
	Supplier Agent	Agente responsável por abstrair uma máquina ou equipamento, cujas principais características são os seus Recursos e *Skills*.
Comportamentos (Customer Agent)	Análise de Propostas	Comportamento responsável por avaliar todas as propostas recebidas pelos agentes do tipo Supplier Agent. Após a avaliação das propostas, é aceite somente uma para cada tarefa, sendo cada uma delas respondida com uma mensagem de aceitação. Todas as restantes são respondidas com uma mensagem de rejeição.
	Decomposição do pedido	Comportamento responsável pela decomposição do pedido em tarefas. Sendo estas registadas como propriedades do pedido num serviço de Páginas amarelas suportado pelo JADE.
	Criação da equipa	Comportamento responsável por criar a equipa de colaboração responsável pelo pedido do cliente abstraído por este agente. Aqui é também realizado um planeamento para realizar as tarefas pela ordem correta.
	Dissolução da equipa	Comportamento responsável pela dissolução da equipa, uma vez que o processamento do produto foi completo. Neste, são atualizados estados de todos os Agentes presentes no sistema.
Comportamentos (Supplier Agent)	Subscrição a novos pedidos	Comportamento responsável pela "vigilância" da inserção de um novo pedido no sistema. Este foi criado com o objetivo de dar um sentido de captura de oportunidade ao agente.
	Análise de novo pedido	Comportamento responsável pela análise das tarefas do novo pedido presente no sistema. Este, passa pela verificação do que o agente pode (ou não) ter para a criação de proposta por alguma das tarefas.
	Criação de proposta	Comportamento responsável pela criação de uma proposta caso exista a competência (Skill ou Recurso) necessária para a realização da tarefa em questão.

Serviços	Páginas Amarelas	Serviço disponibilizado pela JADE, o qual é utilizado para o registo de agentes e pedidos no sistema.

Estes comportamentos terão uma abordagem mais desenvolvida no capítulo seguinte, onde será apresentada a implementação de cada um deles.

4.2 Requisitos funcionais e não funcionais

Baseado na descrição das componentes da arquitetura apresentada na secção anterior, neste vão ser apresentados os requisitos funcionais e não funcionais do sistema.

Neste sistema existem duas entidades principais: "Cliente" e "Fornecedor" que são abstraídas, respetivamente, pelos agentes do tipo *Customer Agent e Supplier Agent*.

Na Tabela 4.2 são descritos os principais requisitos funcionais relativos à entidade "Cliente".

Tabela 4.2-Requisitos Funcionais relativos ao "Cliente".

Requisito	Descrição
Fazer pedido	Funcionalidade que realiza a criação de um novo pedido de produto. Este pedido é associado ao cliente que o fez, que por sua vez é abstraído por um agente do tipo Customer Agent.
Adicionar preferências ao pedido	Funcionalidade que permite, no ato da solicitação de um novo produto, que o cliente possa associar ao pedido uma preferência (Ex: Tempo ou Custo).
Decomposição do pedido	Funcionalidade que permite aos agentes do tipo Customer Agent fazer uma decomposição do pedido em tarefas. Desta forma, o processo de análise do pedido realizado pelos agentes do tipo Supplier Agent é simplificado, uma vez que analisam tarefa a tarefa.
Analisar propostas	Funcionalidade que permite aos agentes do tipo Customer Agent fazer uma análise das propostas recebidas para cada tarefa necessária de realizar para corresponder ao pedido do cliente cujo está responsável.
Criar equipa Colaborativa	Funcionalidade que permite os agentes do tipo Customer Agent criar a equipa de colaboração, quando todas as tarefas têm um Supplier Agent responsável para as realizar.

Revogar pedido	Funcionalidade usada caso não seja possível a fabricação de um produto solicitado pelo cliente devido à falta de recursos.
Planeamento da colaboração	Funcionalidade que permite um Customer Agent planear a sequência de ações a serem realizadas pelos membros da equipa (Supplier Agents) responsável pelo pedido do cliente que abstrai. Sequencialmente, este agente envia uma ordem de execução da tarefa para o membro responsável por ela.

Os principais requisitos funcionais relativos à entidade "Fornecedor" são descritos na Tabela 4.3.

Tabela 4.3-Requisitos Funcionais relativos ao "Fornecedor".

Requisito	**Descrição**
Subscrever serviços do tipo pedido	Funcionalidade que permite a um Supplier Agent subscrever aos serviços do tipo "pedido". Isto é, quando um novo pedido é inserido no sistema, os agentes deste tipo são informados de modo a poderem prosseguir à análise das tarefas presentes neste pedido.
Analisar pedidos	Funcionalidade que permite a um Supplier Agent analisar as tarefas necessárias para a fabricação de um produto, com a finalidade de encontrar aquelas que a máquina/equipamento que abstrai está mais habilitada para cumprir.
Fazer proposta	Funcionalidade que permite a um Supplier Agent enviar uma proposta, ao Customer Agent responsável pelo pedido analisado, para uma ou mais tarefas que pretenda realizar.
Executar tarefa	Funcionalidade que permite a um Supplier Agent realizar a execução de uma tarefa.
Pedir stock	Funcionalidade que permite a um Supplier Agent requisitar a um agente da mesma classe, *stock* para fazer proposta a determinada tarefa.
Disponibilizar stock	Funcionalidade que permite a um Supplier Agent disponibilizar stock a agente da mesma classe.
Adicionar novo Fornecedor	Funcionalidade que permite incorporar um novo fornecedor ao sistema, criando um agente do tipo Supplier Agent para o abstrair.

Possuir Ranking	Funcionalidade que permite cada Supplier Agent ter uma posição no "ranking de Supplier Agents", tendo em consideração alguns parâmetros do mesmo, como por exemplo o número de colaborações que já fez parte.

De modo a explicitar os requisitos "Pedir stock" e "Disponibilizar stock", é importante referir que os agentes da classe "Supplier Agent" podem assumir um papel de "Buddy" uns para os outros. Esta é uma classe que permite o contacto entre dois agentes da classe em questão, a fim de aumentar a agilidade do sistema. É de salientar que estas funcionalidades são exercidas, ainda, na fase de identificação da oportunidade de colaboração. Isto quer dizer que se trata do contacto entre dois membros da considerada VBE, no qual estes acabam por colaborar entre si apenas de modo a aumentar a eficiência do sistema, não tendo qualquer relação com a fase de operação.

Para além das funcionalidades relacionadas com as entidades "cliente" e "fornecedor", também são apresentados na Tabela 4.4 alguns dos principais requisitos não funcionais que estão presentes neste sistema.

Tabela 4.4-Requisitos não funcionais do sistema.

Requisito	**Descrição**
Monitorização de Fornecedores	Funcionalidade que permite monitorizar os dados relativos aos fornecedores presentes no sistema.
Monitorização de Pedidos	Funcionalidade que permite monitorizar os dados relativos aos pedidos presentes no sistema.
Monitorização das colaborações	Funcionalidade que permite monitorizar as colaborações ativas, inativas e, também finalizadas, apresentado um histórico de colaborações do sistema.

Estes requisitos descritos na Tabela 4.4 estão relacionados com a usabilidade do sistema. Estes são suportados por um conjunto de interfaces gráficas que têm como objetivo possibilitar a monitorização das principais entidades do sistema, mostrando fundamentalmente os seus dados em tempo real.

4.3 Modelo de Dados

De maneira a conceitualizar todas as entidades presentes neste sistema e a criar um entendimento comum entre os agentes, foi utilizado o *Protégé* para a definição de uma ontologia. Esta é uma Framework que vai ter uma abordagem mais aprofundada posteriormente.

O *Protégé*, posteriormente à criação da ontologia, permite gerar ficheiros java que a representam. A ideia será consultar e gerir esta ontologia através do sistema multiagente já discutido anteriormente.

Sendo que esta Framework permite criar Classes, criar relações entre elas e adicionar propriedades a cada uma delas, na Figura 4.2 é apresentado um diagrama de classes que foi utilizado como base para que o desenvolvimento da ontologia, no *Protégé*, cumprisse as necessidades do sistema.

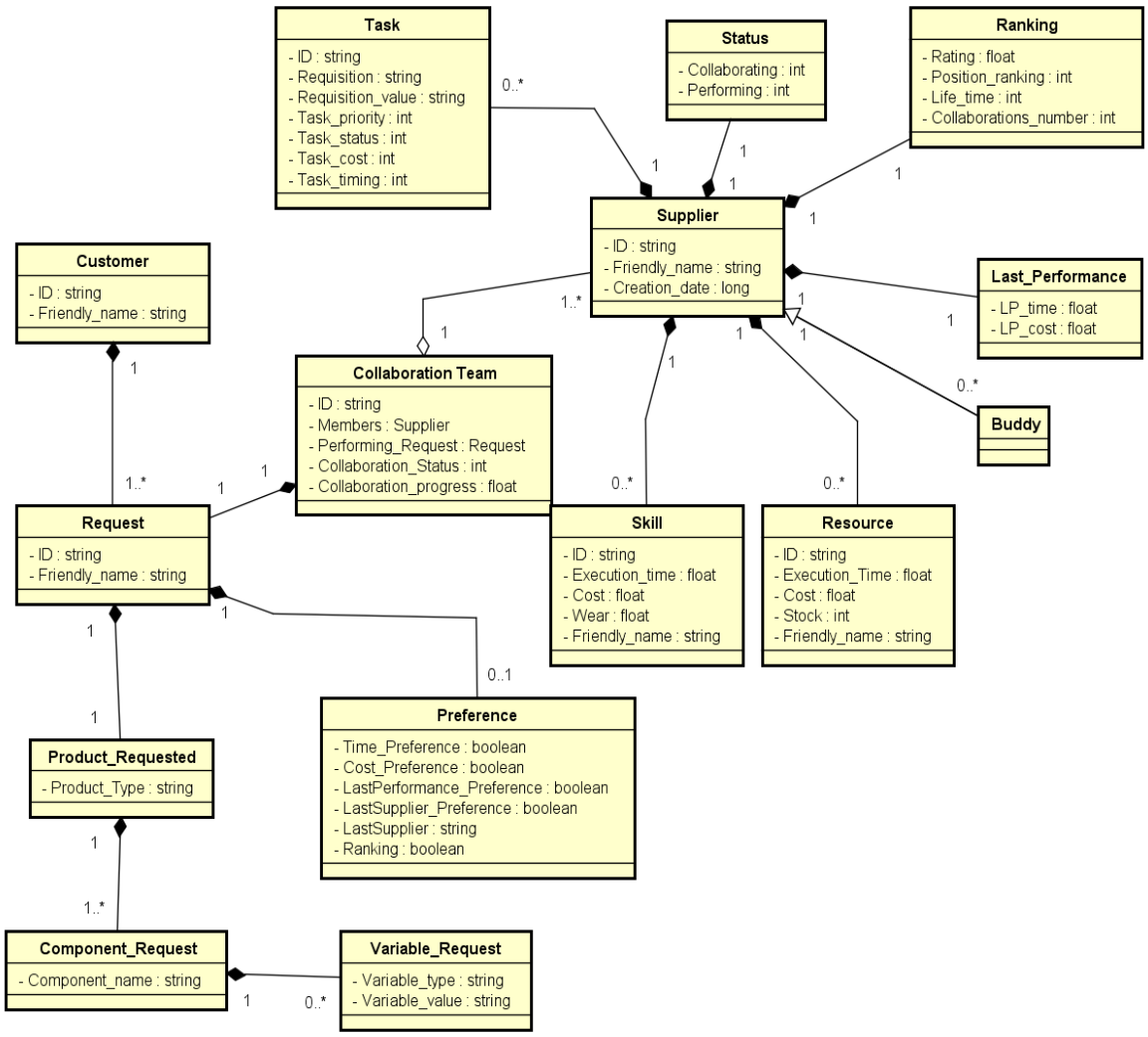


Figura 4.2-Diagrama de classes do sistema.

É importante referir que uma ontologia não é vista como uma base de dados mas sim como uma "base de conhecimento" em que é permitido a criação de instâncias, as quais são representações das classes com os valores propriedades definidas. Por exemplo, podemos criar uma instância da classe "Status" (presente na figura acima) com o nome "status1", que tenha os valores de 1 e 0 nas propriedades "*Collaborating*" e "*Performing*", respetivamente. Presume-se que esta "base de conhecimento" seja capaz de suportar todas as funcionalidades do sistema.

Nesta ontologia estão presentes as seguintes classes/entidades e relações:

- **Classe Customer**
  Classe referente à entidade "cliente" e que tem como propriedades: ID e Friendly_name. Tem uma relação 1-1* com a classe Request, pois um cliente poder ter um ou mais pedidos e, também, uma relação 1-0…1 com a classe Preference dado que um pedido pode ter uma ou nenhuma preferência, escolhida pelo cliente.
- **Classe Request**
  Classe referente à entidade "pedido" e que tem, também, como propriedades: ID e Friendly_name. Tem uma relação 1-1 com a classe Product_Requested, pois cada pedido pode ter unicamente um produto associado.

- **Classe Product_Requested**
  Classe referente à entidade "produto" que tem a Product_Type como propriedade. Tem uma relação 1-1* com a classe Component_Request, pois cada produto pode ser composto por vários componentes.

- **Classe Component_Request**
  Classe referente à entidade "componente" que tem a Component_name como propriedade. Tem uma relação 1-0* com a classe Variable_Request, pois cada componente pode ter várias variáveis para a detalhar.

- **Classe Variable_Request**
  Classe referente a possível variável que uma componente possa ter para a detalhar. Tem como propriedades: Variable_type e Variable_value.

- **Classe Preference**
  Classe referente às preferências que um pedido pode possuir.

- **Classe Supplier**
  Classe referente à entidade "fornecedor" que tem como propriedades: ID, Friendly_name e Creation_date. Esta Classe tem as seguintes relações:

- Supplier-Status
  Cada classe Supplier tem na sua composição uma classe Status de maneira a efetuar algumas verificações para certas funcionalidades do sistema.

- Supplier-Collaboration_Team
  Cada classe Collaboration_Team tem na sua composição um determinado número de classes Supplier responsáveis por um pedido.

- Supplier- Skill
  Cada classe Supplier pode ter na sua composição zero ou mais classes Skills.

- Supplier-Resource
  Cada classe Supplier pode ter na sua composição zero ou mais classes Resources.

- Supplier-Last_Performance
  Cada classe Supplier tem na sua composição uma, e uma só, classe Last_Performance de maneira a representar o seu desempenho na última tarefa realizada.

- Supplier-Ranking
  Cada classe Supplier tem na sua composição uma, e uma só, classe Ranking, de maneira a representar o seu Ranking entre todas as classes Supplier existentes no sistema.

- Supplier-Task
  Cada classe Supplier pode ter na sua composição uma ou mais classes Task, de maneira a representar as tarefas pelas quais este é responsável de executar.

- Supplier-Buddy
  Cada classe Supplier pode estar relacionado com zero ou mais classes Buddy. Esta é uma relação de herança dado que a classe Buddy representa determinada entidade Supplier.

Capítulo 5

5. Estrutura de Implementação

No presente capítulo será abordada a implementação de cada um dos módulos deste sistema, apresentando também as tecnologias utilizadas para a efetuar.

5.1 Tecnologias de implementação

A implementação do sistema de colaboração multiagente, base desta dissertação, passa pelos três módulos seguintes: Ontologia, Sistema Multiagente e Interface gráfica (GUI). Formando assim, o protótipo proposto para esta dissertação.

Desta forma, na Figura 5.1 surge uma representação da estrutura deste protótipo.

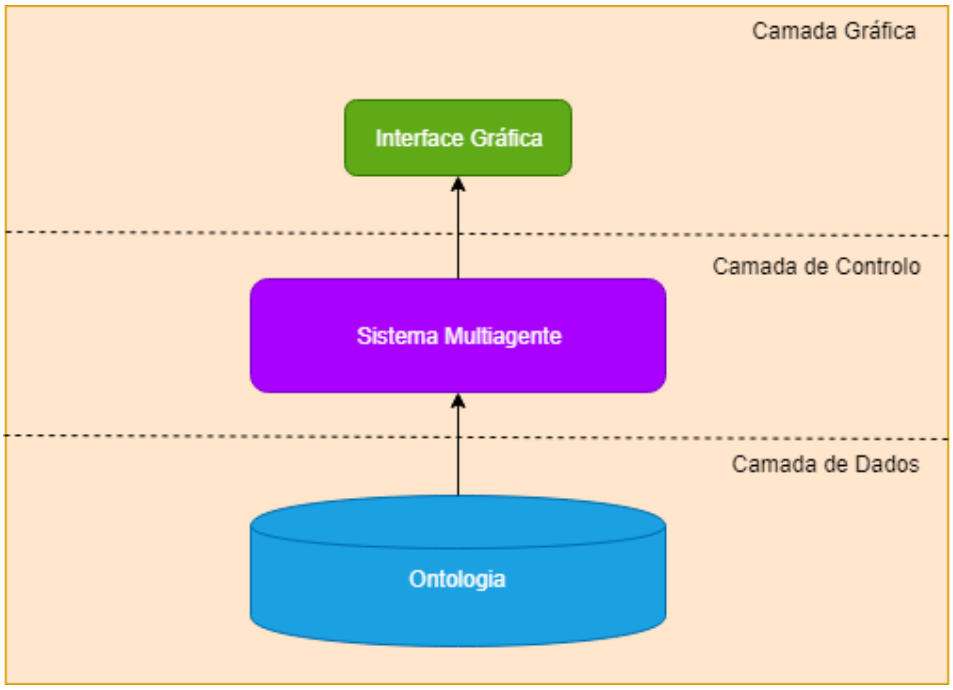


Figura 5.1-Diagrama de camadas da estrutura do sistema.

