Pedro Grilo Barroca

Licenciado em Ciências da Engenharia Eletrotécnica e de Computadores

Ambiente de simulação para sistemas de manufatura baseados em agentes

Dissertação para obtenção do Grau de Mestre em

Engenharia Eletrotécnica e de Computadores

Orientador: José António Barata de Oliveira, Professor Doutor, FCT-UNL

Coorientadores: André Dionísio Bettencourt da Silva Parreira Rocha, Mestre, CTS Uninova

Júri:

Presidente: Prof. Doutor Tiago Oliveira Cardoso

Arguentes: Prof. Doutor João Paulo Pimentão

Vogais: Prof. Doutor José António Barata de Oliveira

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro de 16

Ambiente de simulação para sistemas de manufatura baseados em agentes

“Challenges are what make life interesting;

overcoming them is what makes life meaningful”

-Joshua J. Marine

***Dedicada à família e amigos**,*

Pela amizade, pela paciência, pelo incentivo,

por não me deixarem desistir nas piores alturas,

por serem exigentes, honestos, apoiantes incondicionais

e principalmente pela confiança nas minhas capacidades.

Em muito vos devo, e vos dedico este trabalho.

Agradecimentos

Nesta secção quero expressar os meus mais sinceros agradecimentos a todos os que, de forma mais ou menos direta, contribuíram para a conclusão deste trabalho que é o culminar de um percurso académico cheio de alegrias, sofrimentos, amizades e sacrifícios.

Em primeiro lugar quero agradecer à minha família pela paciência e motivação, em especial aos meus pais pela oportunidade de poder atingir mais um objetivo e poder disfrutar de toda esta educação que me proporcionaram. Pela personalidade e caráter que me permitiram desenvolver e pelos valores com que me formaram.

Ao meu orientador, Professor Doutor José Barata, por me ter dado a oportunidade de desenvolver uma dissertação na área da robótica e manufatura e por conseguir sempre mostrar as capacidades e o potencial desta temática através da forma motivadora e cativante como expõe os assuntos, relacionando sempre a matéria com aplicações reais e inovadoras.

Ao meu coorientador, estudante de doutoramento André Rocha, um sincero agradecimento pela paciência, motivação, tempo despendido e orientação prestada, que sem dúvida foi essencial para o desenvolvimento desta dissertação.

É importante não esquecer a empresa TTS e o seu porta voz, neste processo, o Giovanni Dal Maso, que contribuiu de forma incansável e disponibilizou todo o apoio técnico que fosse necessário. Por isso lhes quero endereçar um grande agradecimento.

Quero prestar também um forte agradecimento a todos os colegas que fizeram parte do meu percurso académico, pela amizade, simpatia e companheirismo. Foi com muito prazer que partilhei esta experiência convosco e tenho certeza de que muitas das amizades irão tornar-se permanentes e de longa duração.

Um especial agradecimento ao Professor Doutor Pedro Maló pela amizade, motivação e pelas oportunidades que me tem proporcionado, revelando-se uma das peças chave para a minha formação, não só académica, mas também como pessoa.

Por último, e um dos mais importantes, um agradecimento muito especial à Beatriz João pela paciência, carinho e compreensão demonstradas, que fazem dela uma das pessoas que mais prezo e que mais desejo manter por perto.

A todos um sincero obrigado, com a consciência de que sem todo o vosso apoio este processo teria sido impossível.

Resumo

Hoje em dia existe uma forte tendência na manufatura para que os sistemas industriais ganhem novas características. Com a globalização dos mercados surgiram novos paradigmas que se baseiam na personalização de produtos em alternativa à implementada produção em massa. A flexibilidade e adaptabilidade a adversidades são novos fatores críticos e que podem diferenciar um falhanço empresarial de uma solução sustentável e bem-sucedida no mercado.

Para além dos novos paradigmas, uma forte pressão, causada pela competição, na redução de custos, leva à utilização de novas alternativas no teste de arquiteturas inovadoras. O método tradicional de parar a produção para reconfigurar, não é, nos dias de hoje, posto em prática com facilidade, pois a cada segundo dinheiro é perdido. Para evitar estes tempos mortos recorre-se à simulação. Esta utilização tem outras vantagens, entre elas a avaliação de indicadores de desempenho em tempo reduzido, a utilização para formação de trabalho especializado e a redução de acidentes em períodos experimentais.

A presente dissertação visa propor uma arquitetura de integração entre um simulador comercial e uma arquitetura multiagente, capaz de satisfazer os requisitos da manufatura atuais, sem perder a qualidade dos simuladores modernos.

Palavras-chave: Sistemas Multiagente, Simulação, Sistemas distribuídos, Indústria 4.0, Conectar e produzir, Interoperabilidade

Abstract

Nowadays, the manufacturing systems tend to embrace new features. The globalization of markets introduced new paradigms that are based on products customization instead of mass production, currently implemented. Flexibility and adaptability to adversities are, from now on, critical factors that can distinguish between business failure and a sustainable and successful market solution.

In addition to the new paradigms, strong pressure from competition to reduce costs, leads to the use of alternative ways to test new architectures. The traditional method of stopping the entire production line to reconfigure it, is no longer practicable because every second money is lost. In order to avoid these production downtimes, simulators are used. Those solutions have some other advantages, including the faster achievement of performance indicators, the usage as training tool for specialized workers and the reduction of accidents rate in experimental periods.

This thesis proposes an integration between a commercial simulator and a multi-agent architecture, in order to meet the current manufacturing requirements without losing the quality of modern simulators.

Keywords: Multi-agent Systems, Simulation, Distributed Systems, Industry 4.0, Plug and Produce, Interoperability

Acrónimos

ADACOR	***ADAptative holonic COntrol aRchitecture for distributed manufacturing systems***
BMS	**Sistemas Biónicos de Manufatura**, do inglês *Bionic Manufacturing Systems*
CAD	**Desenho Assistido por Computador**, do inglês *Computer Aided Design*
EAS	**Sistemas Evolutivos de Manufatura**, do inglês *Evolvable Assembly Systems*
EPS	**Sistemas Evolutivos de Produção**, do inglês *Evolvable Production Systems*
FIPA	***Foundation for Intelligent Physical Agents***
FIPA-ACL	**Linguagem de Comunicação de Agentes FIPA**, do inglês *FIPA Agent Communication Language*
FMS	**Sistemas Flexíveis de Manufatura**, do inglês *Flexible Manufacturing Systems*
HMS	**Sistemas Holónicos de Manufatura**, do inglês *Holonic Manufacturing Systems*
IMS	**Sistemas Inteligentes de Manufatura**, do inglês *Intelligent Manufacturing Systems*
JADE	***Java Agent Development Framework***
MAS	**Sistemas Multiagente**, do *inglês Multi-Agent Systems*
PRIME	***Plug and Produce Intelligent MultiAgent Environment***
PROSA	***Product-Resource-Order-Staff Architecture***
RMS	**Sistemas Reconfiguráveis de Manufatura**, do inglês *Reconfigurable Manufacturing Systems*
TTS	***Technology Transfer System S.r.l.,*** (Technology Transfer System S.r.l., 2016)
XML	***eXtensible Markup Language***

Índice de Conteúdos

Índice de Figuras

Índice de Tabelas

1

1.Introdução

1.1. Apresentação do problema

Nos primórdios da produção industrial, a manufatura era caraterizada pelos escassos modelos e pela pouca diversidade de produtos, assim sendo o paradigma que vingou foi, na altura, a produção em massa. Este paradigma baseia-se numa produção em larga escala, recorrendo a uma linha de montagem composta por várias estações, onde cada uma realiza uma pequena tarefa, o que permite criar uma grande quantidade de produtos num curto espaço de tempo.

Com o evoluir dos tempos, diversos aspetos sociais, económicos e tecnológicos provocaram mudanças nos mercados. Muitas empresas de manufatura sentiram a necessidade de reformular os seus conceitos de produção. As exigências dos consumidores foram mudando, e os paradigmas mudaram de uma perspetiva padrão para uma perspetiva de personalização, surgindo a customização/personalização em massa.

Por um lado, a personalização em massa exige uma continuidade no ritmo de produção, por outro, é preciso garantir a possibilidade de alterar as especificações o mais ágil e rapidamente possível, sendo imprescindível para qualquer empresa que deseja atingir a sustentabilidade, garantir a adaptabilidade das suas linhas de produção às exigências do mercado e aos desafios impostos pela globalização dos mesmos.

Deste modo os métodos tradicionais de manufatura tornaram-se obsoletos e inadequados às realidades existentes, pois numa sociedade cada vez mais exigente e competitiva no que diz respeito à qualidade do produto, à redução dos custos de produção e às renovações de tendências cada vez mais frequentes, características como a flexibilidade e a agilidade são imprescindíveis em qualquer ramo da manufatura. É neste sentido que a manufatura tem evoluído, consoante as novas necessidades, novos paradigmas surgem, e novas soluções são propostas com vista a cobrir todas as adversidades que possam surgir (André Dionísio Rocha, Barata, Di Orio, Santos, & Barata, 2015).

As abordagens propostas têm passado por sistemas com capacidade de reajuste, sistemas flexíveis e até sistemas evolutivos. A aplicação da informática na manufatura tem trazido

grandes vantagens, uma delas é a implementação de sistemas mais complexos e com mais capacidade. Face à crescente complexidade dos sistemas é imperativo que surjam novas soluções baseadas em sistemas distribuídos. Cada vez mais a escalabilidade de uma solução é um dos principais critérios a par da capacidade de adaptação a falhas e recuperação das mesmas.

Por outro lado, com a aplicação de métodos avançados no controlo da produção é necessária uma mão de obra mais especializada e com aprendizagem focada no controlo de equipamento industrial autónomo e cooperativo. Assim sendo é necessário adotar abordagens que permitam a minimização dos riscos e acidentes. Uma das opções prende-se com a simulação dos ambientes de produção. A simulação permite não só criar os mais diversos modelos, em qualquer área, como simular o seu comportamento e prever quaisquer falhas inerentes à má utilização ou programação dos sistemas.

No caso das linhas de manufatura, pretende-se que estas estejam em constante funcionamento, pelo que é necessário recorrer a simuladores para testar novas ideias ou novas metodologias a implementar no futuro. Recorrendo a ambientes de simulação é possível criar também uma ferramenta de suporte à especialização da mão de obra, pois como todo o treino é feito virtualmente, são eliminados todos os riscos de potenciais acidentes de aprendizagem.

A presente dissertação propõe, apresenta e valida uma implementação baseada nos mais recentes paradigmas de manufatura e num simulador presente no mercado, de modo a integrar estas duas tenologias separadamente desenvolvidas. A arquitetura proposta visa a utilização de sistemas existentes, com determinadas características, nomeadamente o comportamento modular, independentemente das tecnologias utilizadas.

1.2. Perguntas de investigação

Tendo por base os temas acima sugeridos, com a evolução contínua da manufatura e a necessidade do avanço tecnológico surgem algumas questões pertinentes, as quais esta tese pretende abordar.

- Será possível integrar um simulador industrial de excelência, presente no mercado e desenvolvido segundo as linhas de produção em massa, com os novos paradigmas da manufatura?

- Que tipo de características são necessárias em ambos os sistemas para que a sua integração seja independente da tecnologia utilizada na implementação de cada um?

- Utilizando uma abordagem multiagente, será possível sugerir algumas interfaces de integração entre os sistemas de modo a poder generalizar a implementação a várias arquiteturas?

É cada vez mais importante a interoperabilidade de modo a melhorar, em vez de substituir totalmente, os sistemas. Criar novas soluções com base em trabalhos desenvolvidos provoca um menor consumo de tempo e uma melhor especialização nas componentes, visto serem desenvolvidas cada uma pelos seus especialistas.

Este documento descreve detalhadamente a integração entre um simulador industrial que tem por base o funcionamento modular, e uma arquitetura multiagente capaz de implementar, genericamente, os mais atuais paradigmas da manufatura.

1.3. Visão Geral do Trabalho Realizado

Com o intuito de contribuir para a resposta às questões acima formuladas, este documento sugere uma arquitetura, e sua implementação, que pretende integrar um sistema baseado em conceitos de Sistemas Multiagente, **MAS** do inglês *Multi-Agent Systems*, e um simulador presente no mercado, sem que sejam feitas quaisquer alterações no funcionamento do mesmo, provando que a interoperabilidade entre estes dois tipos de sistema é possível.