5.1.1 – Camada de Dados

Com o intuito de ser partilhado um conhecimento em comum entre os agentes deste sistema, foi criada uma ontologia. O *software* utilizado foi o Protégé, que é um editor de ontologias escrito em Java.

Nesta, são criadas as classes que representam as entidades deste sistema e todas a relações que existem entre elas. A ontologia permite a criação de instâncias (*individuals*) para cada classe, o que possibilita o armazenamento de todos os dados e a informação pertencente a estas entidades.

Posteriormente à criação da ontologia, foi gerado o seu código Java através da ferramenta "Protege-OWL Code Gerator" concedida pelo *Protégé*.

5.1.2 – Camada de Controlo

Nesta camada está presente o sistema multiagente, que é responsável por manipular a ontologia criada.

A plataforma NetBeans foi a utilizada como IDE (*Integrated Development Environment*) para a implementação deste sistema, sendo o desenvolvimento deste sistema multiagente feito com base em JADE *(Java Agent DEvelopment Framework*). Esta é uma *framework* completamente implementada na linguagem Java que facilita o desenvolvimento de aplicações baseadas em agentes peer-to-peer, facultando critérios FIPA.

5.1.2.1 FIPA (Foundation for Intelligent Physical Agents)

A FIPA foi criada com o objetivo de formar padrões de *software* para serem usados por agentes heterogéneos de modo a que possam interagir entre eles. Desta forma, num sistema muti-agente qualquer tipo de agente pode comunicar com outro sem ter que se incomodar se ele o irá perceber ou não (Poslad, 2007).

A FIPA padronizou um conjunto de atos comunicativos, FIPA Communicative Acts(CAs), com o objetivo de expressar o estado mental e intenção dos atos de um remetente. Na tabela 5.1 são mostradas as classificações dos diferentes atos comunicativos (Poslad, 2007).

Tabela 5.1-Tipos de Atos Comunicativos (Poslad, 2007).

Communicative Act (CA)	Base CA	Assertive	Commisive	Directive	Mediate	Phatic	Query
accept-proposal	inform			X			
agree	inform		X				
cancel	disconfirm					X	
cfp	query-ref			X			
confirm	confirm	X					
disconfirm	disconfirm	X					
failure	inform					X	
inform	inform	X					
inform-if	inform	X					
inform-ref	inform	X					
not-understood	inform					X	
propagate	inform				X		
propose	inform			X			
Proxy	inform				X		
query-if	request						X
query-ref	request						X
refuse	disconfirm; inform					X	
reject-proposal	inform			X			
request	request			X			
request-when	inform			X			
request-whenever	inform			X			
subscribe	Request -whenever						X

Na implementação deste trabalho, é utilizada a linguagem de comunicação para sistemas multiagente, FIPA ACL (*Agent Communicative Language*), em que cada mensagem é um ato comunicativo que tem vários atributos de suporte a uma boa argumentação e, devendo, ser todos bem especificado (Xiong et al., 2012).

Tabela 5.2-Estrutura de uma mensagem FIPA ACL (Xiong et al., 2012).

Element	Discription
performative	What action the message performs
sender	Initiator of the message
receiver	Recipient of the message
reply-to	Recipient of the message reply
content	Content of the message
encoding	Encoding used for content
ontology	Ontology context for content
protocol	Protocol message belongs to
conversation-id	Conversation message belongs to
reply-with	Reply with this expression
in-reply-to	Action to which this is a reply
reply-by	Time to receive reply by

5.1.3 – Camada Gráfica

Todas as interfaces gráficas foram criadas a partir da classe JFrame do pacote Swing. O Swing é um "kit de ferramentas" para o uso com o Java, que neste caso foi utilizado a partir do NetBeans. Esta é uma classe que contêm todos os componentes como botões, menus, caixas de texto, etc., permitindo a criação de janelas de acordo com as funcionalidades pretendidas para o protótipo desenvolvido. Não sendo o principal foco deste trabalho, a GUI foi criada de forma simples apenas para propósitos de simulação e monitorização do sistema desenvolvido.

Na tabela 5.3, é apresentado um resumo das contribuições das tecnologias indicadas anteriormente para a obtenção das funcionalidades pretendidas no protótipo desenvolvido, com as respetivas justificações.

Tabela 5.3-Tecnologias utilizadas na implementação do protótipo proposto.

Tecnologia	Contribuição	Justificação
Protégé	Responsável pelo fornecimento de um conhecimento comum da estrutura do sistema, assim como o armazenamento de informação relativa às entidades do mesmo.	- *Software* livre (*Open source*). -Bom desempenho em relação a outras opções. -Extensível – permite gerar o código Java representante da ontologia criada.
JADE	Responsável por simplificar a programação do sistema multiagente.	- *Software* livre (*Open source*). - Possui especificações FIPA (discutidas anteriormente). - Completamente implementado em Java. - Extensa biblioteca de suporte aos comportamentos de cada agente. - Permite a comunicação entre agentes baseada em mensagens assíncronas.
Java	Responsável pela programação das camadas de controlo e gráfica do sistema.	- Orientação a objetos. - Linguagem de programação bastante utilizada com inúmeros recursos disponíveis a nível de trace de erros. - Linguagem utilizada pelo JADE.

5.2 Implementação

Considerando a secção anterior, tendo como base as tecnologias mencionadas, esta secção descreve a implementação de cada uma das funcionalidades pretendidas. Sendo que, serão abordadas com maior destaque as principais funcionalidades do sistema.

A implementação vai ser descrita pela seguinte ordem: ontologia, sistema multiagente, interfaces gráficas.

5.2.1 – Ontologia

Como já foi referido anteriormente, a criação de uma ontologia revelou-se essencial para o suporte do desenvolvimento deste trabalho pelas seguintes razões:

- Partilhar um conhecimento comum entre os agentes.
- Separar o domínio lógico do domínio operacional.
- Conceitualizar o ambiente de aplicação do sistema multiagente.

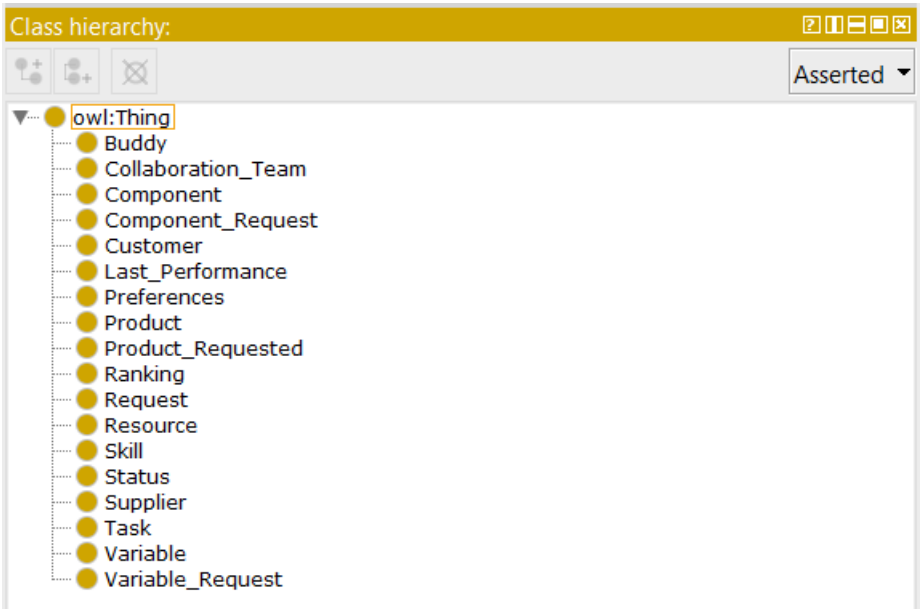


Figura 5.2-Classes da ontologia.

Na Figura 5.2 podem-se observar as classes criadas na ontologia. Para além das classes descritas previamente, podemos observar a Product, Component e Variable. Estas, foram classes criadas de maneira a representar um produto, componente ou variável existente na fábrica, diferentemente daquilo que foi descrito para as classes: Product_Requested, Component_Request, Variable_Request, as quais têm a intenção de representar o produto requisitado.

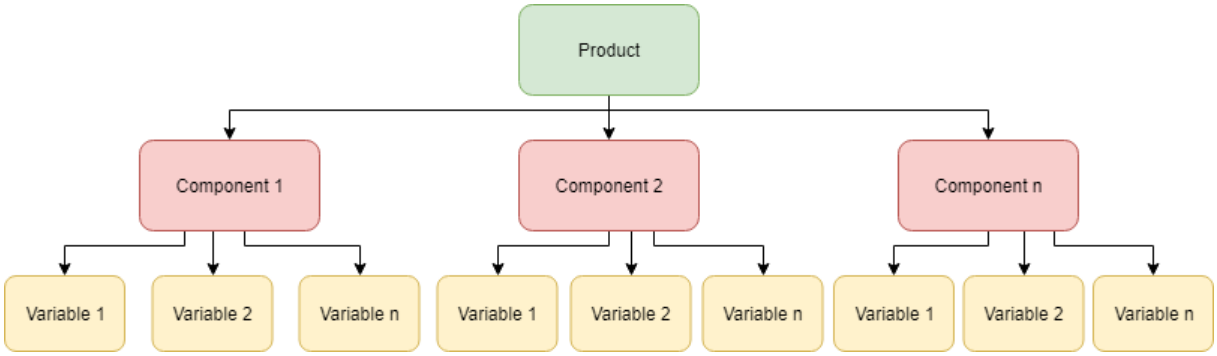


Figura 5.3-Relações entre as classes Product, Component e Variable.

De maneira a esclarecer como estas classes se relacionam, na Figura 5.3 pode-se observar que um produto pode ser composto por n componentes, em que cada um deles pode ser composto por n variáveis. As variáveis possibilitam a descrição de cada componente de um produto. Por exemplo, imagine-se o produto mesa, cujos componentes são um tampo preto e quatro pernas pretas. Neste caso, o tampo seria um componente e cada uma das pernas seria outro componente diferente. Uma das variáveis para cada um destes componentes seria a cor preta.

Esta ideia foi criada com as principais finalidades:

- Criar um sistema o mais genérico possível, de maneira a que possa ser aplicado a um cenário independentemente do tipo de produto.
- Permitir que um cliente realize um pedido customizado (dentro das possibilidades do cenário).

O *Protégé* permite criar relações entre as diferentes classes, através da definição de "*Object Properties*". A Figura 5.4 mostra a "*Object property hierarchy*" criada para esta ontologia.

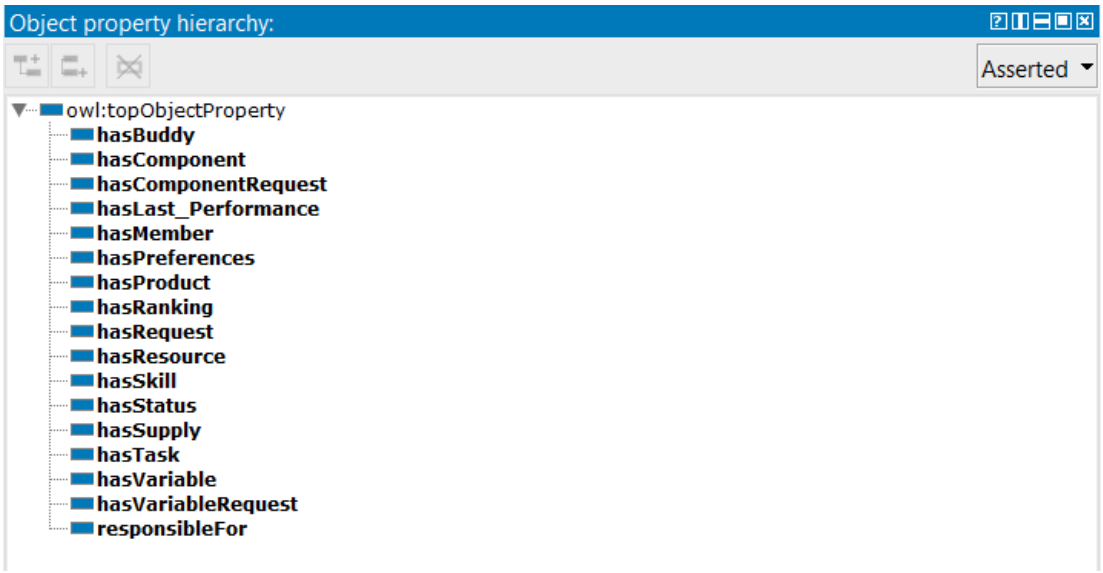


Figura 5.4-Object property hierarchy

Cada "*Object Property"* tem dois parâmetros denominadas "*Domains*" e "*Ranges*", onde é permitido a inserção, em cada um deles, das classes que pretendemos relacionar. Deste modo, na Tabela 5.4 é sumariado os "*Object Properties"* inerentes a esta ontologia.

Tabela 5.4-Object Properties.

Object Property Name	Domains	Ranges
hasBuddy	Supplier	Buddy
hasComponent	Product	Component
hasComponentRequest	Product_Requested	Component_Request
hasLast_Performance	Supplier	Last_Performance
hasMember	Collaboration_Team	Supplier
hasPreferences	Request	Preferences
hasProduct	Request	Product_Requested
hasRanking	Supplier	Ranking
hasRequest	Customer	Request
hasResource	Supplier	Resource
hasSkill	Supplier	Skill
hasStatus	Supplier	Status
hasTask	Request	Task
hasVariable	Component	Variable
hasVariableRequest	Component_Request	Variable_Request
responsibleFor	Supplier	Task

Como pode ser constatado, as "*Object Properties*" foram criadas para conceber uma relação de pertença entre classes, como sugere o nome de cada uma delas.

De forma a descrever as propriedades de cada classe, foi também definida uma "*Data property hierarchy*". Esta permite a atribuição de valores às propriedades de cada instância (individual) criada no sistema.

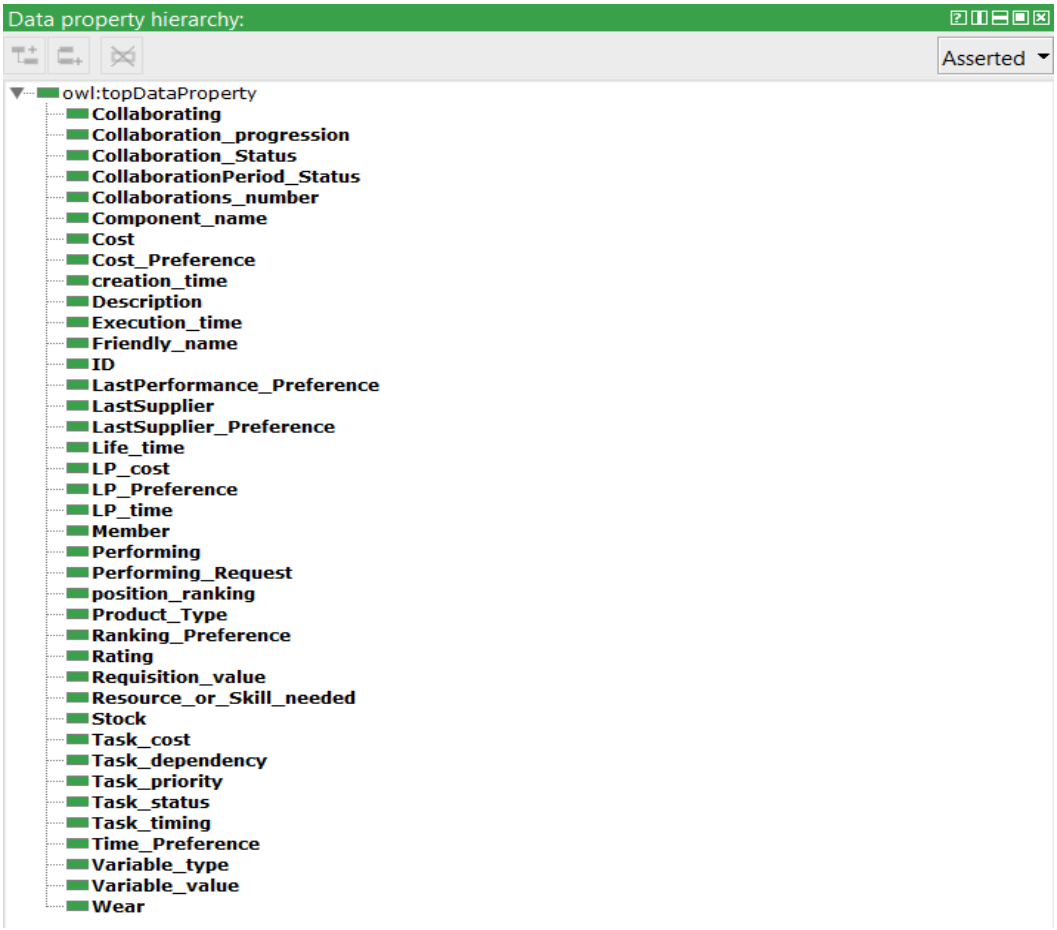


Figura 5.5-Data property hierarchy.

A Tabela 5.5 sumariza as principais entidades da ontologia e as respetivas "*Data Properties*" (mostradas na figura 5.5) que as constituem, mostrando o tipo de dado para cada propriedade.

Tabela 5.5-Entidades e respetivas propriedades.

Class	Data Properties
Supplier	ID: String Friendly_name: String Creation_time: Long
Collaboration_Team	ID: String Friendly_name: String Collaboration_Status: Integer Collaboration_progression: Float Performing_Request: String
Customer	ID: String Friendly_name: String
Last_performance	ID: String LP_cost: Float LP_time: Float

Preferences	ID: String Time_Preference: boolean Cost_Preference: boolean LastPerformance_Preference: boolean Ranking_Preference: boolean LastSupplier_Preference: boolean LastSupplier: String
Ranking	Collaborations_number: Integer Life_time: Long position_ranking: Integer Rating: Float
Request	ID: String Friendly_name: String Description: String
Resource	ID: String Friendly_name: String Cost: Float Stock: Integer Execution_time: Float
Skill	ID: String Friendly_name: String Cost: Float Execution_time: Float Wear: Float
Status	Collaborating: Integer Performing: Integer
Task	ID: String Friendly_name: String Resource_or_Skill_needed: String Requisition_value: Integer Task_cost: Float Task_dependency: String Task_Priority: Integer Task_status: Integer Task_timing: Float

O *Protégé* possui o OntoGraf, que é uma funcionalidade que possibilita visualizar graficamente as diferentes relações entre as entidades. Podemos observar o aspeto da ontologia na Figura 5.6 e Figura 5.7, nas quais estão presentes, respetivamente, o gráfico e as "*Object Properties*" (representadas como arcos de diferentes cores no gráfico).

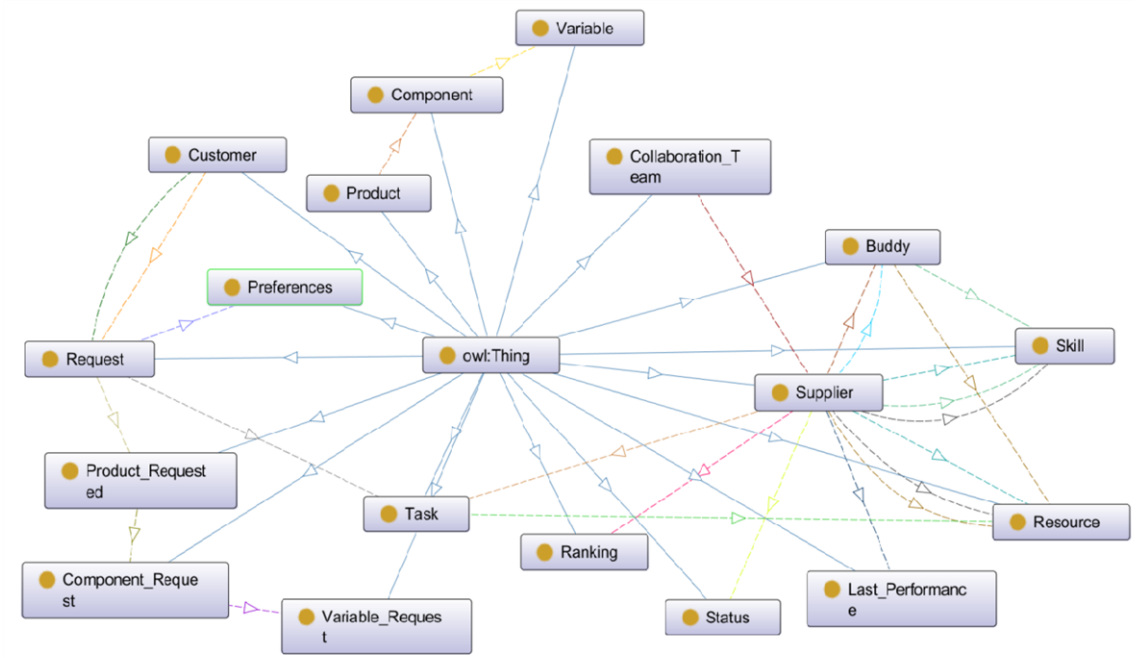


Figura 5.6-OntoGraf.

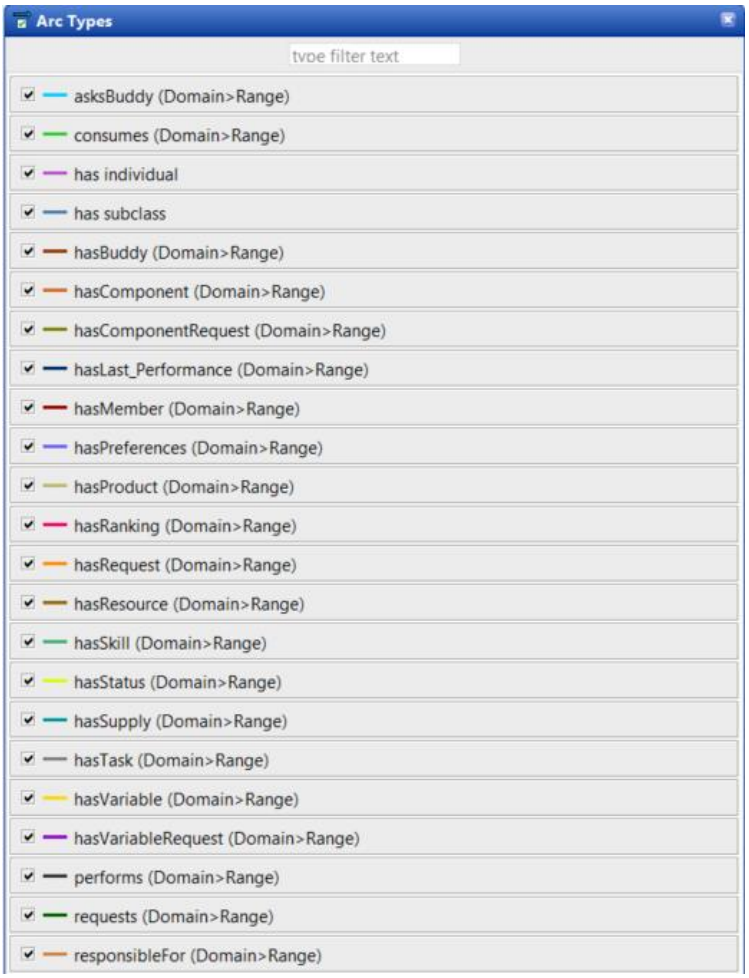


Figura 5.7-OntoGraf Arc Types.

5.2.2 – Sistema Multiagente

Nesta secção serão abordadas as implementações de cada um dos tipos de agente existentes neste sistema: Console Agent, Supplier Agent e Customer Agent. Mostrando, principalmente, os comportamentos (*behaviours*) que constituem os dois últimos mencionados.

5.2.2.1 – Console Agent

Este é o agente responsável por iniciar o protótipo desenvolvido. Este tem as seguintes funcionalidades:

- Lançar todos os agentes presentes no sistema (*Supplier Agents* e *Customer Agents).*
- Apresentar interface para submeter novo *Request.*
- Lançar um novo *Customer Agent.*
- Lançar um novo *Supplier Agent.*

Na Figura 5.8 é descrito o funcionamento deste agente e a sua relação com a Ontologia (base de conhecimento).

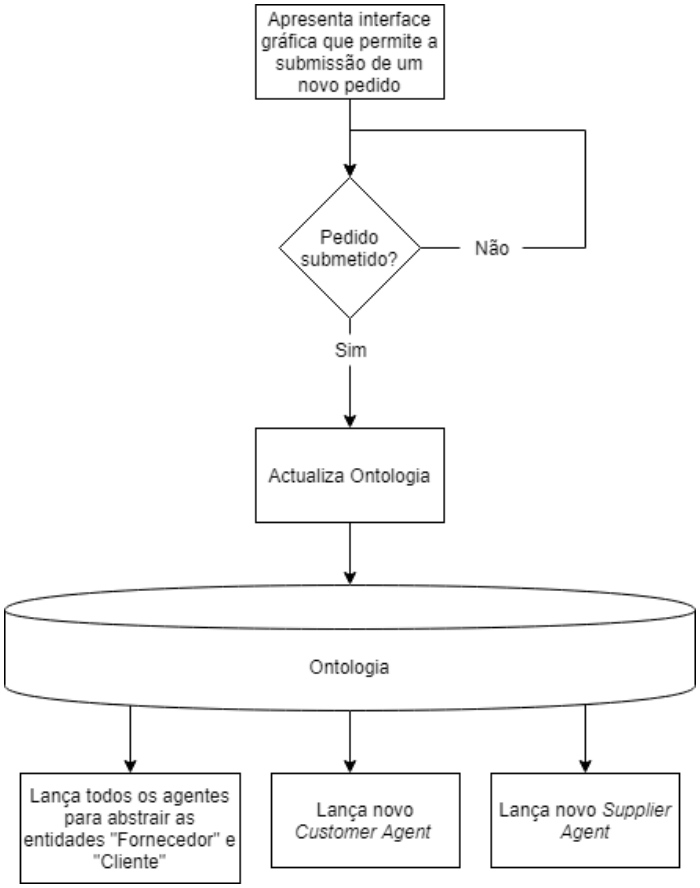


Figura 5.8-Esquemático do Console Agent.

Como pode ser observado, uma das funcionalidades deste agente é a exposição da interface gráfica (apresentada no capítulo seguinte) que permite a submissão de um novo pedido. Após a submissão do pedido, os dados referentes ao pedido, ao cliente que o efetuou e às suas preferências são criados na Ontologia.

De modo a iniciar a plataforma, este agente possui a funcionalidade de lançar os agentes necessários para abstrair todas as instâncias relativas às classes "*Supplier*" e "*Customer*" presentes na ontologia.

A funcionalidade "Lança novo *Customer Agent"* é aplicada sempre que é submetido um novo pedido no sistema. Para tal é necessário, também, aceder à ontologia para o agente adquirir os dados relativos ao novo *Customer.*

A funcionalidade "Lança novo *Supplier Agent*" surge quando é inserida uma nova máquina/equipamento em determinado cenário, de maneira a abstrai-la.

5.2.2.2 – Supplier Agent

Este é o agente responsável por abstrair cada entidade "Fornecedor" do sistema, que representa uma máquina/equipamento do ambiente de manufatura em que se encontra. Na Figura 5.8 encontra-se esquematizado o funcionamento geral deste agente, baseado nos *behaviours* implementados, que serão descritos posteriormente.

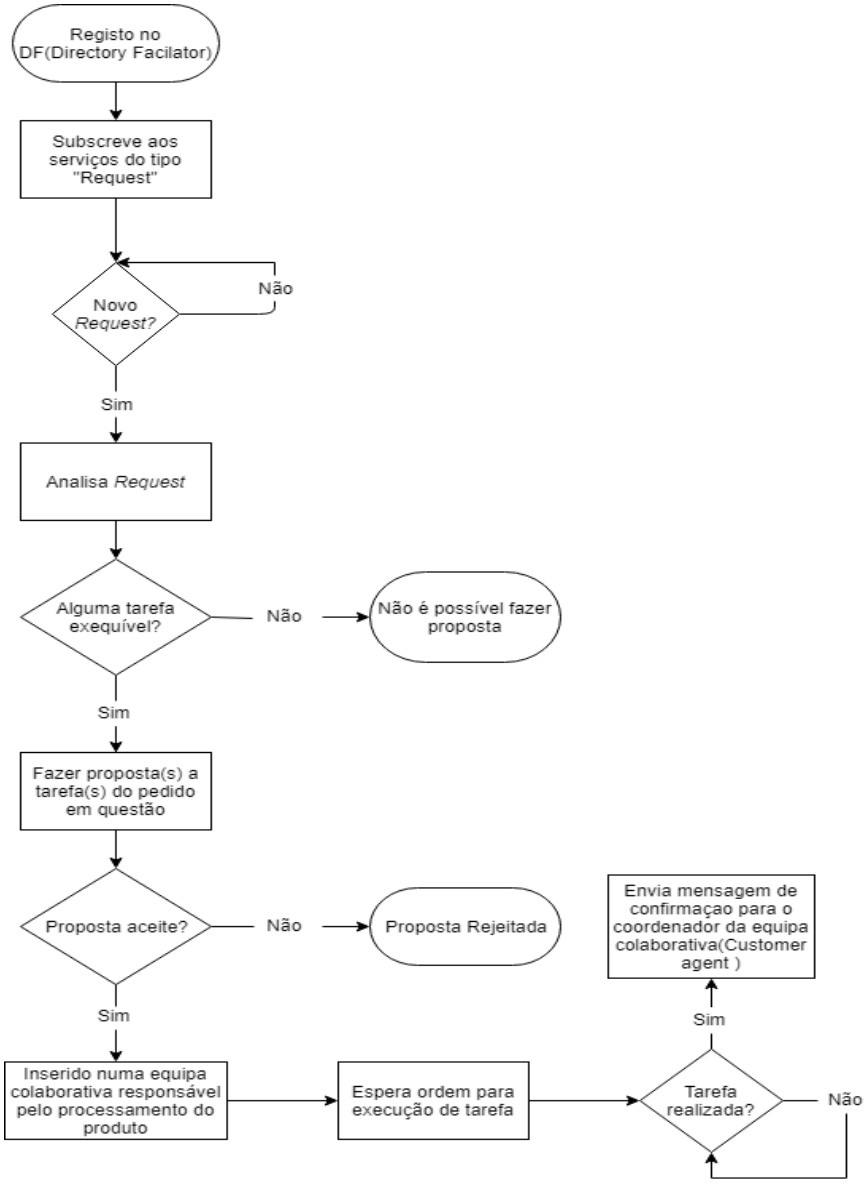


Figura 5.9-Esquemático do Supplier Agent.

Não é possível constatar na Figura 5.9, mas é importante referir que é possível um Supplier Agent fazer a análise de um novo pedido, assim como qualquer processo depois deste, sempre que exista um novo serviço do tipo "*Request*". Isto deve-se ao facto destes processos corresponderem aos *behaviours* deste agente, que permitem ser executados em paralelo uns com os outros. Ou seja, por exemplo, caso esteja a ser realizada uma proposta a uma tarefa e nesse momento surja um novo "*Request",* o agente realiza os processos paralelamente.

Em primeiro lugar, é importante explicar o serviço do JADE que funciona como páginas amarelas. Este é o DF (*Directory Facilitator*) e permite que todos os agentes se registem com um AID (*Agent Identifier*) e com os serviços que podem oferecer aos outros agentes presentes no sistema. Este serviço fornece, como já foi referido, métodos de registo, procura, subscrição, etc. Dos quais, o método de registo é efetuado por todos os agentes na sua inicialização de modo a estarem acessíveis no DF.

Vão ser apresentadas, tomando um seguimento do esquemático da Figura 5.8, as funcionalidades (já discutidas no capítulo anterior) que este agente possui.

Foi utilizado o método de subscrição de modo a satisfazer a funcionalidade "Subscreve aos serviços do tipo "*Request*"", representada no esquemático da Figura 5.9.

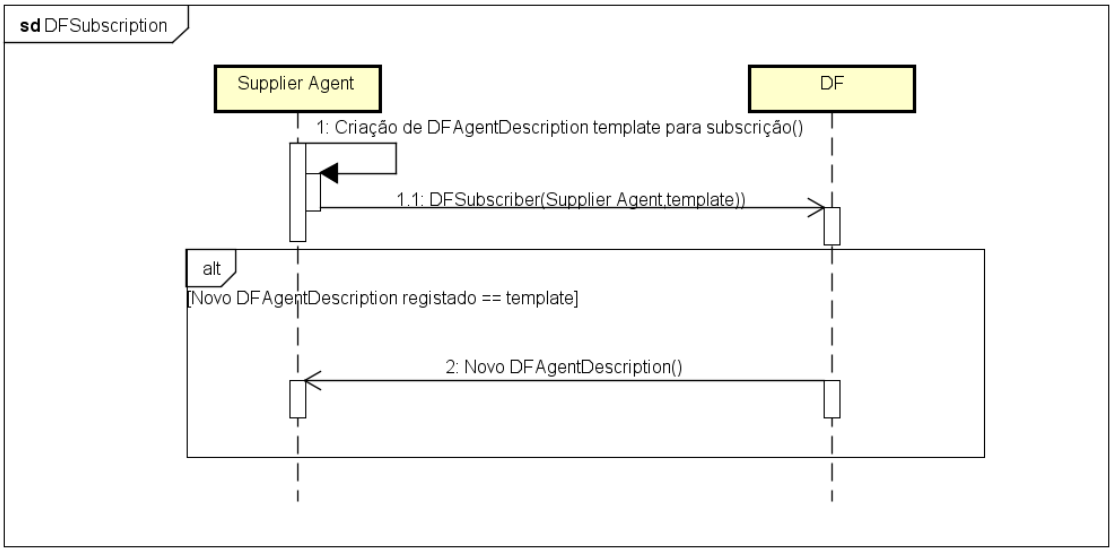


Figura 5.10-Diagrama de sequência – DFSubscriber behaviour.