Para implementação desta arquitetura foi utilizado o ambiente de programação NetBeans e consequentemente a linguagem Java. Esta escolha deveu-se à linguagem base do simulador utilizado, o DDDSimulator, e à qualidade inerente à plataforma multiagente **JADE**.

A arquitetura proposta baseia-se nos mais recentes paradigmas da manufatura. A divisão do sistema em agentes permite garantir ao sistema grande agilidade e dinâmica na produção e transporte dos produtos. Com o intuito de sugerir uma arquitetura genérica, foram criados vários grupos de agentes, os de transporte, os de execução, os de abstração e os de controlo. Os agentes de controlo servem para garantir a integridade do sistema, permitindo a pesquisa de agentes na plataforma e o lançamento dos mesmos.

Os agentes de transporte pretendem abstrair as principais componentes numa linha de montagem, entre elas, passadeiras e pontos de encaminhamento. Estas duas entidades têm como funcionalidade controlar o bom funcionamento dos módulos, monitorizando todas as etapas e todos os estágios de funcionamento dos módulos. Outro agente, não tão fácil de compreender, é o transport agent. Este é responsável pela organização e consistência do transporte em toda a linha, permitindo que o produto apenas tenha de conhecer o seu destino e que o seu deslocamento esteja à distância de um simples pedido de transporte.

Outro grupo, os agentes de execução, são responsáveis pelas ações sobre o produto, começando pela entrada e a saída da linha, responsáveis por criar e destruir o produto, até aos agentes de recurso, ou estações, onde são disponibilizadas e executadas as tarefas características à produção industrial.

Por último o agente de abstração pretende controlar a produção do produto, sendo responsável pela requisição do transporte, pela execução das tarefas e pelo controlo na sua ordem.

Todos os agentes carregam informações através de ficheiros existentes na diretoria do projeto, assim sendo alterar as especificações de cada módulo é fácil e pode ser feito em tempo

real. Por exemplo no caso do produto, é possível alterar o ficheiro e o próximo produto desse tipo que for lançado no sistema já terá em conta as novas especificações.

Para a validação da arquitetura proposta foi implementado, no simulador, uma célula de produção, a NOVAFLEX, que para além de estar disponível num laboratório da faculdade é uma célula que apresenta alguma complexidade e permite testar vários tipos de interceções e cenários presentes em sistemas de produção industrial.

O primeiro passo na construção do ambiente de simulação é a criação dos modelos tridimensionais que representam as peças presentes na linha original. Para tal foi utilizado o programa, de **CAD**, SolidWorks, escolhido pelo seu grande leque de opções e pelo formato de exportação requisitado pelo simulador ser suportado. Foi tido em conta o aspeto visual do ambiente de simulação, ou seja, os modelos criados tentam ilustrar de forma fidedigna o aspeto real dos módulos e para isso foram criados vários componentes de cada módulo que posteriormente careceram de uma montagem, como explicado mais aprofundadamente nos capítulos seguintes.

Após a construção das peças que integram o ambiente de simulação foi necessário dispô-las no simulador e criar todos os referenciais necessários ao bom funcionamento do mesmo. Estas especificações relacionam-se com o modo de funcionamento do simulador, que visto ser baseado em pontos de referência, caracterizados por três coordenadas e três orientações, carece duma fase de especificação onde estes são definidos.

Para garantir a integração entre as duas componentes foram pensados vários protocolos de comunicação, dividindo o transporte entre dois elementos do sistema em quatro fases distintas. A primeira fase serve para perceber qual o próximo módulo para onde vai o produto, utilizando tabelas de encaminhamento, todos os módulos sabem qual o próximo módulo para chegar a um destino. A segunda fase tem como propósito a alocação de uma posição para o novo produto ocupar. A terceira fase serve para libertar a posição que era ocupada no módulo de origem. Por fim a quarta fase retrata o transporte em si, onde existe o movimento da peça entre os dois referenciais, a origem e o destino

Por último a criação de uma classe de lançamento, permite, em tempo real, adicionar produtos à fila de espera do módulo de entrada. Assim sendo é possível lançar no sistema novos produtos quando se desejar.

1.4. Principais Contribuições

As principais contribuições propostas pelo trabalho realizado relacionam-se com uma nova forma de integrar tecnologias já existentes. Utilizando dois componentes completamente distintos, foi possível cruzar conhecimentos e criar uma arquitetura inovadora que pretende alcançar a agilidade e capacidade de adaptação às adversidades, que tanto se pretende hoje em dia na manufatura, sem pôr em causa a importante componente estética, fundamental nos simuladores modernos.

A arquitetura desenvolvida tem por base o comportamento modular dos atuais sistemas de produção. Com isto pretende-se que todas as tecnologias sejam compatíveis com esta abordagem.

O sistema multiagente, tenta ser o mais genérico possível para garantir que a sua substituição, por qualquer arquitetura de transporte ou de produção, baseada em agentes, seja fácil e intuitiva.

A simples interação entre o sistema multiagente e o simulador pretende facilitar a implementação aquando da transferência para o mundo real. Tentou-se que estas interações simulassem o atual sistema de comunicação entre máquinas, com a habitual rede de sensores e atuadores a trocar informações e a despertar eventos.

Em suma, com este trabalho pretende-se criar um ponto de partida para a integração de sistemas complexos de agentes com simuladores de grande qualidade presentes no mercado.

1.5. Descrição do documento

Este documento é composto por sete capítulos, sendo eles a introdução, o estado da arte, a arquitetura, a implementação, a validação, a conclusão e trabalhos futuros e as referências utilizadas.

Este capítulo, a **Introdução** pretende contextualizar o projeto desenvolvido através da caracterização do problema que pretende abordar e dum simples resumo do trabalho realizado. Apresentando por fim as contribuições que surgem para trabalhos futuros e para a investigação cientifica na área da simulação e da manufatura baseada em agentes.

O segundo capítulo, o **Estado da Arte**, é onde são tidas em conta bases teóricas para o desenvolvimento do projeto. É neste capítulo que se encontra grande parte do trabalho referenciado por este documento, onde é possível encontrar a evolução dos paradigmas da manufatura, alguns simuladores industriais presentes no mercado e por fim alguns programas úteis ao trabalho desenvolvido.

No terceiro capítulo, a **Arquitetura**, é descrita a arquitetura do trabalho que se baseia numa solução multiagente e no funcionamento da maior parte dos simuladores industriais, com o seu comportamento modular.

O quarto capítulo, a **Implementação**, descreve detalhadamente o trabalho realizado e todas as etapas que culminaram com o desenvolvimento do trabalho aqui apresentado, resumidamente é a implementação da arquitetura descrita no capítulo anterior.

No capítulo cinco, o qual é designado por **Validação**, ilustra os resultados obtidos, aplicando o sistema desenvolvido a uma linha de produção real. Neste caso simulou-se a linha NOVAFLEX, uma linha de investigação com alguma complexidade e que permite estudar a aplicabilidade da arquitetura proposta.

Em penúltimo lugar, o capítulo **Conclusões e Trabalho Futuro**, onde são enumeradas as conclusões obtidas através da realização do trabalho e são sugeridas novas diretorias de investigação relacionadas com o trabalho desenvolvido.

Por fim o capítulo das **Referências**, composto pelas referências aos documentos de literatura que suportam o conhecimento contido neste projeto.

2

2.Estado da Arte

Neste capítulo o objetivo é introduzir a simulação e os sistemas de controlo inteligentes visto serem duas peças fundamentais de suporte à arquitetura proposta, assim sendo segue-se uma descrição do trabalho realizado nestas temáticas, de modo a enquadrar o desenvolvimento desta arquitetura na literatura, até à data, elaborada.

Existem muitas abordagens no que diz respeito a processos de fabrico e paradigmas de manufatura. No século XIX, onde o paradigma era a manufatura propriamente dita, o trabalho era totalmente feito à mão, pelo que todos os produtos eram únicos e, ao seu jeito, imperfeitos devido às variáveis que os operários incutem aos produtos em que trabalham (Oliveira, 2003).

Com o passar do tempo e a evolução dos mercados, novas necessidades surgiram, e novos paradigmas apareceram (Ribeiro & Barata, 2011). É o caso da produção em massa, celebrizada por Henry Ford no inicio do seculo XX, provando que a utilização de linhas de montagem no fabrico de produtos, trazendo a sistematização de processos e a distribuição de tarefas específicas, permite a produção de quantidades significativas de produtos, (Di Orio, 2013). Como o leque de especificações era pouco abrangente, um sistema que funcionasse com poucas regras era o ideal, permitindo criar sistemas com uma grande fluidez de produção, mas com fraca flexibilidade e variedade de produtos, prova disso é a celebre frase protagonizada também por Henry Ford "*Any customer can have a car painted any colour that he wants so long as it's black*", onde refere que todos os clientes podem escolher a cor do seu automóvel, desde que seja preto.

Com a mundialização dos mercados, novas soluções foram exigidas, sendo imprescindível a personalização dos produtos em substituição da produção em massa. Assim sendo foram necessárias alterações, tanto nas linhas de produção como nas soluções nelas aplicadas. Para que a adaptação às necessidades dos clientes fosse mais dinâmica adotaram-se metodologias reconfiguráveis e mais ágeis, promovendo a possibilidade de adicionar e remover componentes consoante a procura o exigisse. Estas modificações na produção industrial trazem, cada vez mais, novos paradigmas, permitindo que hajam varias abordagens ao mesmo problema o que provoca uma melhor e mais rápida evolução.

2.1. Paradigmas de produção emergentes

Como já referido, com a evolução surgiram novos paradigmas de produção tais como os Sistemas Flexíveis de Manufatura do inglês *Flexible Manufacturing Systems*, **FMS**, e os Sistemas Reconfiguráveis de Manufatura, em literatura denominados por *Reconfigurable Manufacturing Systems*, **RMS**. Os mais recentes paradigmas surgem numa perspetiva da criação de sistemas adaptativos e reconfiguráveis utilizando estratégias de otimização e também a utilização de entidades inteligentes, autónomas e distribuídas. surgem então paradigmas como os Sistemas Holónicos de Manufatura, *Holonic Manufacturing Systems*, **HMS**, os Sistemas Biónicos de Manufatura, referenciados como *Bionic Manufacturing Systems*, **BMS**, e os Sistemas Evolutivos de produção do inglês *Evolvable Production Systems*, **EPS**. O ramo mais forte da investigação prende-se agora com os Sistemas Multiagente descritos em literatura por *Multi-Agent Systems*, **MAS**, sendo esta a tecnologia utilizada para implementação dos paradigmas descritos, resolvendo, através das suas características base, os mais recentes problemas apresentados em sistemas de manufatura.

Segue-se uma descrição aprofundada dos paradigmas, referenciando documentos científicos que suportam os conceitos apresentados.

2.1.1. Sistemas Flexíveis de manufatura, FMS

Continuando com a evolução temporal, surgem necessidades de diversidade de produção, o que implica limites no tempo necessário para adaptar as máquinas ou equipamentos à nova produção. Este paradigma prende-se com a necessidade de customização em massa e de uma melhor adaptação às alterações exigidas nos produtos, tecnologias e mercados. Estas constantes alterações lideraram uma procura por sistemas flexíveis, ou seja, capazes de produzir uma gama mais alargada de produtos sem ser necessária uma reconfiguração total do sistema, (ElMaraghy, 2005).

Os FMS caracterizam-se por possibilitar aos equipamentos de trabalho a capacidade de realizar mais de uma tarefa, e para que o mesmo equipamento possa realizar mais que uma operação de manufatura é necessário dotá-lo da capacidade de trocar a ferramenta ou de ter diferentes rotinas consoante o produto em que está a atuar, permitindo diferentes combinações e incrementando o número de operações possíveis. Como exemplo em (Cavalcante, Pereira, & Barata, 2010), onde um FMS se distingue por considerar a diversidade dos produtos a produzir, e as características adaptativas do sistema, mais propriamente das máquinas, garantindo a versatilidade dos produtos elaborados e assim prevendo oscilações nos modelos.

Contudo, com o aumento a variedade de soluções oferecidas, o volume de produção é em grande parte afetado, sendo esta solução ideal para combater nichos de mercado ao invés de grandes lotes. A fabricação de produtos com características similares, por exemplo propriedades geométricas, dimensões de tolerância semelhantes e componentes iguais permite uma fácil

adaptação de sistemas, tornando rentável a produção variada e garantindo uma redução no tempo de alterações na linha (Koren et al., 1999).