Na Figura 5.10 encontra-se o diagrama de sequência descritivo da implementação do *behaviour DFSubscriber.*

Na interação 1, é criado um template do tipo DFAgentDescription (método que permite descrever um agente e os seus serviços), no qual tem incluído um serviço do tipo "*Request*".

Na interação 1.1, o *Supplier Agent* realiza a subscrição ao DF de modo a ser informado quando algum agente com o tipo de serviço indicado no template se regista.

A interação 2 mostra a correspondência do DF, à subscrição do *Supplier Agent,* com a DFAgentDescription do novo agente registado, que neste caso é o *Customer Agent* com o pedido de um cliente.

Uma vez que existe um novo "*Request",* este agente prossegue para uma análise do mesmo. Nesta análise, é feita uma verificação, a cada tarefa do pedido, de maneira a inferir se possui algum *Resource* ou *Skill* correspondente ao necessário para a executar.

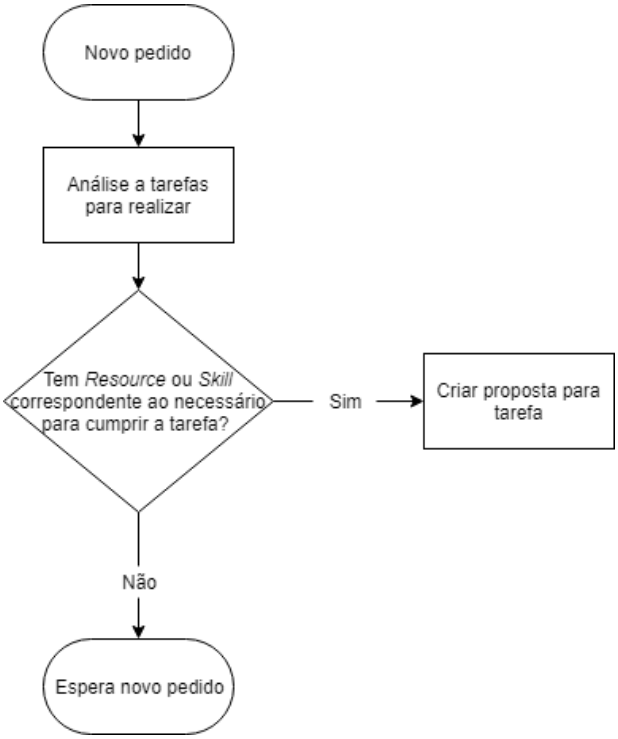


Figura 5.11-Fluxograma – Análise de novo pedido.

Na Figura 5.11 é esquematizado o processo de análise de um novo pedido pelo *Supplier Agent*. É uma análise simples, a qual é realizada por um algoritmo que percorre todas as tarefas a realizar para satisfazer o pedido. Caso o agente tenha um *Resource* ou *Skill* correspondente ao requerido para execução de dada tarefa, prossegue para a criação de uma proposta.

O processo de criação de proposta é realizado sempre que, depois da análise de um pedido, existem tarefas possíveis de executar. Este é suportado por uma classe *Offer,* que permite a criação de um objeto *Offer* com os atributos descritos na Tabela 5.6.

Tabela 5.6-Classe Offer.

Objeto	Atributos	Descrição
Offer	Task_ID	ID da tarefa para qual se deseja criar proposta.
	Execution_time	Estimativa do tempo de execução previsto para a tarefa, baseado na *Resource* ou *Skill* que vai ser utilizada.
	Cost	Custo acrescido ao pedido, baseado no custo da *Resource* ou *Skill* que é necessário para a tarefa.

	Last_Performance	Valores do custo e do tempo de execução da ultima execução realizada pelo agente que está a criar a proposta.
	Ranking_position	Posição do agente representante desta proposta, no ranking de "*Supplier Agents" (*funcionalidade explicada no capítulo anterior*).*

Depois da proposta criada esta é enviada para o *Customer Agent* responsável pelo pedido.

O processo de envio da proposta é efetuada através do Protocolo FIPA *Contract Net,* que é utilizado quando se pretende negociar com todos os agentes que conseguem executar um determinado serviço. Neste caso, a negociação é iniciada pelo *Customer Agent (Initiator),* uma vez que é quem contacta todos os agentes do tipo "*Supplier Agent" (Participant)* de modo a obter propostas para as tarefas a realizar. É importante referir que, apesar do *Customer Agent* ser quem inicia esta negociação, um *Supplier Agent* toma conhecimento do pedido a partir do *behaviour* DFSubscriber anteriormente explicado e somente responde ao pedido de propostas, feito pelo *Customer Agent,* quando tem uma proposta criada.

Deste modo, na Figura 5.12 podem-se observar as interações realizadas no protocolo de negociação implementado, na perspetiva de *Supplier Agent* (*Participant*).

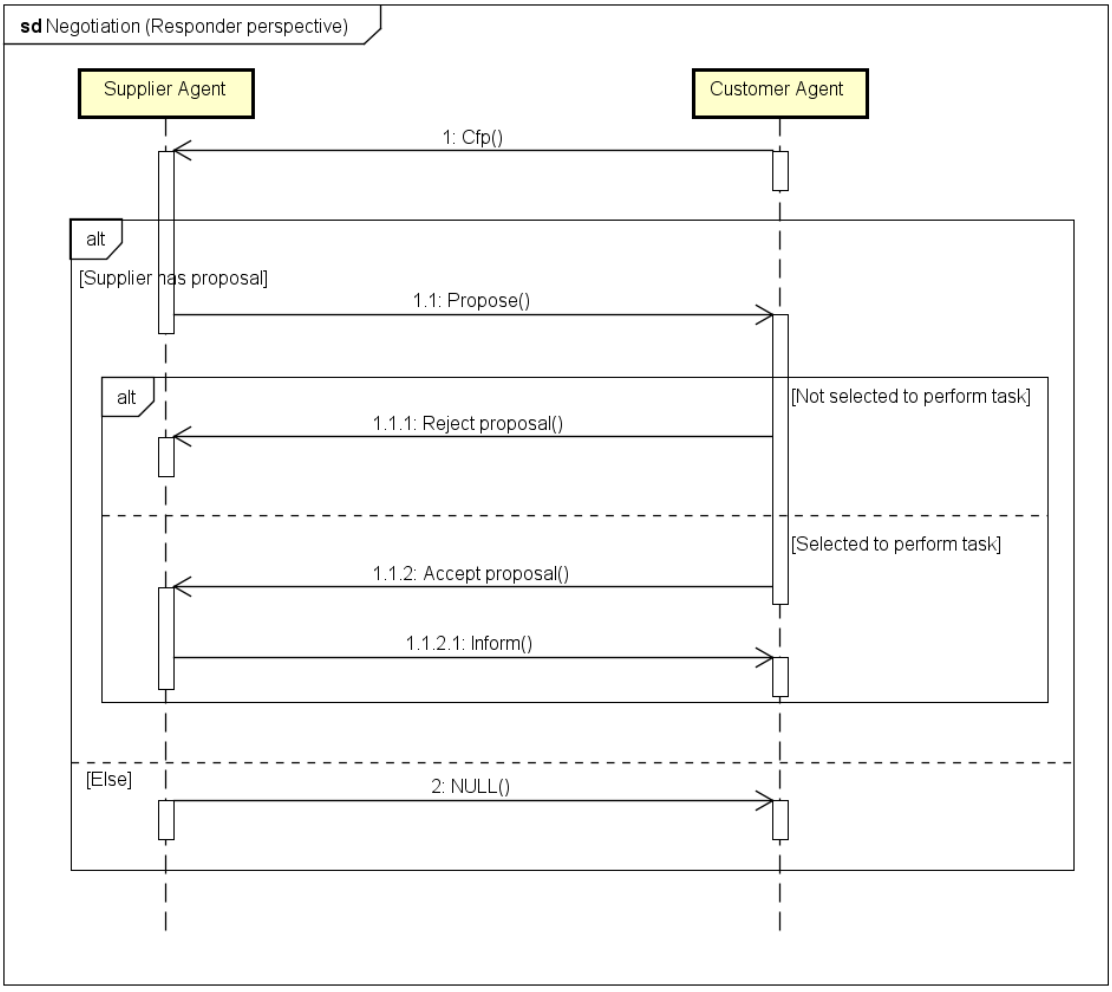


Figura 5.12-Diagrama de sequência – ContractNet Responder.

Como referido, na interação 1 da figura pode-se observar a iniciação da negociação pelo Customer Agent descrita como Cfp (Call for proposal). Esta é meramente utilizada devido á sua obrigatoriedade para iniciar esta negociação, dado que o objetivo inicial seria o *Supplier Agent* iniciá-lo com a sua proposta, depois de ter recebido a informação de um novo registo de pedido no DF, a partir do *DFSubscriber behaviour* anteriormente abordado. Ainda assim, foi utilizado este protocolo devido à sua capacidade de lidar com negociações 1:N (um para muitos), permitindo que um *Customer Agent* lide com múltiplas propostas às tarefas do pedido pelo qual é responsável.

Caso o *proposal* de um *Supplier Agent* seja aceite (interação 1.1.2), este fica responsável pela tarefa à qual fez a proposta e é inserido numa equipa colaborativa, da qual fazem parte os responsáveis por todas as tarefas necessárias para a produção do pedido do cliente. O *Customer Agent* é o responsável por criar a equipa e coordenar as tarefas para a criação de um produto.

Tal como observado anteriormente (Fig.5.9), depois de inserido numa equipa, um *Supplier Agent* espera uma ordem do coordenador da equipa para executar a tarefa, enviando uma mensagem de confirmação quando a execução da mesma estiver completa. Estas funcionalidades serão demonstradas posteriormente.

Este tipo de agente possui também as funcionalidades, tal como referido no capítulo anterior, "pedir *stock*" e "disponibilizar *stock*". Este pedido e disponibilização de stock é realizada, evidentemente, por dois agentes do tipo *Supplier Agent* e tem por base um protocolo de interação da FIPA, denominado *FIPA Request Protocol*, de 1:1 (um para um). A estrutura deste protocolo tem um *Initiator* e um *Responder*, sendo, respetivamente, quem pede o *stock* e quem disponibiliza-o (ou não). Na Figura 5.13 pode-se observar as interações realizadas no protocolo implementado.

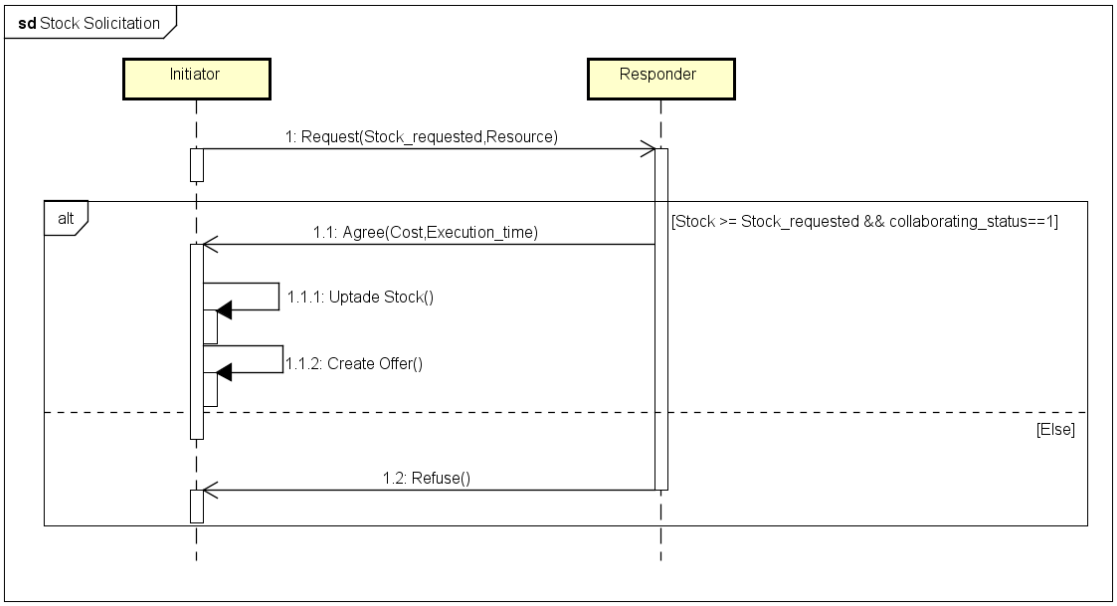


Figura 5.13-Diagrama de sequência – Stock Request.

Na interação 1 o *Initiator* faz o pedido de *stock* para uma *Resource* específica. Isto acontece quando um *Supplier Agent* analisa um pedido e repara que tem uma *Resource* que corresponde à necessária para a execução da tarefa, mas não tem *stock* para poder criar uma proposta.

A interação 1.1 acontece caso um agente *Responder (Buddy)* disponha do *stock* solicitado e esteja de momento a colaborar num pedido (collaboration_status==1).

As interações 1.1.1 e 1.1.2 são, respetivamente, o *update* do *stock* do agente *Initiator,* de modo a tornar possível a criação da oferta para a tarefa desejada, e a criação da oferta.

A interação 1.2 acontece caso não seja possível que um agente *Responder* disponibilize *stock.*

É importante referir que os *Agent Suppliers* que realizam este comportamento estão relacionados como "*Buddys"* (conceito explicado no capítulo anterior), sabendo previamente os tipos de *Resources* que cada um tem de modo a poder pedir *stock* relativo um em específico.

5.2.2.3 – Customer Agent

Este é o agente responsável por abstrair cada entidade "Cliente" do sistema. Na Figura 5.14 encontra-se esquematizado o funcionamento geral deste agente, baseado nos *behaviours* implementados, que serão explicados posteriormente.

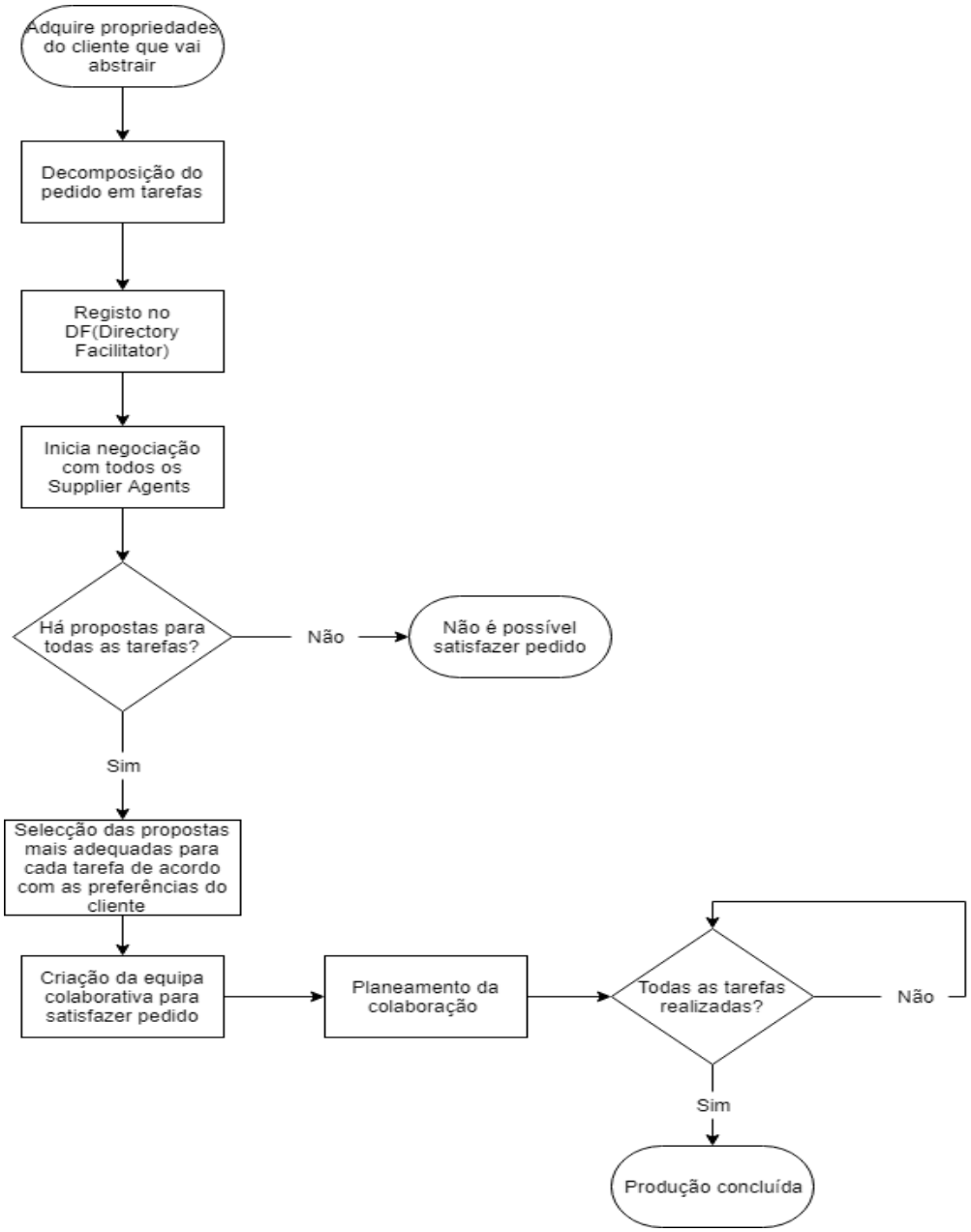


Figura 5.14-Esquemático do Customer Agent.