Outro exemplo é o modelo proposto em (Jahromi & Tavakkoli-Moghaddam, 2012) que pretende dinamizar a seleção do conjunto máquina-ferramenta e alocação dos processos a realizar. Como se trata de um FMS assume-se a existência de um determinado conjunto de ferramentas e máquinas que possibilitam a realização de diferentes tarefas, consoante as combinações entre si. O objetivo é minimizar os custos de produção, tais como os custos de instalação, manuseamento e troca de ferramentas entre máquinas.

Por último os FMS impõem alguns problemas desafiantes que se prendem com a alocação dos recursos aos diversos processos, à programação das atividades sequenciais necessárias para a produção, à centralização do controlo e à difícil escalabilidade, de maneira a tornar eficaz e melhorar o desempenho do sistema. Apesar de permitir a produção mais diversificada de produtos e acomodar mudanças no sistema, só é praticável se as variações forem previsíveis, sendo incapaz de lidar com a incerteza, e um dos principais desafios da manufatura é corresponder instantaneamente à demanda com produtos de alta qualidade (Ribeiro & Barata, 2011). A tão desejada diversidade de produção é alcançada, mas a impossibilidade de utilizar mais de um recurso de cada vez torna-se um grande ponto fraco no que diz respeito à fluidez de produção, e a impossibilidade de lidar com o inesperado tornam a arquitetura suscetível a imprevistos, sendo esses os calcanhares de Aquiles deste paradigma.

2.1.2. Sistemas Reconfiguráveis de Manufatura, RMS

O conceito de produção reconfigurável tem surgido como uma forma de obter funcionalidades mutáveis que ao mesmo tempo sejam escaláveis, onde todo o tipo de componentes, desde máquinas a células podem ser adicionadas, removidas ou modificadas sem por em causa o resto do sistema. Estas características permitiriam aos sistemas responder de forma rápida e consistente à mudança dos requisitos na produção, como descrito em (ElMaraghy, 2005).

A principal ideia dos RMS baseia-se em composições fabris modulares, onde existe uma relação da ferramenta com a máquina que a utiliza sendo assim possível reconfigurar o sistema, através de uma reconfiguração do controlador local à máquina (Mendes, Leitão, Colombo, & Restivo, 2008). Nesta arquitetura é defendida a utilização de hardware e software com características de *plug and play*, ou seja, a conexão destes módulos, idealmente, será realizada sem qualquer necessidade de reconfiguração ou remoção de outros equipamentos presentes no sistema.

Para que um sistema seja considerado um RMS, precisa ser reconfigurável por meio de componentes modulares, tanto em hardware como em software, de modo a facilitar a interligação entre componentes, caso contrário o tempo de reconfiguração será demasiado alargado. As características fundamentais a esse funcionamento prendem-se com a modularidade, integrabilidade, customização diagnosticabilidade e convertibilidade, (Koren et al., 1999). O requisito de modularidade faz com que os RMS sejam capazes de conjugar diferentes módulos de maneira a comportar funcionalidades complexas que individualmente o sistema não suportava.

Esta possibilidade de reconfiguração permite uma maior flexibilidade quando a procura do produto varia, pois apenas com uma troca de ferramenta e uma rápida reconfiguração, as novas necessidades são satisfeitas, e a produção pode ser retomada. Com a facilidade de trocar de ferramenta na mesma máquina, consegue-se uma reconfiguração e reutilização das linhas existentes, o que possibilita à mesma linha fabricar vários tipos de produto sem grandes remodelações e ainda ter alguma flexibilidade no que respeita às especificações de personalização (Koren et al., 1999).

Alguns estudos tendem a melhorar as técnicas de reconfiguração adotadas nestes sistemas, por exemplo em (J. Li, Dai, Meng, Dou, & Guan, 2009) onde são utilizadas redes de Petri para o controlo lógico da execução, sendo possível modelar a execução e assim reconfigurar automaticamente a produção baseando essa reconfiguração nos resultados das redes de Petri.

São de notar as vastas semelhanças entre os RMS e os FMS, contudo os paradigmas diferem na possibilidade dos RMS serem reconfigurados através da adição, remoção ou modificação de componentes sem afetar todo o sistema.

Resumindo, um RMS deve ser capaz de tratar novos produtos com pequenas reconfigurações e adaptar-se às novas condições e exigências dos mercados. Devido à sua componente modular, a integração com novas tecnologias também deve ser tida em consideração como de fácil aplicabilidade, visto que o novo módulo em nada depende dos restastes do sistema.

2.1.3. Sistemas Biónicos de Manufatura, BMS

Tal como descrito pela primeira vez em (Ueda, 1992) os Sistemas Biónicos de Manufatura são inspirados no funcionamento dos órgãos naturais, onde o sistema se baseia numa hierarquia, tal como no corpo humano. Como princípio, a existência de vários órgãos leva à divisão dos organismos em várias camadas. Assim sendo existe uma constante troca de informação entre estas camadas da hierarquia, o que permite uma maior cooperação e visa lidar com mudanças imprevisíveis em sistemas complexos. Este conceito foi desenvolvido na década de 90, onde se pretendeu adaptar as capacidades dos organismos biológicos, nomeadamente a autonomia e o comportamento espontâneo, e que estas fossem utilizadas nos sistemas de Manufatura. O sistema é visto como uma combinação de vários subsistemas autónomos que se combinam para formar uma determinada função. A troca de informação deve respeitar a hierarquia e manter-se constante ao longo da execução do sistema, quer no sentido da base para o topo como vice-versa.

Outras das vantagens deste tipo de arquiteturas são impulsionadas pela evolução biológica e seu funcionamento, nomeadamente a auto-organização, a evolução e adaptação a adversidades, em (Ueda, Hatono, Fujii, & Vaario, 2000), onde os organismos biológicos se caracterizam pela sobrevivência, autorreconhecimento, autorrecuperação e evolução, ultrapassando os problemas com base em dois tipos de informação, a *DNA-type* e a *BN-type*. À medida em que os sistemas vão evoluindo e com o passar do tempo, surgem novas gerações que adquirem a informação do tipo DNA, como se de uma herança genética se tratasse. Por outro lado, a infor-

mação do tipo BN, é adquirida por cada individuo, relacionada com as suas experiências e aprendizagens, permitindo que cada agente evolua de forma diferente, trazendo diversidade às soluções apresentadas.

2.1.4. Sistemas Holónicos de Manufatura, HMS

Outro dos paradigmas relaciona-se com os sistemas holónicos de manufatura, HMS do inglês *Holonic Manufacturing Systems*. Este paradigma é baseado no conceito de holon, inicialmente proposto por Arthur Koestler em 1968 (KOESTLER, 1968). A palavra holon surge da palavra grega "*holos*" que significa "um todo" combinada com o sufixo "*-on*" que por sua vez sugere "uma parte" ou "uma partícula". Com esta junção o que se pretendeu criar foi uma unidade básica de organização em sistemas biológicos ou sociais, que posteriormente foi adaptada para sistemas de manufatura em (Van Brussel, Wyns, Valckenaers, Bongaerts, & Peeters, 1998) surgindo os HMS.

No conceito de holon está implícita uma natureza de bloco autónomo e cooperativo, que nos sistemas de manufatura se destina a transformar, transportar, guardar ou validar informação ou objetos físicos. Outra das características chave deste conceito é a possibilidade de um holon ser uma combinação de vários outros holons e ao mesmo tempo ser parte de um outro holon de calibre superior, confirmando assim a ideia de um holon ser um todo e uma parte ao mesmo tempo. Esta dinâmica de cooperação trás a possibilidade ao sistema de responder a pedidos antes insatisfeitos pelo grau de exigência imposta (Cavalcante et al., 2010).

O surgimento dos HMS teve por base desenvolvimentos no campo dos IMS, do inglês *Intelligent Manufacturing Systems*, sendo que a primeira aplicação dos sistemas holónicos era baseada numa ideologia proposta também pelo filósofo Húngaro, Arthur Koestler, denominada de "*holarchy*". Esta espécie de arquitetura/hierarquia sem fim, ou seja, sem limitações tanto ascendentes como descendentes, pretendia a interação e cooperação entre as entidades para atingir objetivos gerais (Babiceanu & Chen, 2006).

Com este paradigma pretende-se dotar os sistemas de produção e manufatura de capacidades de cooperação entre entidades completamente autónomas e independentes, cada uma com os seus próprios objetivos. Existem vários exemplos onde se descrevem arquiteturas que tem por base o paradigma dos HMS.

Em (Vrba, Tichy, & Mařík, 2011) é proposto um sistema onde a existência de um tipo de agente intermediário, os *Midle-Agent* responsáveis por formar equipas entre eles de modo a disponibilizar tarefas de mais alto nível. Assim sendo a interação com os produtos é simplificada pois a estação pode apenas disponibilizar uma tarefa que por sua vez é composta por várias skills atómicas e cabe ao agente intermediário coordena-las.

Outro exemplo é a ADACOR (*ADAptative holonic COntrol aRchiteture*), primeiramente em (Leitão & Restivo, 2006) e mais recentemente em (Barbosa, Leitão, Adam, & Trentesaux, 2015) com a ADACOR^2, uma versão melhorada que utiliza os mesmos princípios e os interliga com um modelo de auto-organização bidimensional que permite reagir suave ou drasticamente a novas restrições no sistema sem que o desempenho global seja afetado. A ADACOR tem por base a

divisão das entidades constituintes em holons autónomos e com capacidades de cooperação, aproveitando as características de modularidade, descentralização, agilidade, flexibilidade e robustez que caracterizam este conceito. Esta arquitetura propõe uma divisão em quatro tipos de holons, os Product Holons, os *Operational Holons*, os *Task Holons* e os *Supervisor Holons*. Os *Product Holons* representam os produtos no sistema, estes são responsáveis por organizar e gerir a lista de tarefas necessárias à produção de cada produto. Os *Task Holons* por sua vez têm como função gerir a execução em tempo real das ordens de produção. Como o nome indica os *Operational Holons* representam os recursos do sistema, estes podem ser robôs ou postos de trabalho humanos, em suma estações onde a execução das tarefas de produção se realizam. Por fim os Supervisor Holons responsabilizam-se pela otimização do sistema.

PROSA (Product, Resource, Order, Staff Architecture), referenciado em (Van Brussel et al., 1998), é outro exemplo de uma arquitetura que tem por base este paradigma. Como o nome indica esta divide-se em três tipos básicos de holons, os *Product Holons*, os *Resource Holons* e os *Order Holons*. Existe também a possibilidade de considerar os *Staff Holons*, porém a sua presença no sistema não é obrigatória, visto que as suas capacidades se prendem com organização e cooperação de agentes básicos. À semelhança com a arquitetura descrita anteriormente, os *Product Holons* pretendem abstrair os produtos, os *Resource Holons*, abstrair os recursos e os *Order Holons* as tarefas. Os holons de produto e de recurso, interagem com vista a adquirir conhecimento sobre as características de determinado recurso. As interações entre o produto e os holons de ordem, destinam-se à partilha de informação relativa aos métodos necessários para produzir o produto. Por sua vez, os holons de recurso e de ordem interagem de modo a possibilitar a gestão do processo de execução das tarefas nos recursos.

Outros aspetos delineadores da arquitetura PROSA são os seus dois conceitos básicos. A agregação e a especialização, como descrito em (Van Brussel et al., 1998). A agregação pretende explicar a forma como os holons se organizam em questões de cooperação, formando conjuntos entre sim e criando um novo holon de cariz superior com a sua própria identidade. Por outro lado, a especialização pretende garantir a diferenciação entre os diferentes tipos de holons, consoante as suas características e funcionalidades no sistema.

Em suma, os HMS pretendem estabelecer, no contexto da manufatura, os benefícios que as entidades holónicas oferecem aos organismos vivos e às sociedades. São eles a estabilidade face a distúrbios, a adaptabilidade e a flexibilidade enquanto preservam a utilização eficiente de todos os recursos presentes em sistemas de manufatura. Por esses motivos, os HMS preenchem alguns dos requisitos imprescindíveis nos sistemas de manufatura de hoje em dia, sendo eles a reconfiguração, escalabilidade, flexibilidade e capacidade de resposta. Um mais pormenorizada descrição dos HMS pode ser encontrada em (Babiceanu & Chen, 2006).

2.1.5. Sistemas Evolutivos de Produção, EPS

Por último os sistemas evolutivos de produção, EPS do inglês *Evolvable Production Systems*, (Onori & Barata, 2009) representam sistemas de elevada autonomia e adaptabilidade de modo a possibilitar uma resposta rápida às adversidades ou alterações de especificação no que diz respeito à personalização dos produtos, (Cavalcante et al., 2010). Inicialmente abordados

como EAS, *Evolvable Assembly Systems*, em (Onori, Barata, & Frei, 2006) onde se pretendia resolver problemas como a capacidade evolução e adaptação de processos aquando a mudança das condições disponíveis. O termo evolutivo implica a capacidade do sistema em ajustar-se às condições atuais de forma autónoma. Em (Cavalcante et al., 2010) é sugerido que se um módulo falhar, pode ser perdida uma funcionalidade, mas o sistema deve ser capaz de criar novas coligações entre módulos para suprimir a falha do sistema.

O paradigma dos EPS propõem soluções que, apesar de serem baseadas em elementos simples, reconfiguráveis e orientados a tarefas específicas, permitem uma constante evolução e assim melhor resposta a imprevistos. Apesar das grandes parecenças entre os EPS e os RMS, (Onori et al., 2006) vem clarificar as diferenças entre os dois paradigmas.

Em (Barata, Camarinha-Matos, & Onori, 2005) é proposta uma arquitetura multiagente baseada no paradigma dos EPS, onde é introduzido o conceito de tarefa simples e tarefa complexa, pelo que uma tarefa complexa representa um grupo de tarefas simples ou conjugações de outas tarefas complexas, relembrando o conceito dos HMS e os seus holons.

O futuro aponta para a interação entre agentes inteligentes e seres humanos, e é nessa perspetiva que em (Frei, Ribeiro, Barata, & Semere, 2007) foi pensado um possível senário de interação baseado em EPS, inspirado no conceito de casas inteligentes.

Outra das aproximações tem em conta a capacidade de acoplamento automático ao sistema, ou seja, a capacidade de adicionar novos recursos e do sistema ser capaz de se auto-organizar, disponibilizando as novas skills ou tarefas para futuras requisições (Ribeiro & Barata, 2011), esta abordagem implica não só uma nova compreensão e controlo na arquitetura como algum estudo a nível de diagnóstico.

Mais recentemente em (André Dionísio Rocha et al., 2015) e (Di Orio, Rocha, Ribeiro, & Barata, 2015) é exposto um exemplo composto por módulos, distribuído, baseado numa arquitetura multiagente para o projeto PRIME, projeto esse que visa a criação de novas soluções de desenvolvimento de alta adaptabilidade, capacidade de reconfiguração no que respeita a sistemas de *plug-and-play*.

Assim sendo os EPS focam a sua abordagem na agilidade através da modularidade e da evolução passo a passo. Idealmente o seu propósito é adaptar-se dinamicamente às adversidades, como o surgimento de um novo produto, sem grandes consequências para o ambiente envolvente, por exemplo a adição ou remoção de elementos de produção, módulos de manufatura, em resposta à mudança nos requisitos do cliente deve ser suportada em tempo real e de forma automática, sem necessidade de parar todo o sistema para tarefas de reconfiguração.

2.2. Sistemas Multiagente, MAS

Como tecnologia geralmente utilizada para implementação dos sistemas descritos anteriormente, surgem os sistemas multiagente, referenciados em literatura como MAS, *Multi Agent Systems*, são sistemas constituídos por agentes autónomos que colaboram entre si de maneira a satisfazer as necessidades, tanto locais como globais do sistema. Tendo em consideração

estas caraterísticas os MAS são sistemas construídos em ambientes de produção e divididos elementarmente entre as partes constituintes do sistema, exemplos dessas partes são, os produtos, as máquinas, as ferramentas, entre outros. O intuito da divisão das partes é atribuir a cada uma, algumas características fundamentais ao trabalho colaborativo, como são o caso da autonomia, inteligência, sociabilidade e pro-atividade, como descrito em (Monostori, Váncza, & Kumara, 2006). Estas entidades denominam-se de agentes. Os agentes são entidade autónomas, pois através do conhecimento parcial do ambiente, devem basear as suas decisões de modo a controlar o seu estado e do meio em que está inserido. A capacidade de inteligência prende-se com a implementação de determinadas regras de raciocínio, aprendizagem e planeamento, que ajudam o agente a tomar decisões da melhor maneira. A característica de sociabilidade é garantida através da interação entre agentes e o meio envolvente, tomando decisões com base em troca de informações ou baseadas nos acontecimentos que os rodeiam. Por fim, a pro-atividade tem o objetivo de permitir ao agente adaptar os seus comportamentos às alterações nos meios envolventes ou quaisquer intervenções de entidades externas ao sistema.

Apesar da descrição acima, existe uma grande falta de consenso no que diz respeito à descrição dos agentes, no entanto a comunidade parece convergir em algumas características essenciais ao seu funcionamento. Sendo estas definidas pelo funcionamento autónomo, uma margem de aprendizagem, ou seja, evolutivo e com capacidade de executar tarefas (Monostori et al., 2006).

Tal como referido anteriormente os MAS têm tido recentemente um papel fundamental no desenvolvimento de novos paradigmas. Como esta tecnologia se baseia num controlo distribuído e nutre de um enorme potencial para resolver processos de decisão complexos e dinâmicos, permite que através de redes de entidades inteligentes se possam criar mecanismos de cooperação e interoperabilidade para que os agentes interajam de forma autónoma, (Mařík & McFarlane, 2005). O mesmo autor ainda menciona como se podem adaptar as tecnologias multiagente à indústria, referindo a extrema complexidade que por vezes a atribuição de tarefas e decisões de execução implicam, sendo o caminho a tomar nestes casos a descentralização do poder de controlo, permitindo melhorar a reconfiguração dos ambientes de produção e assim melhorar a reação à rápida evolução dos mercados, necessidades dos produtos e até em falhas na produção.

Noutras abordagens, o paradigma é utilizado para resolver problemas de escalonamento. Em (Kanaga & Valarmathi, 2012) é apresentada uma implementação desenvolvida através de uma plataforma multiagente onde se aplicam fundamentos destes sistemas para a gestão de pacientes num hospital. Esta implementação prova que nem só na indústria este paradigma mostra utilidade.

Um outro exemplo da aplicação de MAS, bem sucedido, em substituição de outras arquiteturas é apresentado em (Sabar, Montreuil, & Frayret, 2009), onde uma solução baseada num MAS veio substituir um método tradicional que se sustentava em modelos matemáticos para aproximar o problema apresentado no sistema, este modelo era posteriormente resolvido com o auxílio de programação matemática. A principal vantagem que o MAS trouxe em relação à solução através de modelos matemáticos, foi o encurtar no tempo de cálculo da resposta quando são precisas interações rápidas com o sistema.

Nos sistemas de produção é fundamental ter em conta dois processos fundamentais, o planeamento e o escalonamento da produção. A funcionalidade do planeamento é garantir a eficaz distribuição dos recursos presentes na linha e as operações necessárias à produção. Por outro lado, o escalonamento dedica-se à divisão de tarefas entre as máquinas constituintes do sistema. Como é possível observar, as definições são bastante idênticas, de maneira a simplificar todo o processo e integrar estas duas tarefas é proposta em (X. Li, Zhang, Gao, Li, & Shao, 2010) uma solução que permite melhorar o desempenho do sistema.

Com uma abordagem diferente, mas também com a intenção de abordar a integração das duas etapas referidas no parágrafo anterior, surgem as arquiteturas baseadas em otimização por colónia de formigas, descrito em (Blum, 2005). Como exemplo, em (Leung, Wong, Mak, & Fung, 2010) é proposta uma implementação baseada num MAS, que recriando o comportamento de uma colónia de formigas, pretende integrar o planeamento e o escalonamento do sistema. Supondo que cada formiga é um agente independente, cada uma deixa feromonas nos caminhos pelos quais passou e onde o tempo de execução foi o mais curto. Quantas mais formigas passarem pelo mesmo caminho, mais feromonas este terá e maior será a probabilidade de ser utilizado.

Em suma, os MAS têm tido um papel preponderante no desenvolvimento de novos paradigmas de manufatura, isto deve-se em parte ao fato dos princípios de autonomia, pro-atividade e sociabilidade resolverem muitos dos problemas presentes na indústria, nomeadamente problemas de versatilidade e interoperabilidade.

2.3. Simuladores de Ambientes de produção e manufatura

2.3.1. Simulação na manufatura

A produção e o desenvolvimento industrial são áreas que requerem sempre grande investimento, devido à abrangência das áreas envolvidas, como a Física, a Mecânica, a Hidráulica e a Pneumática, com intuito de reduzir os custos e aumentar a eficiência dos processos industriais, existe uma constante investigação de novas metodologias e técnicas de produção. Uma das soluções normalmente utilizadas, para reduzir o investimento inicial, é a simulação. Existem várias formas de simular sistemas industriais automatizados, entre elas, modelos mecânicos ou maquetes, contudo a técnica mais utilizada é a modelação através de modelos matemáticos ou sistemas computorizados. É notoriamente difícil reproduzir de forma fidedigna os comportamentos reais de máquinas industriais em maquetes, por vezes extremamente complexas de construir, e não só no meio industrial, noutras áreas a simulação nem sempre pode passar por modelos físicos, mas sim por modelos comportamentais descritos através de modelos, (Clark & Daigle, 1997).

A simulação ou simulação computorizada consiste na utilização de modelos matemáticos, que pretendem criar uma relação entre as variáveis de entrada e as de saída, Figura 2.1, permitindo reproduzir, de forma fidedigna, os comportamentos de sistemas reais. Com um modelo bem definido, é possível tirar vários partidos da simulação de sistemas, tais como, entender o comportamento do sistema, prever e avaliar possíveis pontos críticos e falhas na solução apresentada ou até mesmo estudar e testar novas configurações ou metodologias nos processos em causa.

Figura 2.1 Esquema de simuladores

Normalmente dotados da funcionalidade de acelerar ou atrasar o ritmo da simulação, caso o modelo utilizado esteja bem implementado, os simuladores podem dar novas perspetivas e informações num curto período de tempo. Pelas razões apresentadas a simulação assume cada vez mais um dos principais papéis na criação e otimização de processos industriais, contrariando a tendência do passado de tentativa e erro, que se torna dispendiosa, não só a nível financeiro como no tempo despendido, como descrito em (a M. Law & Kelton, 2000).

Por fim, outro dos critérios que impulsionou o crescimento da utilização de simulações no meio industrial prende-se com a questão de formação, evitando assim problemas de segurança e conflitos de utilização, caso os recursos sejam escassos. Todos estes fatores têm contribuído para um desenvolvimento de plataformas de simulação inovadoras, que primam pela facilidade de utilização, sofisticação e cada vez mais fiel e precisa representação dos comportamentos físicos (Buccioli, Zorzal, & Kirner, 2006).

Atualmente existem duas abordagens divergentes no que diz respeito aos modelos aplicados em simuladores, a abordagem exploratória e a preditiva.

- A abordagem exploratória destina-se a criar hipóteses baseadas em observações passadas. Esta abordagem normalmente tem foco em aspetos específicos do problema, como exemplo em (Drogoul & Ferber, 1994) e uma colónia de formigas.

- A abordagem preditiva é utilizada para prever estados futuros e avaliar tendências, por exemplo em (Jacintho, Batista, Ruas, Marietto, & Silva, 2010) onde um modelo preditivo foi utilizado para, em conjunto com alguns parâmetros de entrada, prever a propagação da febre do Dengue no Brasil.

2.3.2. Tipos de simulação

Nos simuladores atuais são possíveis distinguir dois tipos de simulação, a simulação de eventos discretos, DES, *Discrete Event Simulation* e a simulação geométrica, GS, *Geometric Simulation*, ambas descritas pormenorizadamente em (Klingstam & Gullander, 1999).

- DES ou simulação por eventos discretos, como o nome indica, consiste na simulação de eventos que ocorrem numa determinada unidade de tempo do sistema, (Hoeger & Jones, 1996). Esses eventos ocorrem em todas a entidades presentes no sistema, e implicam uma alteração nas variáveis e estados dessas entidades. Exemplos de simuladores na Tabela 2.1.