Após a solicitação do pedido de um cliente, é lançado um *Customer Agent* no sistema responsável por abstrair este cliente.

Primeiramente, adquire todas as propriedades do *Customer*, ou seja, os dados do cliente, o pedido e as preferências em relação a este pedido. Toda esta informação é inserida pelo cliente na realização do pedido.

Seguidamente, é realizada uma decomposição do pedido em diferentes tarefas (cada uma definida como *Task*). Assim, foi criada uma interface gráfica de modo a poder simular a decomposição do pedido em tarefas realizada pela ação humana.

É registado no DF um serviço do tipo "*Request",* tendo como propriedades as tarefas necessárias para a sua realização. Deste modo, os *Supplier Agents* são informados da existência deste novo pedido através do *behaviour DFSubscriber.*

De maneira a obter *Supplier Agents* para a execução das tarefas necessárias para satisfazer o pedido, foi implementada a negociação já referida anteriormente. A Figura 5.15 mostra as interações deste protocolo na perspetiva de *Customer Agent (Initiator).*

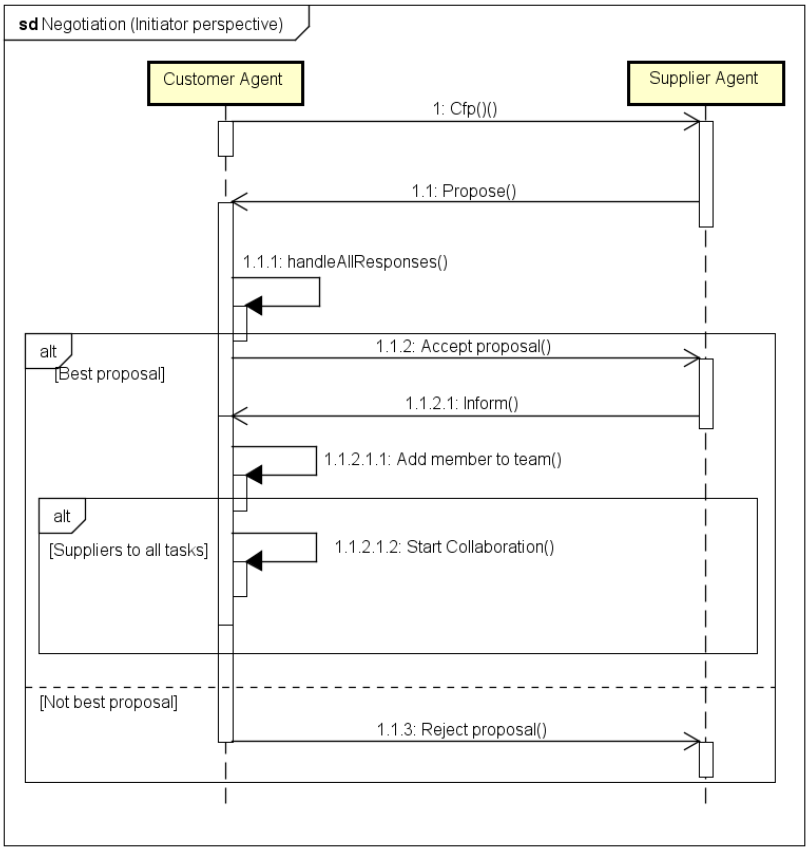


Figura 5.15-Diagrama de sequência – Contract Net Initiator.

Após a criação da equipa de colaboração responsável pelo pedido, o *Customer Agent* é, também, responsável por planear a colaboração. Para tal foi criado um conjunto de *behaviours* responsáveis pela coordenação da colaboração, de forma a

enviar pedidos de execução da tarefa que compete a cada membro da equipa, sequencialmente e ordenadamente. Este processo está implementado da seguinte forma:

1) Os *Supplier Agents* são ordenados por ordem da prioridade (propriedade da tarefa) da task pela qual são responsáveis. Ou seja, por exemplo, se um *Supplier Agent* for responsável por uma tarefa cuja prioridade é 2, então este é colocado em segundo.
2) É enviado um pedido, a cada *Supplier Agent,* para execução da tarefa pela qual é responsável, conforme a ordem criada.
3) Apenas é enviado um novo pedido de execução quando o anterior tiver recebido a informação de que está completo.

O ponto 1) foi implementado através de um algoritmo que ordena os *Supplier Agents* num Array.

O ponto 2) foi implementado por meio de um *FIPA Request Protocol,* em que é permitida a comunicação ponto a ponto (1:1) entre agentes. Neste caso, o *Customer Agent (*responsável pela equipa de colaboração*)* toma o papel de *Initiator,* enviando o pedido de execução e o *Supplier Agent* que se pretende contatar toma o papel de *Responder,* respondendo uma vez que a execução esteja completa.

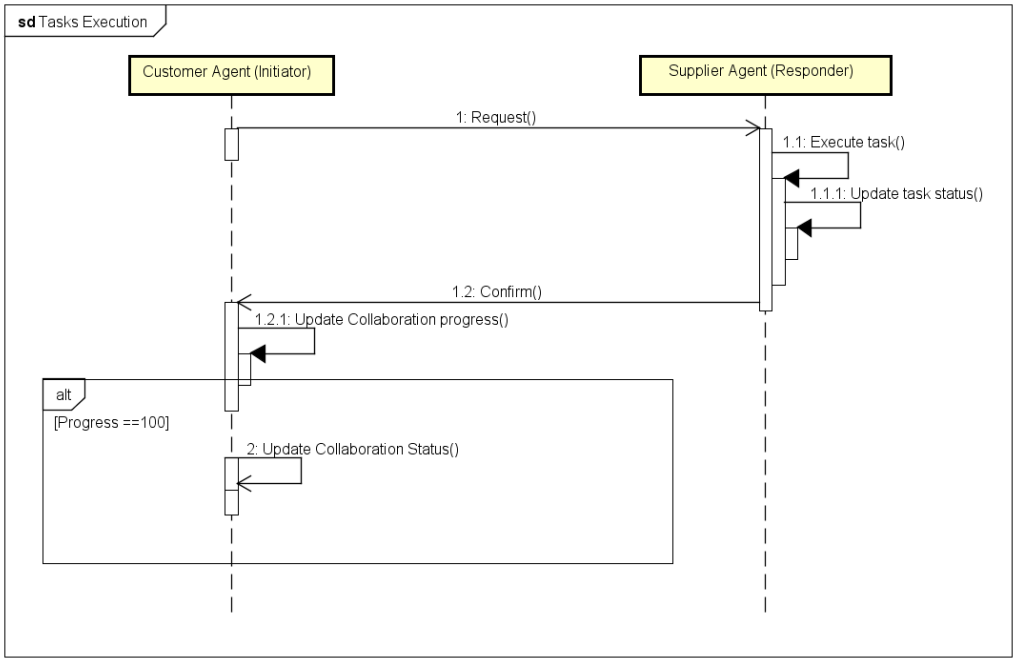


Figura 5.16-Diagrama de sequência – FIPA Request Protocol – Tasks Execution.

Na interação 1, o *Customer Agent* envia uma mensagem com a performativa *Request* para o *Supplier Agent* que pretende que execute a tarefa.

Quando o *Supplier Agent* recebe este pedido, executa a tarefa e atualiza o estado da tarefa para o valor correspondente a "feita". Na realidade, este ato é uma simulação do tempo de execução da *Resource* ou *Skill* que a tarefa necessita que seja executada.

A interação 1.2 é uma mensagem de confirmação da execução da tarefa enviada pelo *Supplier Agent.* Quando o *Customer Agent* a recebe, atualiza o progresso da colaboração e, caso esteja finalizada (Progress==100), atualiza o estado da colaboração para o valor correspondente a "feita".

A funcionalidade descrita em 3) foi implementada através de um *Sequential Behaviour,* que é um comportamento que permite a adição de sub comportamentos que são executados sequencialmente e apenas quando o anterior estiver completo. Cada sub comportamento corresponde ao *behaviour* implementado para enviar mensagens no protocolo descrito em 2). Desta maneira, as mensagens *Request* para execução de tarefas são enviadas sequencialmente e apenas quando a tarefa anterior estiver completa.

Capítulo 6

6. Validação

Este capítulo pretende validar o funcionamento do protótipo desenvolvido, apresentando todas as interfaces gráficas implementadas.

De modo a validar as funcionalidades que constituem este protótipo é simulado um cenário no âmbito da indústria moderna.

6.1 Cenário de Aplicação

De maneira a testar a aplicação do protótipo desenvolvida, considere-se o seguinte cenário:

- No âmbito de uma fábrica de telemóveis customizáveis, na qual os telemóveis são produzidos a partir da junção dos seus vários constituintes, ou seja pratica um tipo de produção discreta, é conveniente que exista um sistema a fim de aumentar a eficiência e agilidade da fábrica, de maneira a ser capaz de disputar a competitividade do mercado.
- Nesta fábrica encontram-se presentes máquinas/equipamentos, em que cada tem determinados *Resources* ou *Skills* de maneira a contribuir para a produção de um telemóvel.
- Um cliente tem a possibilidade de fazer um pedido à fábrica, escolhendo os componentes que o constituirão.

Para a simulação da produção de um telemóvel serão considerados as seguintes componentes fictícias:

Tabela 6.1-Componentes considerados no cenário.

Componente	Variáveis
Ecrã	- LCD - LED
CPU	- Snapdragon710 - Snapdragon660 - Snapdragon670
Memória	- 32GB - 64GB - 128GB
Câmara	- 10MP - 16MP - 24MP
Bateria	- 3000mAh - 4000mAh

Os componentes descritos na Tabela 6.1 representam, também, os tipos de *Resources* que as diferentes máquinas/equipamentos dispõem. Sendo que também existe a possibilidade de disporem de *Skills*, serão consideradas algumas, tais como, aparafusar e soldar, de forma a validar esta entidade.

6.2 Simulações do sistema

Visto que o sistema irá comportar-se de acordo com a constituição da fábrica, ou seja, quantas e quais as máquinas que dela fazem parte, foram criadas simulações para diferentes casos para demonstrar as funcionalidades do protótipo implementado.

6.2.1 Primeira Simulação

Nesta simulação pretende-se fazer uma demonstração simples do protótipo implementado, apresentando as suas principais funcionalidades. Desta forma, as máquinas/equipamentos presentes neste caso serão proporcionais às componentes deste cenário. Ou seja, cada máquina/equipamento possui uma *Resource* ou *Skill* diferente de todas as outras.

Na Figura 6.1 pode-se observar as instâncias criadas na ontologia relativas a cada equipamento com a respetiva *Resurce* ou *Skill*.

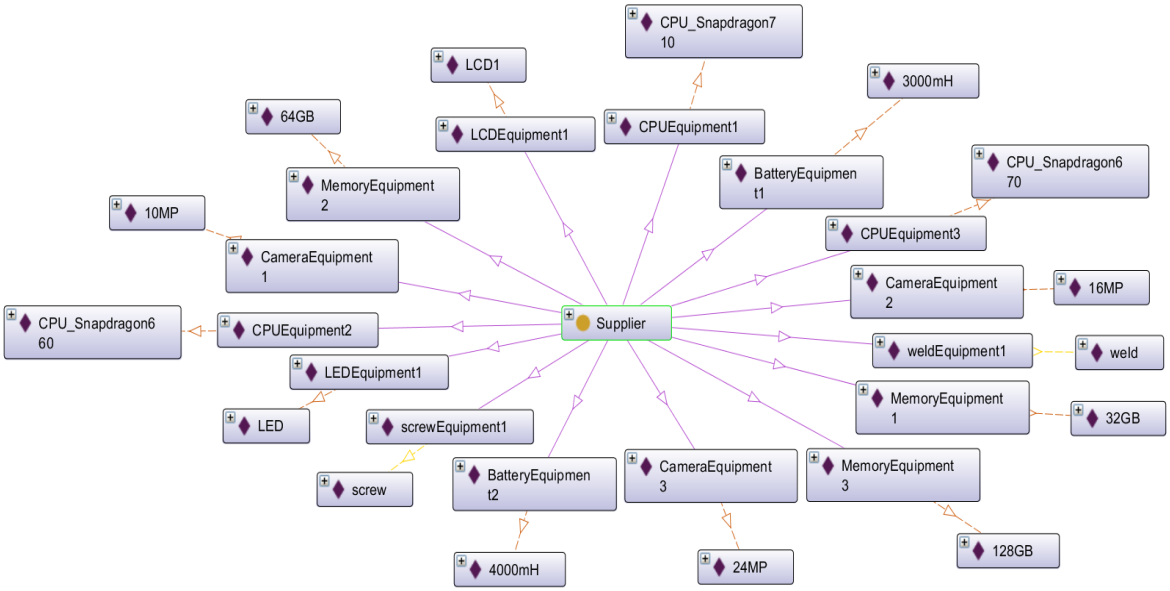


Figura 6.1-Primeira Simulação – instâncias de Suppliers com respetivos Resources e Skills.

Deste modo, iniciando o protótipo, a primeira janela que aparece é a que permite ao cliente a solicitação de um pedido. Na Figura 6.2 é apresentada esta janela com o Produto telemóvel, com os componentes e variáveis correspondentes.

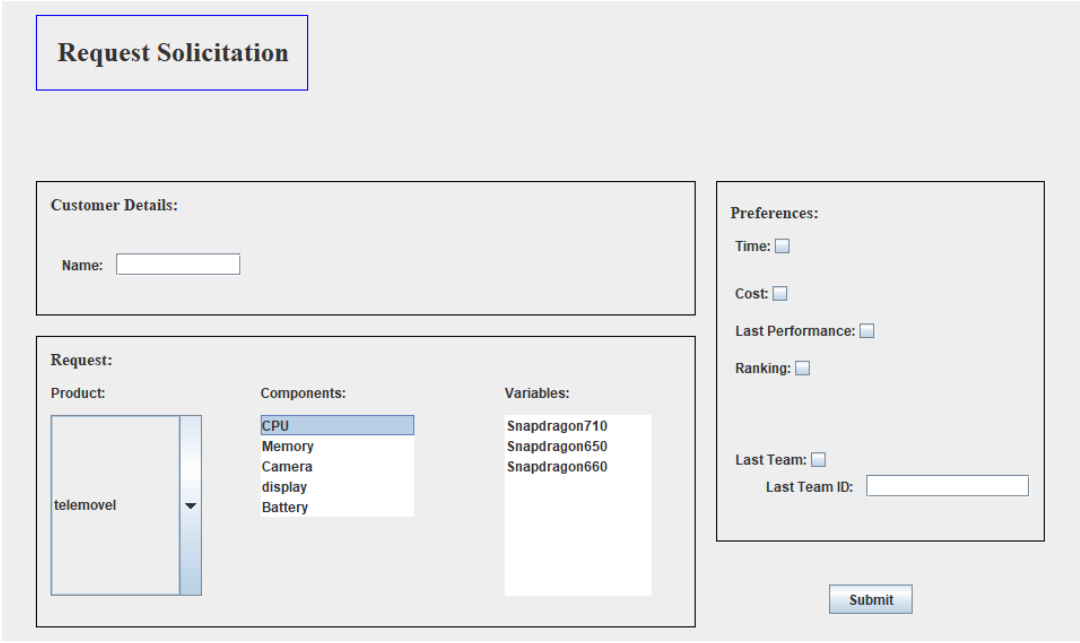


Figura 6.2-Janela de solicitação de pedido.

Dado que esta interface foi principalmente implementada com o objetivo de ser possível simular o sistema, relativamente aos dados pessoais do cliente, apenas foram considerados o ID (criado aleatoriamente) e o nome.

Nota: Como pode ser observado, é permitida a escolha da preferência que tem pelo pedido realizado. No entanto, esta escolha não vai influenciar em nada a escolha dos agentes responsáveis para processar o pedido devido ao facto de apenas existir um *Supplier* para cada tipo de *Resource.* Assim, para esta simulação não foi escolhida nenhuma preferência.

Uma vez que o pedido seja submetido, aparece a janela referente à decomposição do pedido em tarefas realizado pela ação humana. Na Figura 6.3 é apresentado como exemplo a decomposição de um pedido e, consequentemente, a criação das tarefas para o satisfazer.