- GS ou simulação geométrica, sugere uma simulação da geometria, ou da física envolvente nos processos a simular. Geralmente o ambiente utilizado é tridimensional, o que é bastante atrativo ao utilizador, face à sua parecença com o mundo real. Simuladores com este comportamento criam um modelo composto por uma componente lógica e outra de visualização, permitindo assim representar fidedignamente o sistema a simular (Klingstam & Gullander, 1999). Exemplos de simuladores geométricos encontram-se na Tabela 2.2.

Outra maneira de diferenciar os simuladores é através do método de utilização ou programação. Existem simuladores de mais alto nível, que se caracterizam pela programação por blocos, onde o utilizador usa blocos pré-programados, com possibilidade de parâmetros de entrada ou não, o que permite construir um simulador de simples programação, mas bastante limitado no que diz respeito às possibilidades de configuração. Por outro lado, a existência de simuladores programáveis, restringe a sua utilização a programadores, no entanto são melhor configuráveis e oferecem maior variedade de possibilidades/especificações, (A. M. Law & Mccomas, 1998).

Tabela 2.1 Simuladores DES, retirado de (Veiga, 2013)

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

Simprocess	CACI Products Company (http://www.caciasl.com/)
SIMUL8	SIMUL8 Corporation Products (http://www.simul8.com/)
SLX	Wolverine Software (http://www.wolverinesoftware.com/)
Spar	Clockwork Solutions (http://www.clockwork-solutions.com/)
Witness	Lanner Group (http://www.lanner.com/)

Os dois tipos de simulador são bastante abrangentes, permitindo os mais diversos testes e garantindo um grande leque de possibilidades na representação de ambientes. Na próxima tabela surgem os simuladores do tipo GS, que são maioritariamente utilizados em representações mais fidedignas e com ambientes de simulação mais aprimorados.

Tabela 2.2 Simuladores GS, retirado de (Veiga, 2013)

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

2.3.3. Simuladores

Em seguida são apresentados quatro simuladores que atualmente têm bastante impacto na indústria. Todos eles são simuladores DES e possuem uma componente de visualização, que permite visualizar o que acontece durante a simulação. Pretende-se descrever o modo de funcionamento dos simuladores, destacando as suas qualidades de distinção.

2.3.3.1. ARENA

O Arena é um simulador, fornecido pela Rockwell Automation, de alto nível, onde o utilizador pode criar os seus modelos de simulação através de uma tecnologia de *drag-and-drop*. O simulador dispõe de uma arquitetura modular, o que revela grande dinâmica no controlo dos blocos, separando as entidades e os seus processamentos. A utilização de módulos para definir todas as entidades da simulação, sugere uma fácil configuração de sistemas distribuídos e consequente melhoramento do desempenho dos modelos, (Rockwell Automation, 2016).

Como dito acima, é disponibilizada uma vasta lista de módulos pré-programados, isto liberta o utilizador da necessidade de ter conhecimentos de programação, facilitando a modelação e abrangendo um maior leque de clientes possíveis. Apesar da facilidade de utilização desta ferramenta, a sua utilização encontra-se limitada aos módulos existentes. O simulador não nos dá a possibilidade de alterar tanto as variáveis de entrada como os comportamentos do módulo, estando sujeitos à preconceção do modelador, e da sua intenção.

Normalmente utilizado para sistemas de logística e análise de negócios o Arena não dispõe de um ambiente de visualização muito desenvolvido, pelo que são percetíveis os modelos oferecidos, mas ficam longe do seu aspeto real.

2.3.3.2. V-REP

O V-REP ou Virtual Robot Experimentation Platform, (Coppela Robotics Gmbh, 2016), é um dos simuladores mais versáteis e completos do mercado. Pertence à empresa Coppela Robotics Gmbh e ao contrário do simulador da Rockwell Automation o V-REP disponibiliza o controlo da modelação de seis maneiras diferentes, sendo elas *Embedded script*, *add-on, plugin, remote API cliente, ROS node e Custom cliente/server*, como descrito em (Freese, Singh, Ozaki, & Matsuhira, 2010).

O V-REP também apresenta um funcionamento modular, o que permite distribuir, se necessário, os mecanismos de controlo. Apesar da sua capacidade de alteração dos controladores de cada módulo, o simulador é que disponibiliza a biblioteca com os tipos possíveis de elementos no sistema. Esta definição limita a escolha dos módulos, mas já trás a liberdade de alterar as variáveis e o comportamento dos sistemas individualmente.

2.3.3.3. SIMIO

SIMIO ou SImulation Modeling framework based on Intelligent Objects, é oferecido pela SIMIO LLC. Esta ferramenta de simulação em muito se assemelha ao Arena, tal é justificável devido ao facto das pessoas que o desenvolveram e implementaram terem sido as mesmas. A sua filosofia de modelação é através da tecnologia de *drag-and-drop*, e tal como já foi dito limita em muito as possibilidades de utilização.

As diferenças são o suporte de arquiteturas descentralizadas e a sua arquitetura orientada a objetos. Uma descrição mais detalhada pode ser encontrada em (SIMIO LLC, 2016) e em (Pegden, 2007).

2.3.3.4. DDDSimulator

O DDDSimulator, simulador da (Technology Transfer System S.r.l., 2016), baseia-se na execução de eventos registados numa escala temporal. Esta escala temporal é responsável pela sincronização do ambiente de visualização, onde se integram os modelos 3D e toda a parte gráfica, e da parte lógica, responsável pelo controlo e comportamento dos vários módulos constituintes da simulação. Para que esta sincronização seja possível é utlizado um Relógio de Simulação, o qual é responsável pela transição entre os eventos programados, Figura 2.2.

A representação gráfica e o controlo da simulação funcionam em camadas diferentes, Figura 2.3, isto faz com que os módulos lógicos não estejam acoplados a nenhum recurso 3D, mas consigam escrever e ler o estado de qualquer entidade da simulação.

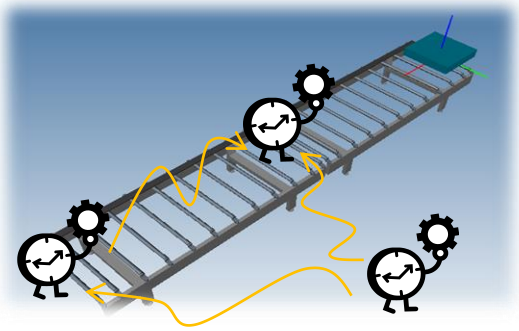

Figura 2.2 Simulador DDD
Relógio de sincronização

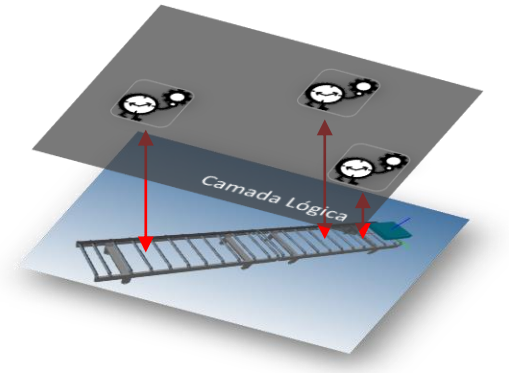

Figura 2.3 Simulador DDD
Separação de camadas

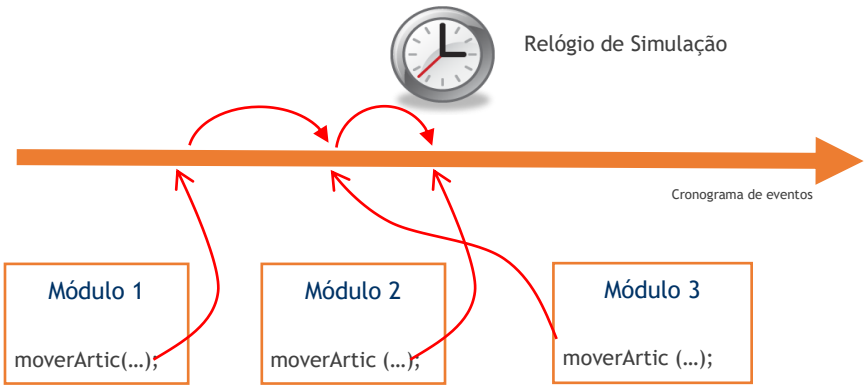

Figura 2.5 Simulador DDD Agendamento de eventos

Como referido anteriormente os controladores lógicos estão divididos em módulos, estes módulos funcionam de forma independente, mas têm total acesso ao ambiente de simulação, visto conseguirem gerar e publicar eventos na escala temporal, Figura 2.5. Isto permite uma grande flexibilidade do sistema no que respeita à gestão de eventos, garantindo assim uma execução de eventos de forma paralela. Assim sendo, quando acabam de ser processados todos os pedidos agendados a simulação é terminada.

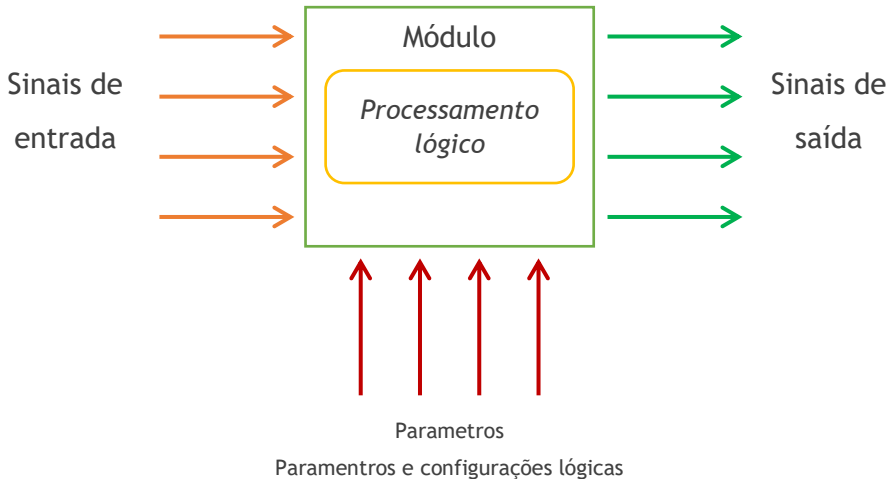

Figura 2.4 Simulador DDD esquema de módulo

Os módulos lógicos, para além da sua capacidade de interagir com o sistema, também podem comunicar entre si através do envio e receção de sinais, o funcionamento desta comunicação assemelha-se em muito ao funcionamento de circuitos elétricos. A codificação dos módulos no simulador DDD, carece de duas etapas. A primeira etapa rege-se pela definição de um código padrão, sequencial em Java, que define o comportamento de um tipo de módulo. Por outro lado, na segunda etapa é necessário instanciar, ou até mesmo interligar, cada objeto da simulação através duma das classes criadas anteriormente.

Para interagir com o meio envolvente, e retratando o mundo real, os módulos podem ser definidos com portos de entrada e de saída, de configuração/informação, Figura 2.4.

Segue-se uma tabela de comparação entre os simuladores analisados, destacando alguns dos indicadores preponderantes na utilização dos mesmos.

Indicadores	*Arena*	*V-REP*	*SIMIO*	*DDDSimulator*
Método de programação intuitivo	✓	✗	✓	✓
Liberdade de programação	✗	✓	✗	✓
Simulação por eventos	✓	✗	✓	✓
Simulação de comportamentos físicos	✗	✓	✗	✗
Aspeto visual atrativo	✗	✓	✗	✓
Funcionamento modular	✓	✓	✓	✓

2.4. Sistemas CAD

A componente visual nos simuladores tem, cada vez mais, um papel fundamental na decisão de que solução utilizar. Por isso mesmo é necessário recorrer a sistemas de renderização para garantir que a representação do modelo é no mínimo percetível. Estes motores de renderização, designados por sistemas CAD, Computer Aided Design ou Desenho Assistido por Computador, permitem a modelação de objetos tridimensionais, recorrendo a formas básicas como planos, linhas, curvas, polígonos, cubos, entre outros. Através da conjugação de várias formas é possível representar qualquer objeto existente, como descrito em (Shapiro & Farin, 2002).

Dada a importância destes elementos, foram considerados três ferramentas do tipo CAD, descritas a seguir, AutoCAD, SolidWorks e Blender.

2.4.1.1. AutoCAD

O AutoCAD é um programa desenvolvido pela empresa AutoDesk, uma das grandes líderes do mercado de sistemas CAD atualmente.