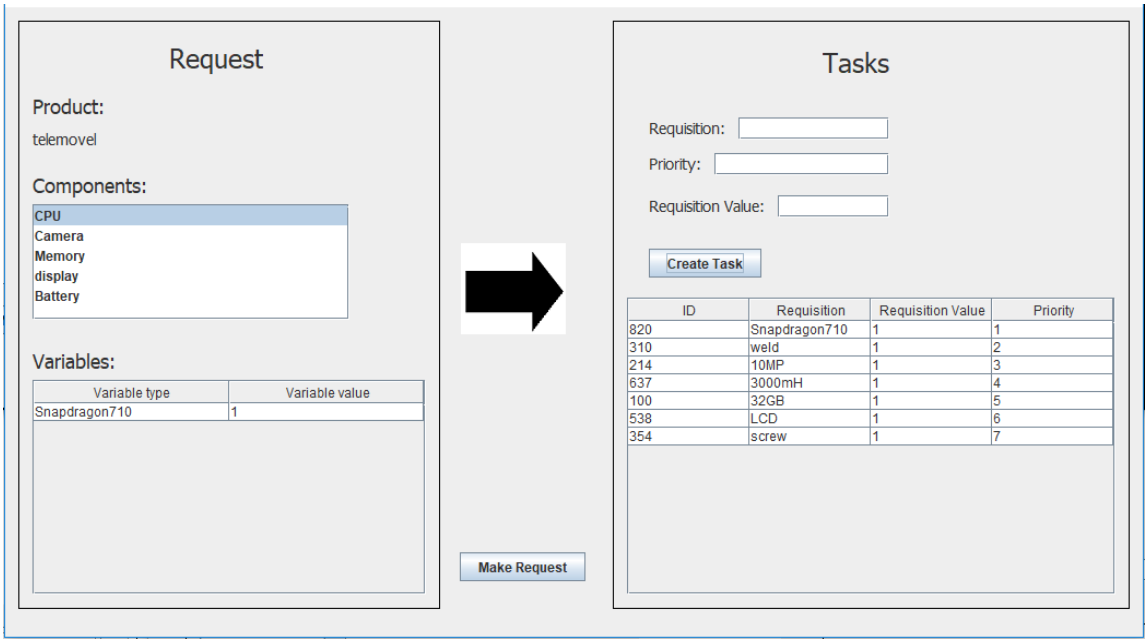


Figura 6.3-Janela de decomposição do pedido.

No exemplo acima, à esquerda é descrito o pedido efetuado, podendo ser selecionado cada tipo de componente para verificar qual foi a escolha do cliente. À direita é efetuada a criação das tarefas, as quais são caracterizadas pelo ID, requisito (*Resource* e *Skill*), prioridade e valor requisitado (identificado pelo *Variable value* que, neste caso, corresponde ao valor da variável "quantidade").

Pode-se observar que também foram criadas 2 tarefas relativas ao uso de *Skills,* sendo que estas também necessitam de ser criadas nesta etapa de decomposição do pedido pela ação humana.

Depois do pedido ser decomposto em tarefas, é apresentada uma janela que tem como objetivo a monitorização dos equipamentos, pedidos e colaborações. Para este caso, a janela que aparece assim que o botão "Make Request" é pressionado, é a mostrada na Figura 6.4.

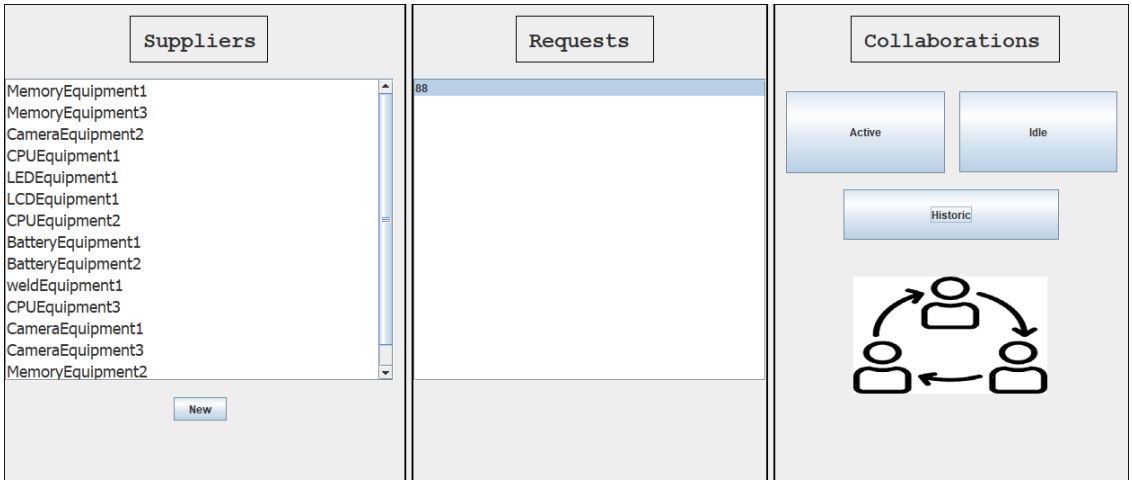


Figura 6.4-Janela de monitorização.

Nesta janela é possível ver os *Suppliers* que atualmente pertencem à fábrica, sendo possível ver uma descrição de cada um deles. Na Figura 6.5 é usado o exemplo da descrição do CPUEquipment1.

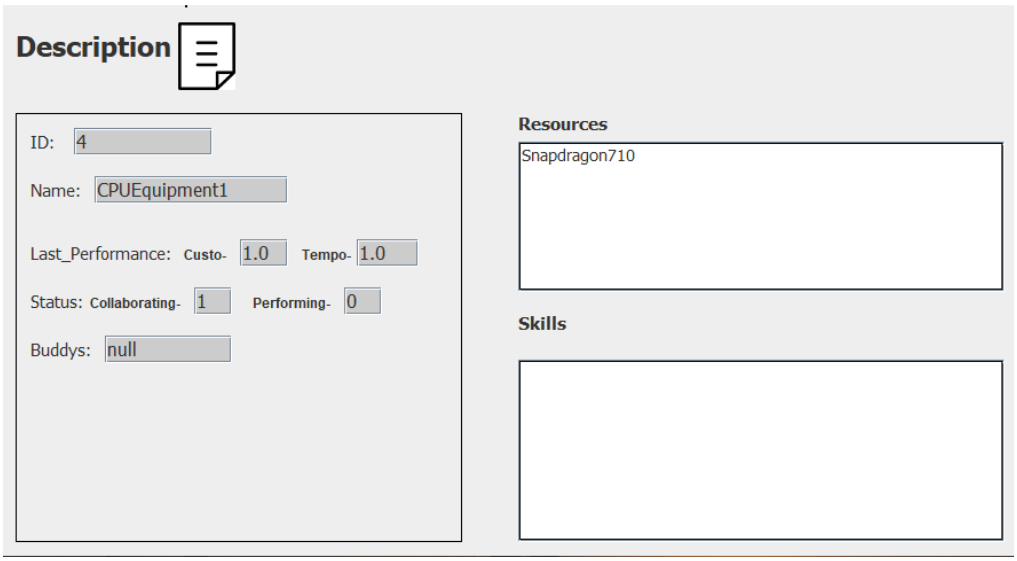


Figura 6.5-Janela descrição de Supplier.

Também é permitida a criação de um novo *Supplier*, caso, por exemplo, seja necessário acrescentar ao cenário uma nova máquina/equipamento (Fig. 6.6).

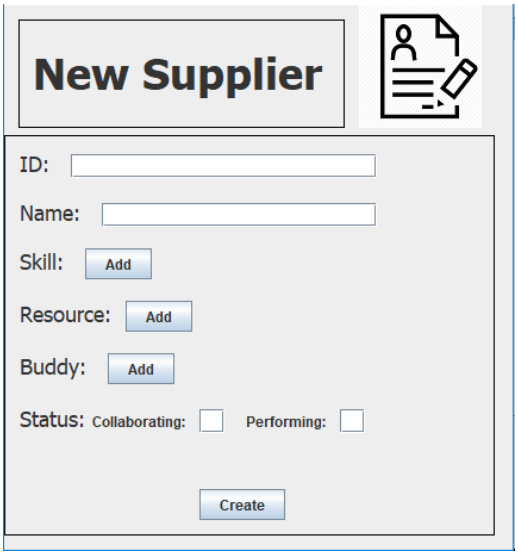


Figura 6.6-Janela New Supplier.

De maneira a associar *Skills, Resources* ou *Buddys* a um novo *Supplier,* quando pressionados os botões "*Add*" é apresentada uma nova janela para descrever, respetivamente, cada um deles (Figura 6.7).

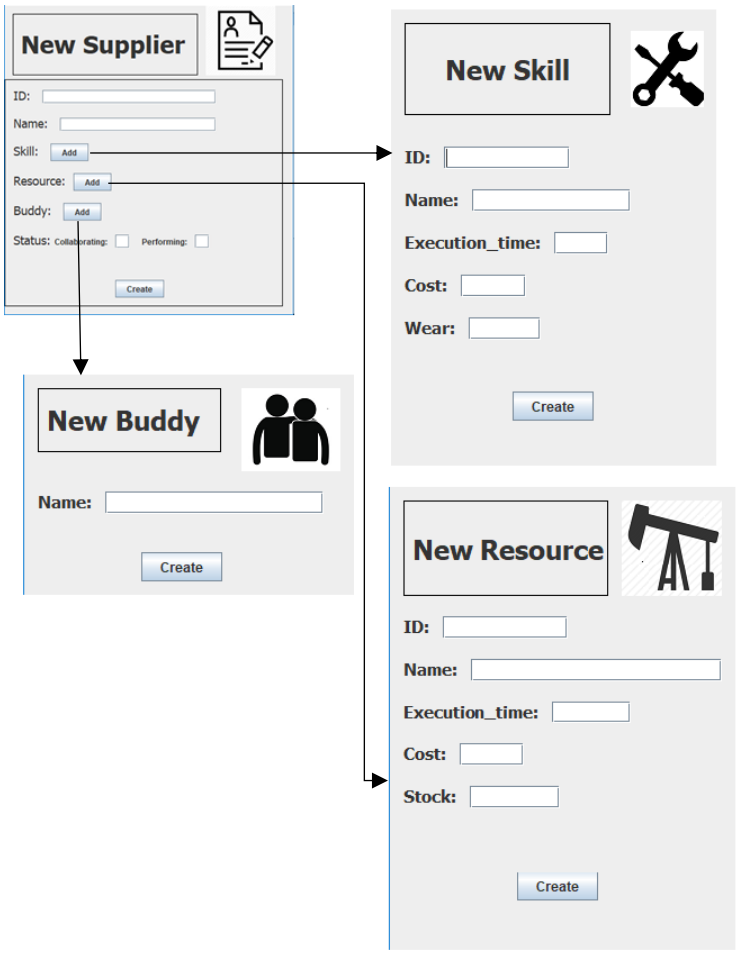


Figura 6.7-Janelas – Associar Skill, Resource ou Buddy.

Nota: É de realçar que a associação de um novo *Buddy* é feita apenas através do nome, sendo que é o suficiente para um *Supplier* reconhece-lo.

Também é possível ver uma simples descrição de cada um dos pedidos presentes no sistema, neste caso apenas está presente o pedido com o ID = 88 (gerado automaticamente), efetuado pelo cliente fictício, cujo nome é "Carlos" (Fig.6.8).

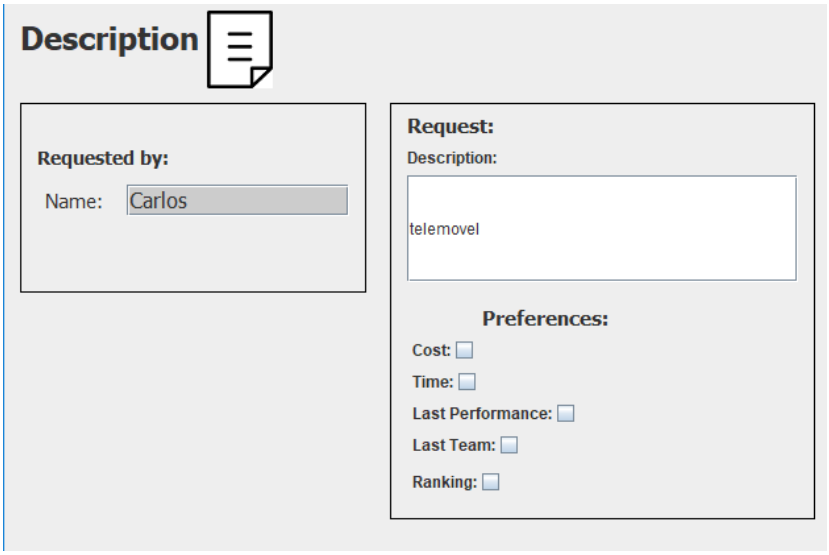
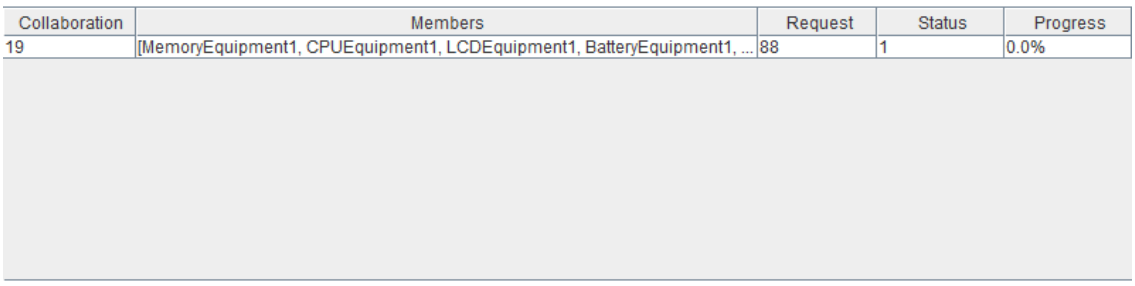


Figura 6.8-Janela – Descrição de pedido.

Uma vez que a equipa colaborativa responsável por satisfazer este pedido seja criada, a sua descrição é listada na janela das colaborações ativas (janela apresentada quando o botão "*Active*" da Figura 6.4 é pressionado). Na Figura 6.9 é exibida a janela para este caso.



Collaboration	Members	Request	Status	Progress
19	[MemoryEquipment1, CPUEquipment1, LCDEquipment1, BatteryEquipment1, ...	88	1	0.0%

Figura 6.9-Janela – Colaborações ativas.

Na descrição da colaboração ativa é possível observar, pela respetiva ordem, o ID da equipa de colaboração, os respetivos membros, o ID do pedido, o estado da colaboração (o valor 1 significa ativa), e o progresso.

O processo de criação da equipa responsável por um pedido é realizado através do Protocolo *FIPA Contract Net,* no qual são realizadas as propostas às diferentes tarefas pelos diferentes *SupplierAgents* e a aceitação ou rejeição como resposta do *CustomerAgent*

responsável pelo pedido realizado. É possível observar as interações deste protocolo através da funcionalidade "*Sniffer"* da interface gráfica do JADE (Fig.6.10).

Devido a esta funcionalidade do JADE representar a *lifeline* de cada agente pelo seu AID (correspondente ao ID associado a cada máquina/equipamento), foi criada a Tabela 6.2 de forma a haver uma melhor compreensão da Figura 6.10, a qual mostra as instâncias presentes na ontologia do sistema com o ID correspondente.

Nota: É importante realçar que as interações mostradas na Figura 6.10 são realizadas pelos agentes responsáveis por abstrair cada uma das instâncias apresentadas na Tabela 6.2, que por sua vez foram exibidas anteriormente na Figura 6.1.

Tabela 6.2-Instâncias da Ontologia abstraídas por agentes.