Esta ferramenta foca-se no design dos modelos tridimensionais, oferecendo ao utilizador uma vasta gama de opções e funcionalidades técnicas de graduação e escala, ou seja, ajudas no traçar, no esboço e na medição de comprimentos. Como é um programa tão completo, é pago, no entanto tem uma versão educacional, que apesar de limitada, permite ter uma primeira experiencia com as suas funcionalidades, (AutoDesk, 2016).

2.4.1.2. SolidWorks

Fornecido pela Dassault Systèmes, o SollidWorks é dos três programas apresentados o mais completo. Para além da possibilidade de desenho do aspeto exterior, a modelação de peças tridimensionais nesta ferramenta pode passar pela definição do material utilizado e das texturas adequadas. As vantagens são inúmeras, para além de disponibilizar um preciso método de medição, e um modo para criar modelos complexos compostos por peças individuais, o SolidWorks oferece uma biblioteca com variados objetos preconcebidos. Tal como o sistema anterior, o SolidWorks é um sistema pago, mas que disponibiliza uma versão para estudantes, (Dassault Systemes SolidWorks Corporation, 2016).

2.4.1.3. Blender

Blender é das três ferramentas apresentadas, a única sem custos de aquisição. Fornecida pela Blender Fondation, o seu objetivo é satisfazer as necessidades de artistas e pequenas equipas de design. O programa permite modelar e definir animações, sendo mais utilizado no fabrico de jogos do que em modelações industriais. Apesar disso é uma ferramenta com grande potencial, devido a forte contribuição comunitária. A única desvantagem é a interface ser pouco intuitiva o que requer algum treino e tempo despendido na sua inicialização, (Blender Foundation, 2016).

Em seguida uma tabela de comparação entre os sistemas de **CAD** referidos, de modo a ter uma perceção comparativa entre as hipóteses disponíveis no mercado.

Indicadores	*AutoCAD*	*SolidWorks*	*Blender*
Formatos de exportação	✗	✓	✗
Qualidade do detalhe	✓	✓	✓
Facilidade de manuseamento	✗	✗	✓
Acessibilidade (pago)	✓	✓	✗

2.5. Conclusão

Como é percetível a quem consultar a informação detalhada acima, os sistemas de manufatura evoluíram ao longo do tempo, partindo de sistemas centralizados e baseados em simples regras, para sistemas distribuídos com capacidades autónomas e alguma inteligência à mistura. Vários foram os paradigmas que surgiram para tentar colmatar as necessidades emergentes, contudo todos têm as suas vantagens e as suas desvantagens. As características fundamentais, que são cumpridas por todos os paradigmas são a robustez, a adaptabilidade e a flexibilidade de reconfiguração.

As abordagens multiagentes vieram trazer novas alternativas aos sistemas, introduzindo a versatilidade, a interoperabilidade e a escalabilidade necessárias para que seja possível adotar os novos paradigmas às linhas existentes.

No outro prato da balança, os simuladores, também têm evoluído a cada nova versão atualizada. A fiabilidade que hoje é alcançada em simulações permite prever um funcionamento perfeito à primeira tentativa, suprimindo a fase de testes em sistemas físicos e reduzindo, a zero, os acidentes por esse método de implementação causados. As arquiteturas descentralizadas e modulares vêm permitir uma ponte entre os paradigmas da produção em massa e a integração com os sistemas mais avançados. Por último, os sistemas de CAD conseguem atualmente criar autenticas obras primas na arte de imitar o objeto real. Esta capacidade ajuda não só no aumento da confiança depositada nos sistemas de modelação como na melhoria do aspeto visual dos componentes e consequente aumento da utilização.

Por último, a transição entre os sistemas atuais e sistemas do futuro deve ser bem planeada e simulada. Uma mudança radical nas linhas de montagem poderia por em causa, não só a subsistência das empresas como a estabilidade dos mercados. Deste modo é essencial que a mudança seja progressiva e faseada. E é neste contexto que são fundamentais as integrações entre as tecnologias existentes, de modo a aproveitar o que há de melhor para fazer algo ainda mais eficiente e proveitoso.

3

3. Arquitetura

Este capítulo tem como objetivo a apresentação da arquitetura proposta por esta dissertação. Tendo em conta o trabalho descrito no capítulo anterior, com a evolução das tecnologias surgem novos paradigmas, e para colmatar necessidades emergentes são precisas soluções inovadoras. No que diz respeito à indústria, o caminho é evoluir para sistemas de manufatura mais robustos e reconfiguráveis. No entanto, muitas das soluções atualmente existentes exigem uma reconfiguração completa do sistema, o que pode por em causa a subsistência da empresa. Assim sendo é importante encontrar uma alternativa mais rápida, eficaz e barata de reconfigurar a produção de maneira a responder mais eficientemente às necessidades de produção sem ter de construir um sistema novo de raiz, e idealmente sem comprometer a produção. Foi nesse sentido que foram desenvolvidos os **MAS**, que capacitam os sistemas de maior flexibilidade e adaptabilidade, permitindo reconfiguração e ajuste quase instantâneo às adversidades ou necessidades que surgem.

Apesar de todo o esforço desenvolvido, no que diz respeito à configuração de sistemas adaptáveis é sempre necessário montar fisicamente o sistema, não só para testar o resultado final, mas para pôr em prática testes e experiências de configuração. E é neste sentido que surge a necessidade de implementar sistemas computorizados capazes de simular, com a maior precisão possível, processos industriais. Estes simuladores permitem não só testar o produto final sem despender todos os recursos necessários que isso implica, como disponibiliza uma ferramenta de formação evitando problemas de segurança e conflitos no que diz respeito à utilização de recursos.

A arquitetura proposta está enquadrada no paradigma dos **EPS**, descrito no capítulo anterior, e visa a integração entre um simulador de plataformas industriais e um sistema multiagente, responsável por controlar o sistema de produção, neste caso a simulação, permitindo assim que sejam criados ambientes de simulação de fácil configuração para teste de novos protocolos, novas arquiteturas, novas ideias e conceitos no que respeita ao setor industrial composto por linhas de produção reconfiguráveis. Utilizando um simulador com funcionamento modular e conceitos dos **MAS**, pretende-se projetar uma solução viável independentemente da tecnologia utilizada.

3.1. Arquitetura do sistema

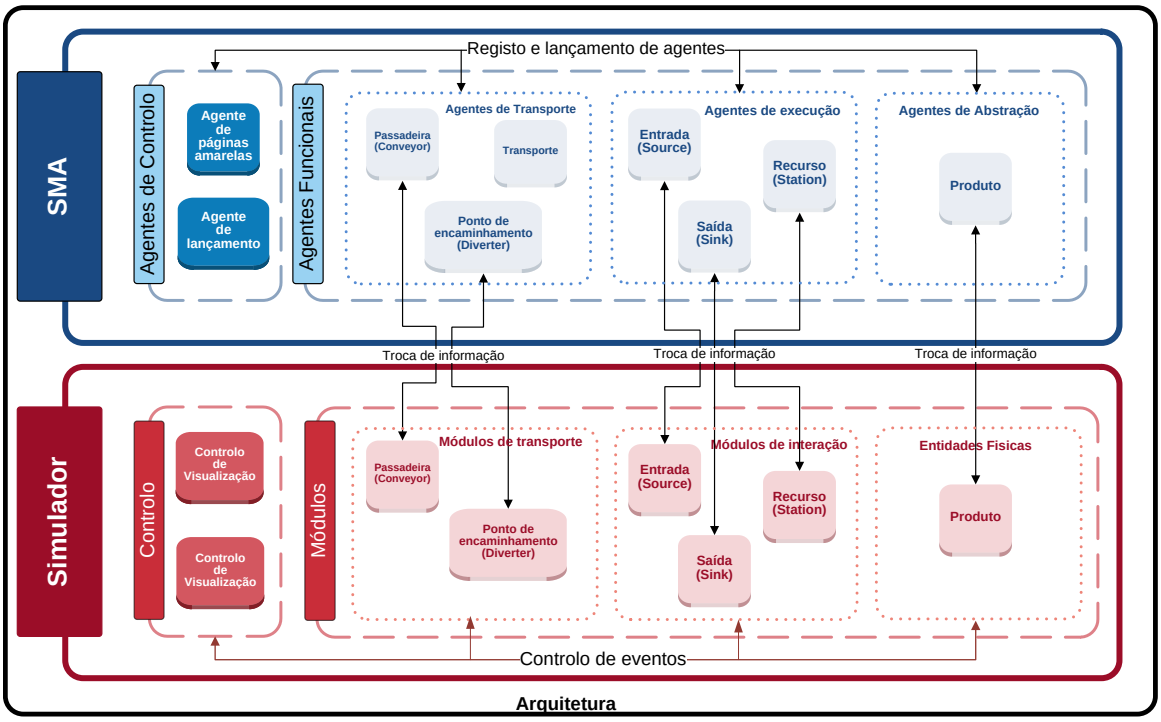

Figura 3.1 Desenho geral da arquitetura

Como já dito neste capítulo, a arquitetura pretende integrar dois sistemas já existentes. Para garantir que a mesma é transversal no que diz respeito, tanto à tecnologia utilizada no simulador, como na plataforma utilizada para desenvolver os agentes, segue-se uma descrição abstrata sobre as capacidades necessárias de cada uma das plataformas integrantes.

Na Figura 3.1 está uma representação geral da arquitetura proposta, que se divide em dois principais núcleos, o simulador e o sistema multiagente. As interações entre estes dois sistemas são feitas diretamente entre os módulos e os agentes onde podem ocorrer todos os tipos de interações, desde troca de informação, execução de tarefas ou despertar de eventos.

A arquitetura apresentada sugere uma abstração total do simulador por parte do controlo do estado dos produtos. Como o que se pretende é que o simulador seja uma representação fiel da realidade, as caraterísticas que este implementa devem reger-se apenas às capacidades das entidades que estes representam, mantendo a simplicidade e garantindo a modularidade das entidades de modo a que a sua implementação seja o mais transparente e direta possível.

Por outro lado, encontramos o **MAS**, ou SMA, sistema este que se pretende que seja capaz de implementar vários tipos de agentes. Agentes que disponibilizem vários comportamentos essenciais à comunicação e interoperação, não só entre eles como também com os módulos do simulador. Tal como é requisito no simulador, no **MAS**, pretende-se que todos os agentes possam ser independentes entre si e que sejam autónomos, tentando satisfazer os pedidos que lhes chegam e as funções que os caracterizam da maneira mais rápida e eficiente possível.

Duma forma geral as capacidades dos agentes regem-se pelas responsabilidades no que diz respeito à noção de estado e implementação de todos os protocolos de controlo, ou seja, todas as atividades que se relacionem com aceitação de novos produtos, o controlo de ocupação dos módulos e até mesmo a disponibilidade do módulo no sistema, deve ser controlada pelo agente. Para que tal seja de possível implementação foram idealizados vários tipos de agentes, que pretendem abstrair os módulos genéricos de uma linha de montagem e integrar a maioria dos sistemas industriais que estão implementados atualmente. Na próxima seção são descritos os comportamentos e funcionalidades dos agentes pensados para esta arquitetura.

3.2. Agentes

Para implementação desta arquitetura pensaram-se em nove tipos de agentes, divididos em dois principais grupos, os agentes de controlo e os agentes funcionais. O grupo de agentes de controlo tem como seus constituintes o agente de lançamento e o agente de páginas amarelas, onde as suas funcionalidades se pretendem de integração do sistema, permitindo assim garantir a integridade do mesmo. Em contrapartida, o grupo de agentes funcionais pretende agregar todos os agentes responsáveis por abstrair os módulos físicos ou comportamentos, como é o caso do transporte. Este segundo grupo reúne ainda três subgrupos, os agentes de transporte, os agentes de execução e os agentes de abstração. O subgrupo dos agentes de transporte tem como objetivo agrupar todos os agentes que são responsáveis pelo transporte de produto entre estações, tais como as passadeiras, os pontos de encaminhamento e até o transport agent. Outro dos subgrupos é o dos agentes de execução, este pretende aglomerar todos os agentes responsáveis por interagir diretamente com o produto, tais como as entradas, as saídas e as estações que efetuam tarefas nos produtos. Por último o subgrupo dos agentes de abstração tem como objetivo reunir as entidades mais abstratas, como é o caso do product agent.

Segue-se uma tabela onde se pretende apresentar uma breve descrição dos agentes constituintes da arquitetura, divididos pelos grupos anteriormente descritos, Tabela 3.1.

Tabela 3.1 Agentes constituintes da arquitetura

Grupo	Sub-Grupo	Agente	Função
Agentes de controlo		Agente páginas Amarelas	Garantir a integridade do sistema, mantendo uma lista atualizada dos serviços ou skills disponíveis e os identificadores dos agentes que as disponibilizam. Desta forma é possível procurar pelos agentes que disponibilizam determinada skill e obter o seu identificador.
		Agente de lançamento	A função deste agente é lançar os vários agentes na plataforma multiagente, registando os mesmos no controlador respetivo.

Agentes funcionais	Agentes de transporte	Agente de passadeira (Conveyor Agent)	Pretende-se com este agente abstrair módulos de comprimento fixo que permitam alocar um ou mais produtos. A sua função é interagir com o módulo do simulador, e garantir um controlo estável e adequado às suas características. Estes módulos têm como função permitir o transporte de produtos entre estações e/ou pontos de encaminhamento.
		Agente de Ponto de Encaminhamento (Diverter Agent)	A função de um ponto de encaminhamento é simular nós entre passadeiras, assim sendo, o seu comportamento é semelhante a um cruzamento rodoviário, onde o produto, consoante o destino, toma rumos diferentes.
		Agente de Transporte (Transport Agent)	O transport agent é responsável por abstrair todo o processo de transporte desde a origem até ao destino. Assim sendo o produto apenas tem de dizer para onde quer ir, transferindo toda a responsabilidade do cálculo de rotas e interação com outros agentes para o transport agent.
	Agentes de execução	Agente de Entrada (Source Agent)	Agente responsável por abstrair o módulo de entrada de produtos no sistema. Também é o agente responsável por lançar os agentes de produto na plataforma.
		Agente de saída (Sink Agent)	Tem como função representar a saída do sistema, não só recebendo os produtos que já completaram as suas operações como tem a responsabilidade de apagar as referências ao produto e registos do sistema.
		Agente de recurso (Station Agent)	A função do station agente é abstrair as máquinas de execução como robôs ou estações de trabalho que têm como objetivo operar sobre os produtos. São estes os agentes que disponibilizam as skills ou serviços que, mais tarde, serão procuradas pelos product agentes.
	Agentes de abstração	Agente de produto (Product Agent)	A função do product agent é abstrair o módulo físico do produto, sendo responsável por gerir as tarefas que este pretende realizar e interagir com os agentes necessários ao seu transporte e realização das tarefas.

3.3. Conceitos primários

Para que possamos entender melhor as interações e o papel de cada agente na arquitetura é necessário assumir alguns conceitos. Tais como o conceito de skill, o conceito de ligação entre dois módulos e o modo como estes são referenciados na plataforma.

3.3.1. Skill

De entre os conceitos chave surge o conceito de skill ou serviço. A ideia desta definição é dotar os agentes de funções que posteriormente são disponibilizadas num registo comum onde podem ser consultadas por outros agentes e assim dar a conhecer como cada um dos agentes pode contribuir para o resultado final.

Utilizando como exemplo a Figura 3.2, onde se representam duas estações diferentes, podemos ver que os módulos são abstraídos, cada um pelo seu agente, e que estes disponibilizam uma série de skills ou comportamentos.

O conceito principal é dotar o agente responsável, pelo módulo, de ferramentas de registo num sistema de páginas amarelas, que funciona como um registo global, registo este acessível por qualquer agente do sistema e onde podem ser procuradas todas as skills disponíveis.

Aquando a criação ou lançamento de um novo produto no sistema, este surge com total desconhecimento de onde estão as estações ou que estações disponibilizam os serviços necessários à sua conclusão, assim sendo é necessário recorrer ao serviço de páginas amarelas para as localizar.

As skills podem ser descritas pelos mais variados parâmetros, entre eles, o nome, a duração, o agente que as executa, a localização onde é disponibilizada, entre outros, permitindo ao produto avaliar os vários fatores e decidir qual das opções disponíveis deve escolher.

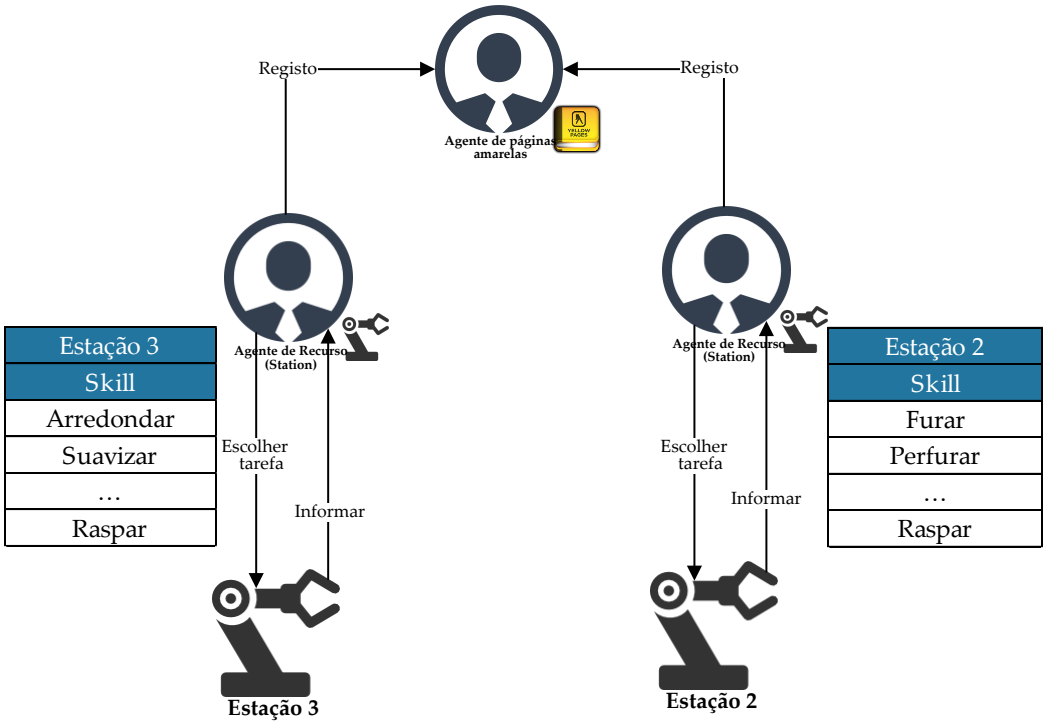

Figura 3.2 Interação de módulos com o seu agente

3.3.2. Ligações

Para aumentar a versatilidade do sistema, não só a procura e oferta de skills deve ser dinâmica. É necessário ter também em conta o deslocamento dos produtos. Nas soluções atuais os percursos dos produtos são estáticos e nunca dependem das condições do momento, para que tal seja corrigido, ou melhorado, é necessário pensar de forma inovadora em como definir o transporte. O transporte é realizado entre estações, e um dos principais fatores é como podem ser as ligações entre estas dinamizadas. E é neste contexto que surge a oportunidade de aplicar alguns conhecimentos adquiridos noutras áreas, como é o caso de redes ou telecomunicações. Para agilizar o sistema de transporte as ligações entre os módulos serão implementadas como tabelas de encaminhamento, Figura 3.3, assim sendo se houver uma alteração na linha, não é necessária uma total reconfiguração, mas apenas recorrer a um algoritmo de otimização de percursos, como é o caso do aplicado em (A. Rocha, 2013), onde o algoritmo de Dijkstra foi utilizado para o cálculo do caminho mais curto.

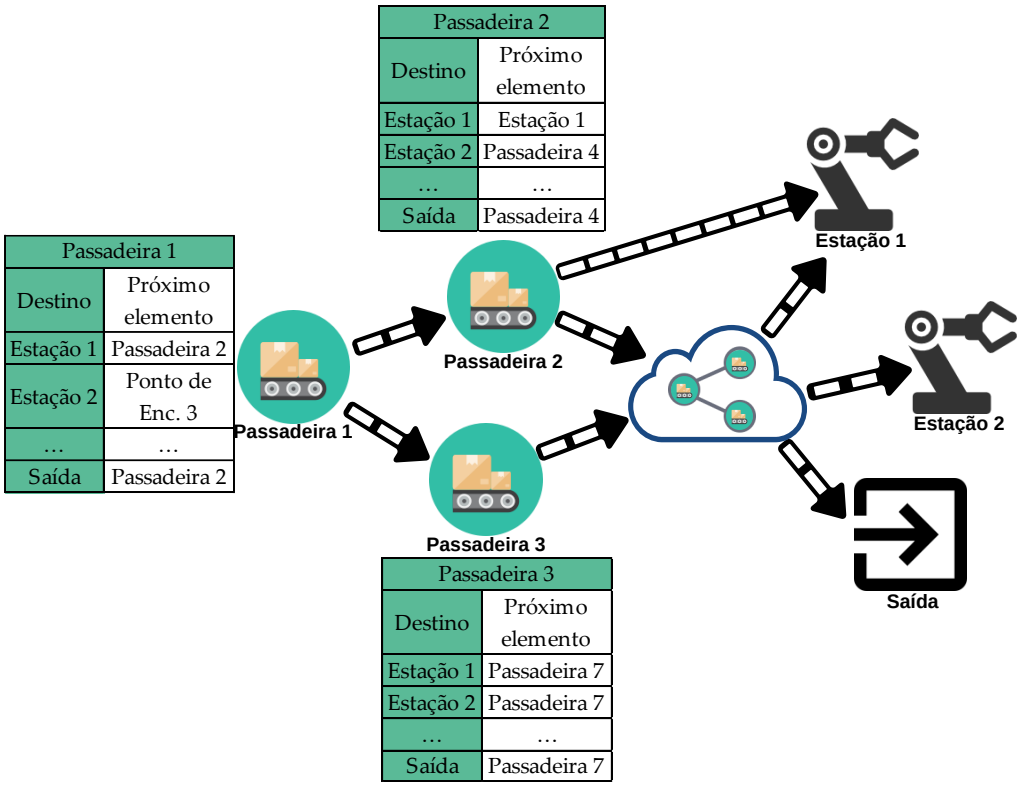

Figura 3.3 Esquema representativo de tabelas de encaminhamento

Deste modo é possível reduzir a complexidade do sistema, pois cada nó da rede, neste caso, cada módulo, não precisa de ter conhecimento de toda a rede, o que tornaria todo o processo de pesquisa lento, mas apenas saber qual o próximo módulo do melhor caminho para aquele destino.

É de frisar que apenas os módulos de execução devem constar como possíveis destinos nas tabelas de encaminhamento, visto que o sistema de transporte é abstraído por uma entidade responsável. Entidade esta que lida com os módulos intermédios e permite que o transporte do produto seja tão simples como um pedido, ver Transport Agent.

3.3.3. Referências fixas

De modo a permitir todas as pesquisas e interações referidas anteriormente é necessário ter um prévio conhecimento de alguns agentes, tal como é o caso do agente das páginas amarelas e do transport agent.

Estas duas entidades devem ser do conhecimento global do sistema pois funcionam como moderadores e garantem a estabilidade e consistência do ambiente.

3.4. Agentes de execução

Para um funcionamento adequado desta arquitetura foram idealizados três tipos de agentes de execução. Tal referência surge devido à sua interação com o produto, pois são estes os agentes responsáveis pela sua criação, alteração e destruição.

3.4.1. Source Agent

O primeiro destes agentes, e o responsável pela criação dos produtos, é o source agent. Tal como o nome indica este é o agente responsável por abstrair o módulo de entrada de produtos na linha, por este motivo é um agente com um comportamento passivo em relação ao seu módulo. As principais funções que se pretendem implementar com este agente são de guardar o produto até que haja espaço na linha para que este seja alocado no próximo módulo e saber quais os módulos este está acoplado, para permitir ao transport agent encaminhar o produto da maneira mais eficiente desde a sua criação.

3.4.2. Sink Agent

Como o nome indica este é o último agente com que o produto tem contato no sistema, o sink agent. As responsabilidades deste agente devem restringir-se a alocar o produto no módulo físico da saída quando este está completo, para que possa ser apagado do sistema, simulando a saída de uma linha de produção.

3.4.3. Station Agent

Por último, o station agent é o agente responsável por abstrair estações de trabalho, pontos estes que podem representar robôs ou postos de trabalho humano. As interações do station agent com o produto são de maior complexidade, visto que são as estações que disponibilizam as skills para completar os produtos que entram no sistema. Ao ser lançado no sistema, o station agent deve registar no serviço de páginas amarelas todas as skills que o seu módulo físico pode oferecer ao sistema, como é possível observar na Figura 3.4. Após esse registo o agente entra num estado de espera por pedidos de execução provenientes dos agentes de produto.