Nome da Instância	**ID**
Carlos (*Customer)*	133
LCDEquipment1	1
LEDEquipment1	3
CPUEquipment1	4
MemoryEquipment1	5
CameraEquipment1	6
BatteryEquipment1	7
screwEquipment1	8
weldEquipment1	9
CPUEquipment2	10
CPUEquipment3	11
MemoryEquipment2	12
MemoryEquipment3	13
CameraEquipment2	14
CameraEquipment3	15
BatteryEquipment2	16

A instância "Carlos", com o correspondente ID=133, diz respeito ao *Customer* gerado a partir desta simulação. Todas as outras instâncias são relativas à representação das máquinas/equipamentos presentes no ambiente, correspondentes a expansões da classe *Supplier.*

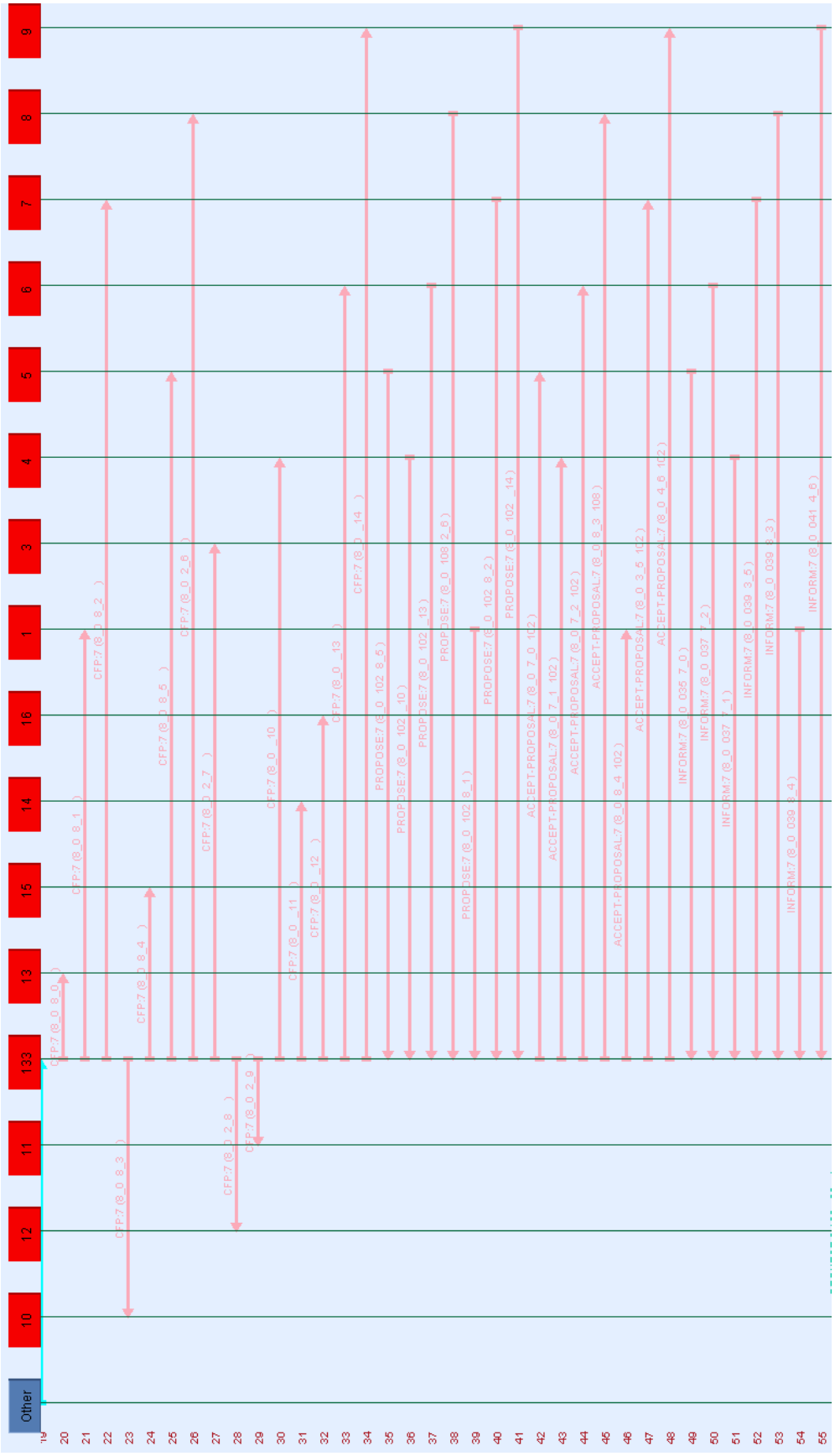


Figura 6.10-JADE- Interações do protocolo FIPA Contract Net.

As interações entre 35 e 41 (inclusive) correspondem às propostas enviadas pelos *Supplier Agents.* Neste caso, apenas sete dos quinze *Supplier* Agents realizaram propostas pelas seguintes razões:

1- Existem, ao todo, sete tarefas que necessitam de ser realizadas para satisfazer o pedido em questão.
2- Como referido, nesta simulação, cada máquina/equipamento tem uma *Resource* ou *Skill* diferente de todas as outras.

Desta forma, as propostas são feitas apenas pelos *Supplier Agents* que representam as máquinas/equipamentos que possuem o requisito que corresponde ao necessário para realizar uma tarefa.

As interações entre 42 e 48 correspondem às respostas, do *Customer Agent*, às propostas mencionadas. Neste caso, são todas com a performativa *Accept Proposal (*aceitação*)* porque apenas existe uma proposta para cada tarefa. Contudo, num caso em que exista duas ou mais propostas para a mesma tarefa, apenas uma é aceite e as restantes são rejeitadas.

As interações entre 49 e 55 correspondem às mensagens enviadas pelos *Supplier Agents* que foram aceites para fazer parte da equipa de colaboração. Esta mensagem tem a performativa *Inform (*informação*)* e é utilizada de maneira a confirmar ao *Customer Agent* a receção da mensagem de aceitação.

Dado que o pedido efetuado já tem uma equipa responsável para o corresponder, esta procede para a sua realização, ou seja, a execução das tarefas. Este processo consiste no envio, sequencial, de mensagens por parte do *Customer Agent* para os responsáveis de cada tarefa. Quando um *Supplier Agent* recebe a mensagem, executa a tarefa e manda uma mensagem de maneira a confirmar a execução (Fig. 6.11).

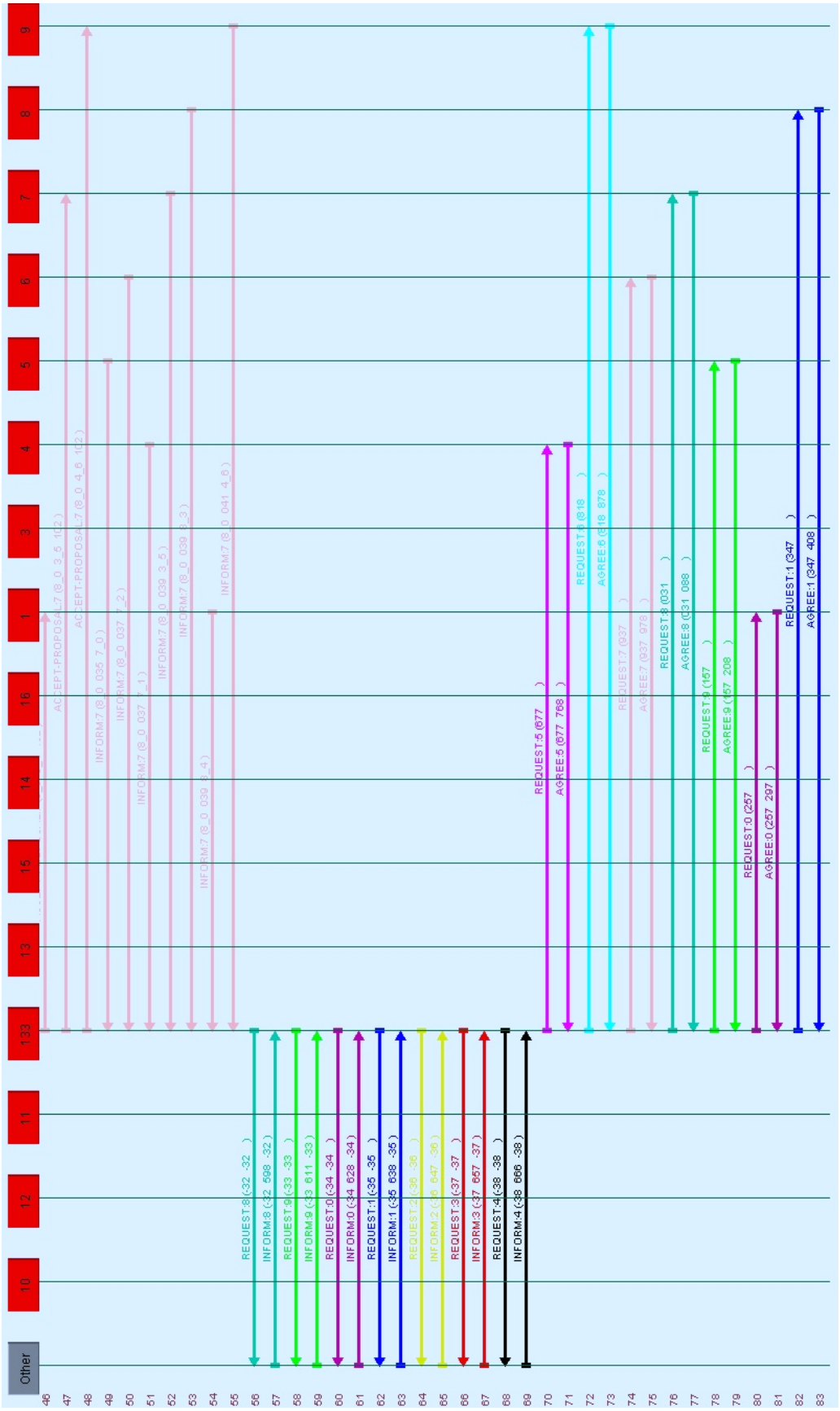


Figura 6.11-JADE- Interações do protocolo FIPA para execução das tarefas.

As interações relativas a este processo estão entre a 70 e a 83.

O agente 133 (*Customer Agent)*, começa por enviar uma mensagem com a performativa *Request* para o agente 4 (*CPUEquipment1).* Esta é respondida com uma mensagem com a performativa *Agree,* de maneira a confirmar a execução da tarefa. Quando o agente 133 recebe esta confirmação, envia uma nova mensagem para o agente responsável por executar a próxima tarefa e assim sucessivamente.

Quando é recebida a confirmação da execução da última tarefa, a colaboração é considerada terminada, são atualizados os estados dos agentes e o estado da colaboração, e esta é apresentada na janela de histórico de colaborações (Fig.6.12).

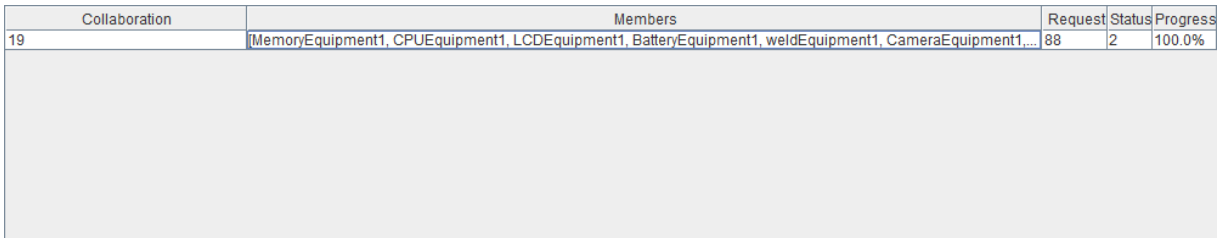


Figura 6.12-Janela – Histórico de Colaborações.

Para finalizar, surge uma janela (Fig.6.13) que permite dar uma classificação ao pedido produzido (perspetiva do cliente). Esta classificação irá, consequentemente, afetar o sistema de *Ranking* entre *Supplier Agents*.

Figura 6.13-Janela – Rating.

6.2.2 Segunda Simulação

Esta simulação visa demonstrar outras funcionalidades que não foram possíveis apresentar com a simulação do caso da secção anterior.

Neste caso considera-se a possibilidade de existir máquinas/equipamentos diferentes, com *Resources* ou *Skills* do mesmo tipo. Isto quer dizer que duas máquinas/equipamentos poderão ter, por exemplo, um *Resource* do mesmo tipo mas com diferentes atributos.

Na Figura 6.14 são apresentadas as instâncias criadas na ontologia para representar cada máquina/equipamento presente no ambiente da fábrica.

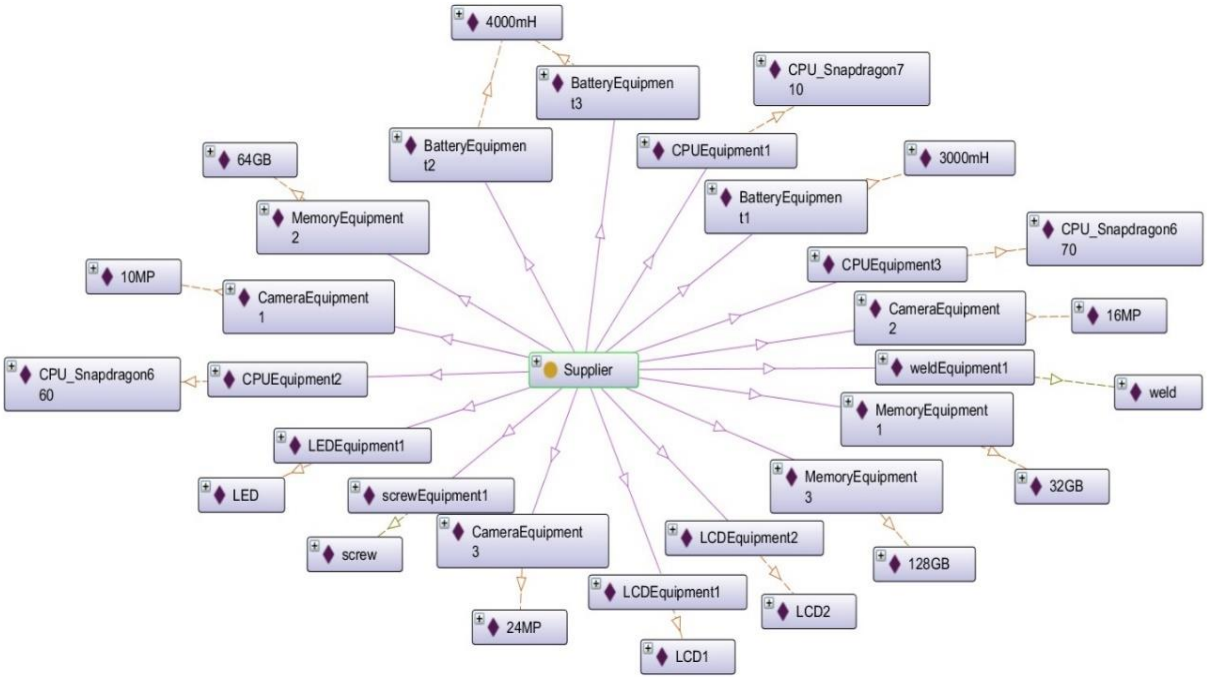


Figura 6.14-Segunda Simulação – instâncias de Suppliers com respetivos Resources e Skills.

É relevante observar que neste caso as instâncias *BatteryEquipment2 e BatteryEquipment3* possuem o mesmo tipo de *Resource,* assim como as instâncias *LCDEquipment1* e *LCDEquipment2*. O motivo pelo qual a *Resource "*4000mH" está associada a ambas as instâncias relativas aos equipamentos das baterias é porque as suas propriedades (*Cost, Execution time,* etc*.)* são exatamente iguais. Por outro lado, as *Resources "LCD1"* e "LCD2" estão associadas separadamente porque possuem o mesmo tipo de *Resource (LCD)* mas as propriedades são diferentes.

Assim, de maneira a apresentar funcionalidades relativas às preferências, suponha-se o caso em que um cliente realiza um pedido (Fig.6.15) de um telemóvel cuja escolha do componente "ecrã" é o LCD e que tem como preferência o tempo de entrega.

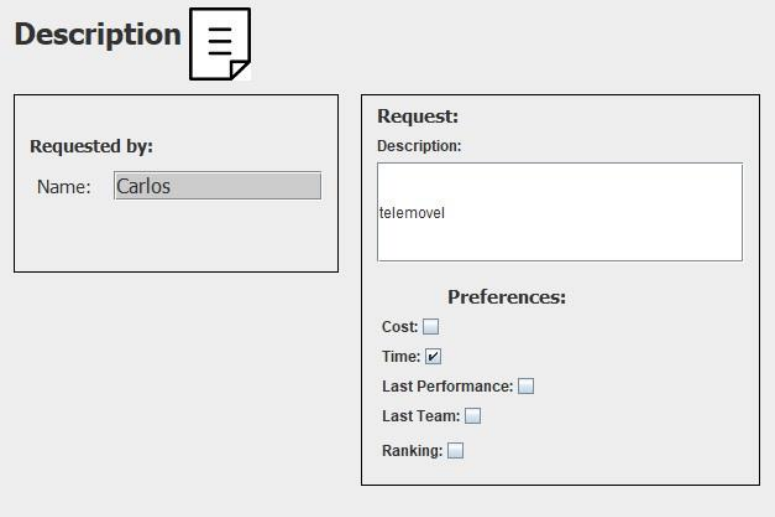


Figura 6.15-Segunda Simulação – Descrição do pedido.

Para este caso, a escolha do *Customer Agent* relativamente às propostas para as tarefas realizadas pelos *Supplier Agents,* vai ser baseada na preferência do cliente.

Na Figura 6.16 são apresentadas as interações entre os *Supplier Agents* que possuem a *Resource* LCD e o *Customer Agent (*ID =45*) responsável* pelo pedido do cliente "Carlos".

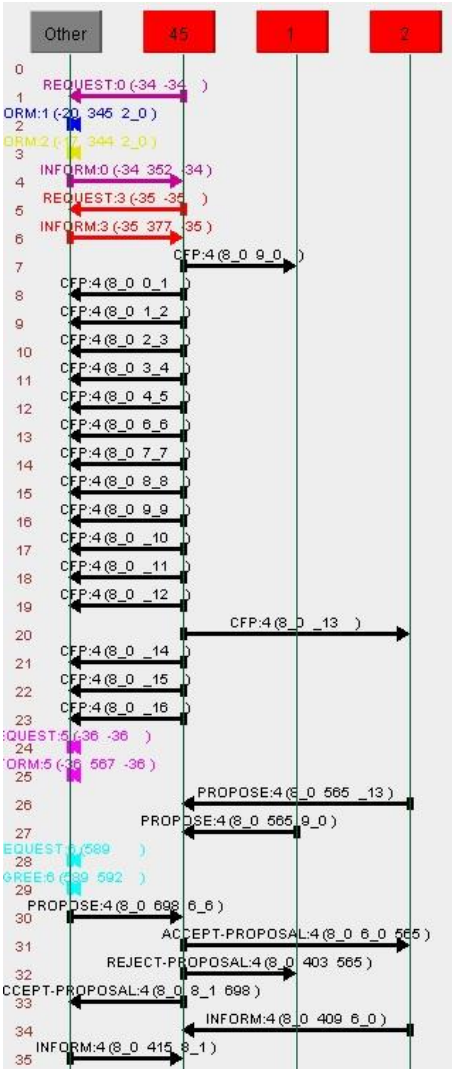


Figura 6.16-Segunda Simulação – Propostas à tarefa que necessita da Resource LCD.

As mensagens 26 e 27 dizem respeito às propostas realizadas pelos agentes 2 e 1 que, respetivamente, abstraem as instâncias relativas às máquinas/equipamentos *LCDEquipment2 e LCDEquipment1.*

Na interação 32 o *Customer Agent* rejeita a proposta do agente 1 e na interação 31 aceita a proposta do agente 2. Isto advém devido ao agente 2 corresponder de melhor forma à preferência do pedido, que neste caso é o tempo. Ou seja, o agente 2 oferece, como parâmetro da sua proposta, um melhor (menor) tempo de execução que o agente 1, o que o torna responsável pela tarefa.

Para o mesmo pedido, foi requisitado para a componente "Bateria" a variável "4000mH". Esta corresponde à *Resource* que os *Suppliers "BatteryEquipment2"* e *"BatteryEquipment3"* têm em comum.

De forma a apresentar as funcionalidades relativas ao protocolo *FIPA Request* implementado para possibilitar a solicitação de *stock,* nesta simulação foram realizadas as seguintes afetações:

- Os *Suppliers "BatteryEquipment2"* e *"BatteryEquipment3"* são "Buddys".
- O "*status" do Supplier "BatteryEquipment2"* é igual a 1, o que o impossibilita de fazer uma proposta para qualquer tarefa.
- O *stock do "BatteryEquipment3"* é igual a 0.

Na Figura 6.17 pode-se observar as interações ocorridas para este caso.

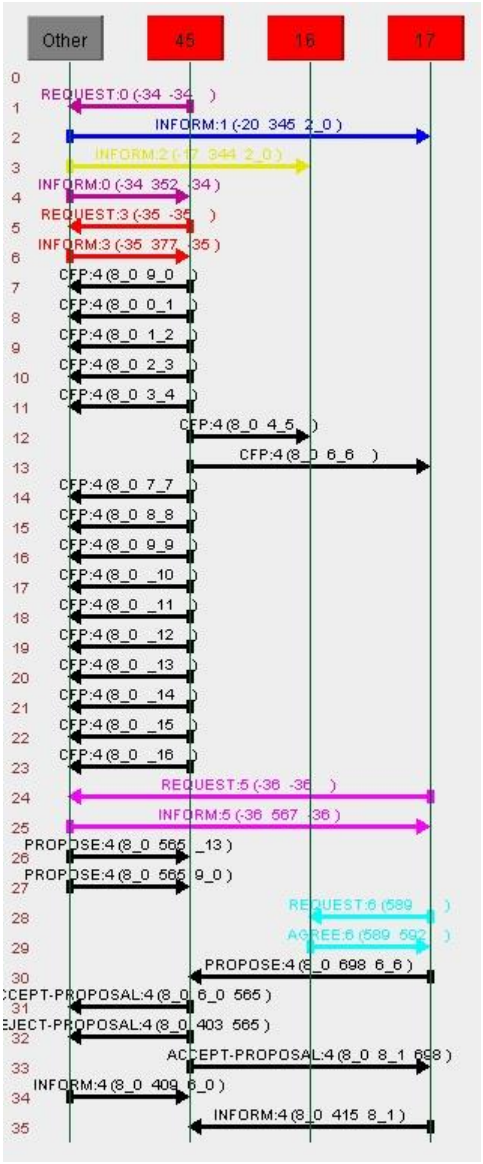


Figura 6.17-Segunda Simulação – Solicitação de stock e consequente proposta.

A *lifeline* representada pelo número 16 é relativa ao agente que abstrai o *"BatteryEquipment2"* e a *lifeline* representada pelo número 17 é relativa ao agente que abstrai *"BatteryEquipment3"*.

As interações representadas pela côr azul-claro são relativas ao protocolo de solicitação de *stock.* Na interação 28 o agente 17 envia uma mensagem ao agente 16, com a performativa *Request,* a fim de solicitar *stock,* a qual é respondida, na interação 29, com uma mensagem com a performativa *Agree,* que representa uma aceitação ao pedido do agente 17.

Uma vez que o agente 17 possui o *stock* necessário para realizar a proposta à tarefa pretendida, na interação 30 é enviada a mensagem ao *Customer Agent* com a proposta à tarefa, a qual é respondida com uma mensagem de aceitação na interação 33, o que faz com que o agente 17 incorpore a equipa responsável pelo pedido em questão.

6.3 Considerações finais

Nas subsecções anteriores foram ilustradas as funcionalidades e interfaces gráficas implementadas na realização deste protótipo, efetuando duas simulações para o cenário descrito no início do capítulo.

Inúmeros cenários poderiam ter sido utilizados para validar este protótipo, porém apenas cenários no contexto de produção discreta. Este tipo de produção permite a padronização e particularização entre produtos. Estes foram os principais conceitos a influenciar o desenvolvimento deste protótipo.

Um outro cenário de aplicação que poderia ter sito utilizado seria, por exemplo, num ambiente de produção automóvel, no qual o produto final também seria composto por múltiplos componentes individuais.

O protótipo demonstrado neste capítulo visou, principalmente, apresentar uma proposta à colaboração entre dispositivos presentes no mesmo ambiente de produção, a fim de aumentar a sua agilidade e eficiência. É, contudo, de salientar que a proposta desenvolvida apenas possibilita a simulação de ambientes virtuais, sendo que, para a simulação de um cenário real teria que haver um aprimoramento no sistema.

Capítulo 7

7. Conclusões e Futuros Desenvolvimentos

Neste capítulo são apresentadas as conclusões do trabalho desenvolvido e ideias para futuros desenvolvimentos.

7.1 Sumário de Resultados

O protótipo desenvolvido nesta dissertação sugeriu a possibilidade de uma arquitetura multiagente, baseada em conceitos relativos às redes colaborativas, ser incorporada num ambiente de produção discreta, tendo como objetivo a sua otimização. Esta solução foi motivada pelos conceitos "*Smart Factory*" e "*Smart Connectivity*".

Um dos principais objetivos para este sistema foi a implementação do conceito de identificação de oportunidade de colaboração por parte das entidades "*Suppliers*". Este considera-se cumprido dado que um *Supplier* não se limita a executar ordens, tendo a possibilidade de analisar os pedidos presentes no sistema. Contudo, este objetivo foi sempre equilibrado com o objetivo de ter um ambiente que correspondesse sempre aos pedidos de um cliente.

Os resultados das simulações realizadas para a validação deste protótipo foram positivas, mostrando ser possível uma colaboração multiagente num ambiente de produção análogo ao apresentado no capítulo anterior. Foi exposto um novo contexto de comunicação e planeamento entre máquinas, equipamentos ou dispositivos que mostra a possibilidade de acrescentar inteligência artificial a cada um deles, através do uso do sistema multiagente.

Finalizando, o protótipo proposto contribui para o estudo relacionado com o planeamento colaborativo em ambientes de manufatura, que atualmente é um requisito para o funcionamento eficiente de uma fábrica, sendo que os seus ambientes são cada vez mais complexos.

7.2 Futuros Desenvolvimentos

Devido ao facto do trabalho desenvolvido ser um protótipo implementado de "raiz", naturalmente este necessitará de otimizações de maneira a poder corresponder a novos requisitos.

Como trabalho futuro, são sugeridos os seguintes pontos que contribuem para um aprimoramento do protótipo:

- Otimização do código implementado;
- Criação de um algoritmo para uma automação da decomposição de tarefas;
- Considerar a criação de uma base de dados tendo como base a ontologia criada, procurando um aprimoramento na gestão dos dados relativos às entidades presentes no sistema;
- Criação de um método de deteção e diagnóstico de erros do sistema com a respetiva interface gráfica para um melhoramento da sua monitorização; e
- Aplicação do protótipo a um cenário real, para que fosse possível comparar os tempos de resposta entre o cenário sem o sistema e o cenário com o sistema. Para tal, seria necessário realizar uma adaptação do código.

Referências

Beer, M., D'inverno, M., Luck, M., Jennings, N., Preist, C., & Schroeder, M. (1999). Negotiation in multi-agent systems. The Knowledge Engineering Review, 14(3), 285-289.

Bititci, U. S., Martinez, V., Albores, P., & Parung, J. (2004). Creating and managing value in collaborative networks. International Journal of Physical Distribution & Logistics Management, 34(3/4), 251-268.

Brandão, C., Reis, L. P., & Rocha, A. P. (2013, June). Evaluation of Embodied Conversational Agents. In 2013 8th Iberian Conference on Information Systems and Technologies (CISTI)(pp. 1-6). IEEE.

Bravo, M., Reyes-Ortiz, J. A., Rodríguez, J., & Silva-López, B. (2015, December). Multi-agent Communication Heterogeneity. In 2015 International Conference on Computational Science and Computational Intelligence (CSCI) (pp. 583-588). IEEE.

Bulling, N. (2014). A survey of multi-agent decision making. KI-Künstliche Intelligenz, 28(3), 147-158. Bititci, U. S., Martinez, V., Albores, P., & Parung, J. (2004). Creating and managing value in collaborative networks. *International Journal of Physical Distribution & Logistics Management*, *34*(3/4), 251-268.

Calado, Gualber. TIPOLOGIAS DA PRODUÇÃO. 28 July 2015, gualber.wordpress.com/2015/07/27/tipologias-da-produo/.

Camarinha-Matos, L. M. (Ed.). (2013). Collaborative Business Ecosystems and Virtual Enterprises: IFIP TC5/WG5. 5 Third Working Conference on Infrastructures for Virtual Enterprises (PRO-VE'02) May 1–3, 2002, Sesimbra, Portugal (Vol. 85). Springer.

Camarinha-Matos, L. M., & Afsarmanesh, H. (2005). Collaborative networks: a new scientific discipline. Journal of intelligent manufacturing, 16(4-5), 439-452.

Camarinha-Matos, L. M., & Afsarmanesh, H. (2018). Roots of Collaboration: Nature-inspired solutions for Collaborative Networks. IEEE Access, 6, 30829-30843.

Camarinha-Matos, L. M., Afsarmanesh, H., & Ollus, M. (2005, September). ECOLEAD: A holistic approach to creation and management of dynamic virtual organizations. In Working Conference on Virtual Enterprises (pp. 3-16). Springer, Boston, MA.

Camarinha-Matos, L. M., Oliveira, A. I., Demsar, D., Sesana, M., Molina, A., Baldo, F., & Jarimo, T. (2008). VO Creation Assistance Services. In L. M. Camarinha-Matos, H. Afsarmanesh & M. Ollus (Eds.), Methods and Tools for Collaborative Networked Organizations (pp. 155-190): Springer.

Camarinha-Matos, L. M., Oliveira, A. I., Ratti, R., Baldo, F., & Jarimo, T. (2007). A Computer-Assisted VO Creation Framework. In L. M. Camarinha-Matos, H. Afsarmanesh, P. Novais & C. Analide (Eds.), Establishing the Foundation of Collaborative Networks (Vol. 243, pp. 165-178). Boston: Springer.

Camarinha-Matos, L., & Afsarmanesh, H. (2008). Collaboration forms. In L. M. Camarinha & H. Afsarmanesh (Eds.), Collaborative networks: reference modeling (pp. 51-66): Springer.

Camarinha-matos, L. M., & DEE Group Author (2011). Adaptation and Value Creating Collaborative Networks. (IFIP Advances in Information and Communication Technology). Springer Heidelberg Dordrecht London New York: Springer.

Chen, X. L., Hesse, C., Riedel, R., & Mueller, E. (2016, December). The choice of a collaboration form-a special insight in the case of R&D consortia. In 2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM) (pp. 1059-1063). IEEE.

Fu, J., Tanner, H. G., & Heinz, J. (2015). Concurrent multi-agent systems with temporal logic objectives: game theoretic analysis and planning through negotiation. IET Control Theory & Applications, 9(3), 465-474.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future generation computer systems, 29(7), 1645-1660.

Islam, M. J., Ferrer, B. R., Xu, X., Nieto, A., & Lastra, J. L. M. (2016, July). Implementation of an industrial visualization model for collaborative networks. In 2016 IEEE 14th International Conference on Industrial Informatics (INDIN) (pp. 720-725). IEEE.

Jiang, Y., Weng, W., & Fujimura, S. (2016, July). Multi-agent Just-in-Time Manufacturing Scheduling System for Dynamic Environment. In 2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI) (pp. 1022-1025). IEEE.

Kallenberg, L. (2011). Markov decision processes. Lecture Notes. University of Leiden.

Kannengiesser, U., & Müller, H. (2013, November). Towards agent-based smart factories: A subject-oriented modeling approach. In Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)-Volume 03 (pp. 83-86). IEEE Computer Society.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In Machine learning proceedings 1994 (pp. 157-163). Morgan Kaufmann.

McArthur, S. D., & Davidson, E. M. (2005, November). Concepts and approaches in multi-agent systems for power applications. In Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems (pp. 5-pp). IEEE.

Nieto, M. J., & Santamaría, L. (2007). The importance of diverse collaborative networks for the novelty of product innovation. Technovation, 27(6-7), 367-377.

Oliveira, A. I. and L. M. Camarinha-Matos (2012). Electronic Negotiation Support Environment in Collaborative Networks. Technological Innovation for Value Creation. L. M. Camarinha-Matos, E. Shahamatni and G. Nunes, Springer: 21-32.

Oliveira, A. I., & Camarinha-Matos, L. M. (2008). Agreement Negotiation Wizard. In L. M. Camarinha-Matos, H. Afsarmanesh & M. Ollus (Eds.), Methods and Tools for Collaborative Networked Organizations (pp. 191-218): Springer.

Oliveira, A. I., & Camarinha-Matos, L. M. (2010). Negotiation and Contracting in Collaborative Networks. In L. M. Camarinha-Matos, P. Pereira & L. Ribeiro (Eds.), Emerging Trends in Technological Innovation (pp. 83-92): Springer.

Oliveira, A. I., & Camarinha-Matos, L. M. (2015). Negotiation Environment and Protocols for Collaborative Service Design. In L. M. Camarinha-Matos, T. A. Baldissera, G. Di Orio & F. Marques (Eds.), Technological Innovation for Cloud-Based Engineering Systems (Vol. 450, pp. 31-41): Springer.

Oliveira, A. I., Camarinha-Matos, L. M., & Pouly, M. (2010). Agreement negotiation support in virtual organisation creation–an illustrative case. Production Planning & Control, 21(2), 160-180.

Panait, L., & Luke, S. (2005). Cooperative multi-agent learning: The state of the art. Autonomous agents and multi-agent systems, 11(3), 387-434.

Pessoa, M., & Spinola, M. (2014). Introdução à automação para cursos de engenharia e gestão. Elsevier Brasil.

Poslad, S. (2007). Specifying protocols for multi-agent systems interaction. ACM Transactions on Autonomous and Adaptive Systems (TAAS), 2(4), 15.

Pynadath, D. V., & Tambe, M. (2002). The communicative multiagent team decision problem: Analyzing teamwork theories and models. Journal of artificial intelligence research, 16, 389-423.

Reis, L. P. (2003). Coordenação em Sistemas Multi-Agente(Doctoral dissertation, PhD thesis, Faculdade de Engenharia da Universidade do Porto).

Rizk, Y., Awad, M., & Tunstel, E. W. (2018). Decision making in multiagent systems: A survey. IEEE Transactions on Cognitive and Developmental Systems, 10(3), 514-529.

Rüßmann, M., Lorenz, M., Gerbert, P., Waldner, M., Justus, J., Engel, P., & Harnisch, M. (2015). Industry 4.0: The future of productivity and growth in manufacturing industries. Boston Consulting Group, 9(1), 54-89.

Silva, L. D. M. (2003). Estudo e desenvolvimento de sistemas multiagentes usando jade: Java agent development framework. Trabalho de Conclusão de Curso (Graduação em Ciências da Computação)–Centro Tecnológico. Universidade Federal de Pernambuco, Pernambuco, 15.

Tošić, P. T., & Ordonez, C. (2012, December). Distributed protocols for multi-agent coalition formation: a negotiation perspective. In International Conference on Active Media Technology (pp. 93-102). Springer, Berlin, Heidelberg.

Wang, K., Kovacs, G. L., Wozny, M., & Fang, M. (Eds.). (2006). Knowledge Enterprise:

Intelligent Strategies in Product Design, Manufacturing, and Management: Proceedings of PROLAMAT 2006, IFIP TC5, International Conference, June 15-17 2006, Shanghai, China (Vol. 207). Springer Science & Business Media.

Xie, J., & Liu, C. C. (2017). Multi-agent systems and their applications. Journal of International Council on Electrical Engineering, 7(1), 188-197.

Xiong, C., Xiang, W., & Ouyang, Y. (2012, July). Argumentation in multi-agent system based on jade. In 2012 Third International Conference on Intelligent Control and Information Processing(pp. 88-91). IEEE.

Yu, Y., Li, Y., & Deng, Y. (2010, September). Research and Realization of the Mobile Agent Communication in KQML Environment. In 2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)(pp. 1-4).