É com esta interatividade que se garante uma maior flexibilidade e eficiência na distribuição das skills pelo sistema, pois se várias estações oferecerem a mesma skill o produto pode escolher qual destas deve utilizar tendo em conta fatores como a disponibilidade, tempo de transporte, custo de execução entre outros fatores determinantes para a eficiência da produção.

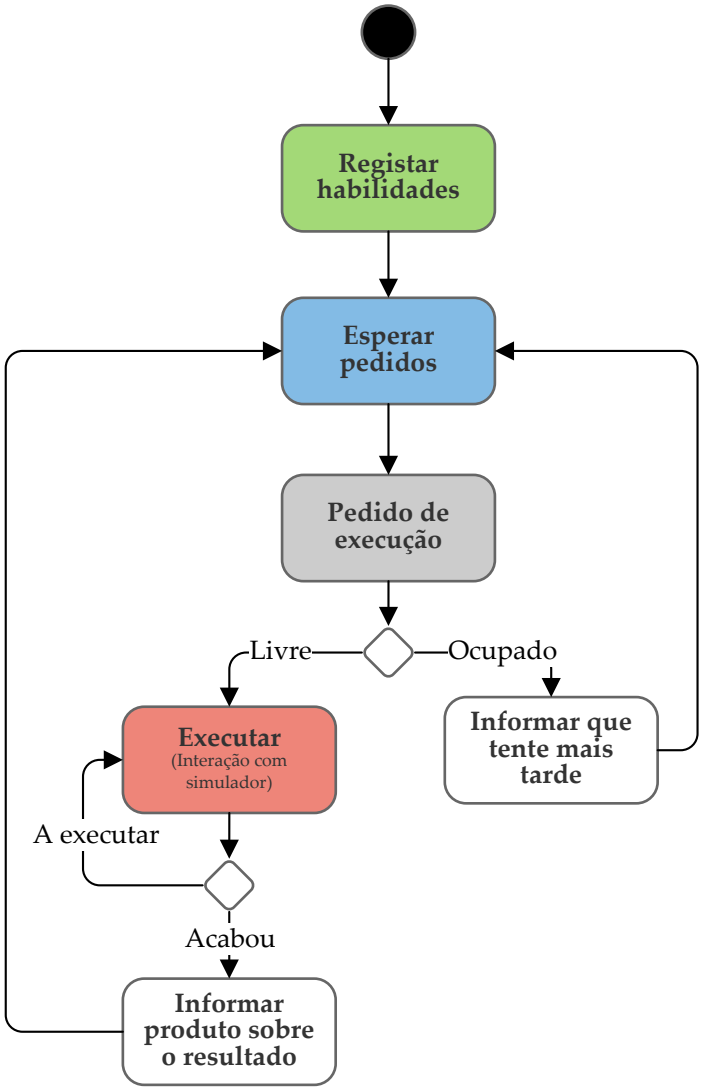

Figura 3.4 Diagrama de atividade das estações

3.5. Agentes de transporte

O sistema de transporte é composto essencialmente por dois tipos de elementos, as passadeiras e os pontos de encaminhamento. Devido às diferenças destes dois módulos é necessário criar dois tipos de agentes, com comportamentos e funções diferentes, de modo a mapear com eficiência o comportamento de cada um.

3.5.1. Conveyor Agent

O primeiro dos agentes de transporte é o agente que abstrai as passadeiras, módulos estes responsáveis por transportar os produtos entre encaminhadores e/ou estações. Para tornar o sistema mais dinâmico, as passadeiras podem ter comprimentos diferenciados, o que implica que o seu agente saiba quantos produtos cabem na passadeira que este representa. De modo a garantir que toda a inteligência está do lado do agente, este é responsável por controlar quantos e onde estão os produtos na passadeira, permitindo assim que o módulo da passadeira represente fidedignamente uma passadeira real, onde apenas existe o botão de funcionamento ou paragem.

Figura 3.5 Representação de passadeira

Como foi dito anteriormente todos os módulos do transporte têm de conhecer o próximo módulo de maneira a que os agentes de execução sejam alcançáveis. Deste modo, e tal como acontece com os agentes descritos anteriormente, existe uma primeira fase de inicialização onde o agente carrega as informações sobre as suas ligações e os destinos disponíveis no sistema.

Outra das caraterísticas fundamentais das passadeiras é a alocação de um ou mais produtos ao mesmo tempo, assim sendo, quando um produto se move é necessário atualizar a posição de todos os produtos que se encontram na passadeira, deste modo, e particularmente no caso de passadeiras, precisamos ter conhecimento de todos os produtos que são da responsabilidade do módulo. Como exemplo na Figura 3.5, é guardada uma lista com os identificadores de cada produto.

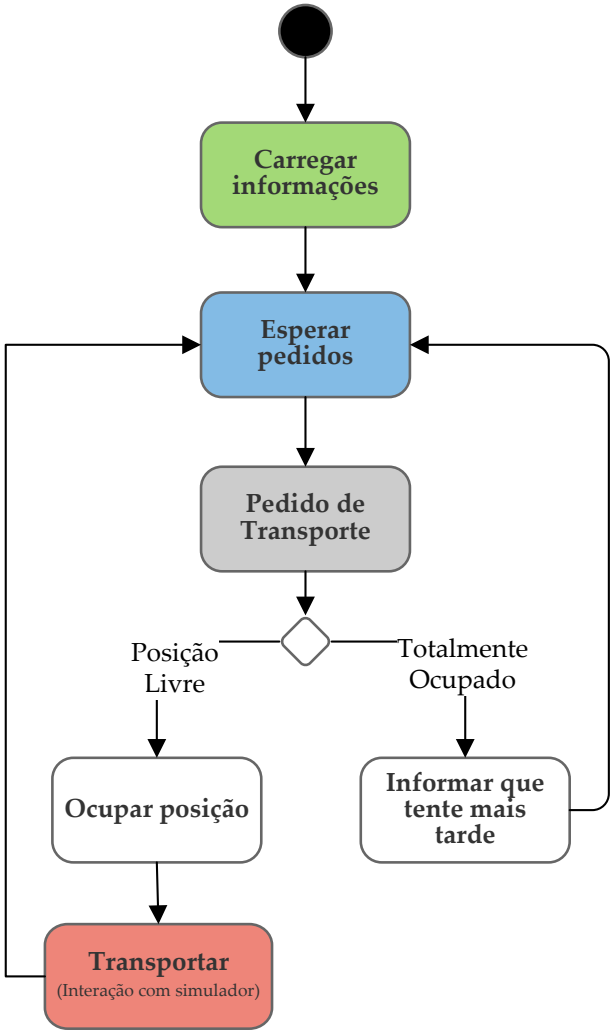

Figura 3.6 Diagrama de atividade da passadeira

3.5.2. Diverter Agent

Os pontos de encaminhamento são módulos essenciais ao bom funcionamento de sistemas de transporte, sendo responsáveis por desviar os produtos dos caminhos lineares das passadeiras. Os encaminhadores funcionam como cruzamentos de tráfego onde o melhor caminho é adotado, e é possível tornar eficaz todo o sistema de transporte dinâmico. Tal como acontece com as passadeiras, os pontos de encaminhamentos necessitam de ter um conhecimento sobre as suas ligações, e deste modo saber o melhor caminho para chegar a cada agente de execução.

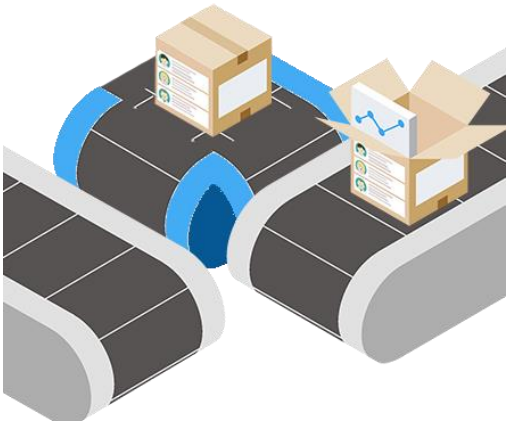

Figura 3.7 Representação dum ponto de encaminhamento

Com a grande disparidade de possibilidades nos encaminhadores e devido às suas capacidades adaptativas é necessário ter em conta uma variável até agora inexistente, variável esta que se prende com as configurações de receção ou envio dos produtos, ou seja, como os encaminhadores podem ser flexíveis e móveis, é necessário ter em conta posições e orientações no que diz respeito à receção e ao envio de produtos para outros módulos. Como exemplo na Figura 3.7, o ponto de encaminhamento, a azul, tem posições diferentes, quanto às interações entre as duas passadeiras. Para interagir com a passadeira do lado direito o ponto de encaminhamento precisa de estar numa posição mais alta, de modo a que a transição do produto entre os dois módulos seja a mais real e suave possível é necessário garantir que a posição é correta, tendo em conta a origem ou o destino do produto. Estas características devem ser controladas também pelo agente, permitindo assim abstrair o módulo físico de quaisquer perceções ou capacidades lógicas.

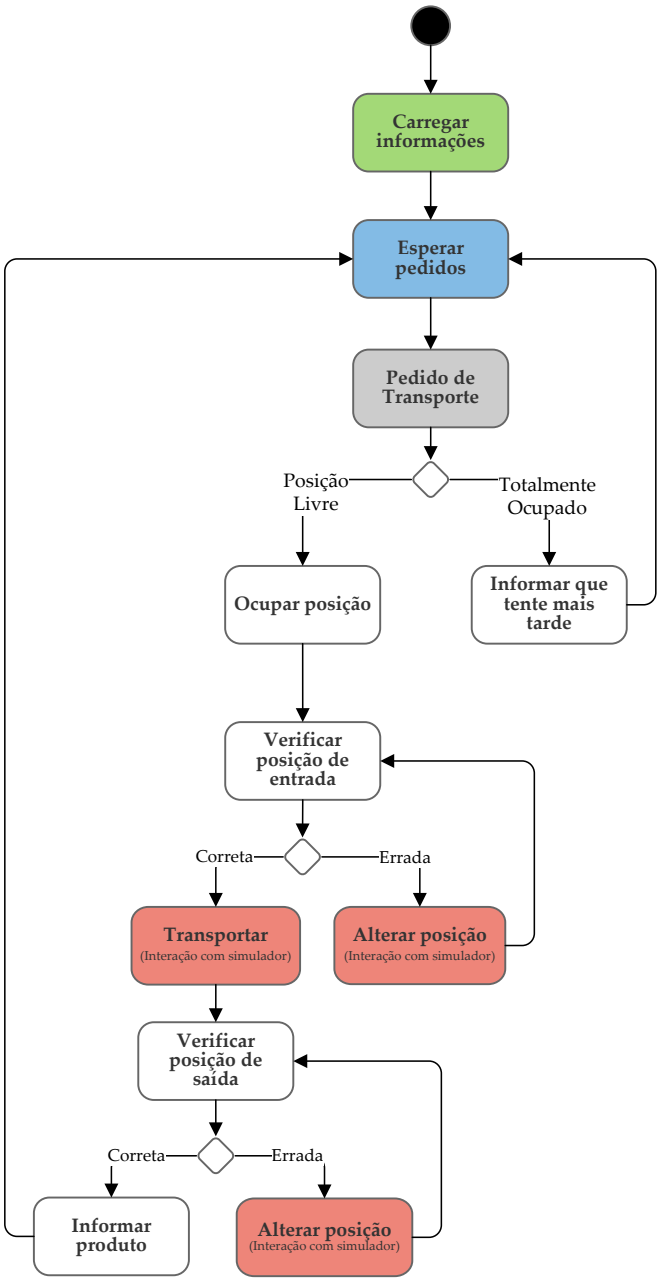

Figura 3.8 Diagrama de atividade do ponto de encaminhamento

3.5.3. Transport Agent

O mais abstrato e talvez o mais importante dos agentes na arquitetura de transporte é o transport agent. Este agente é responsável por coordenar o transporte do produto desde o módulo onde este se encontra até ao seu destino. De modo a simplificar o conhecimento do produto

sobre o sistema, surge a necessidade de abstrair deste todas as passadeiras e pontos de encaminhamento pelo qual tem de passar para chegar aos módulos de execução, assim sendo é inevitável passar essa responsabilidade a outra entidade, e é neste sentido que surge o transport agent.

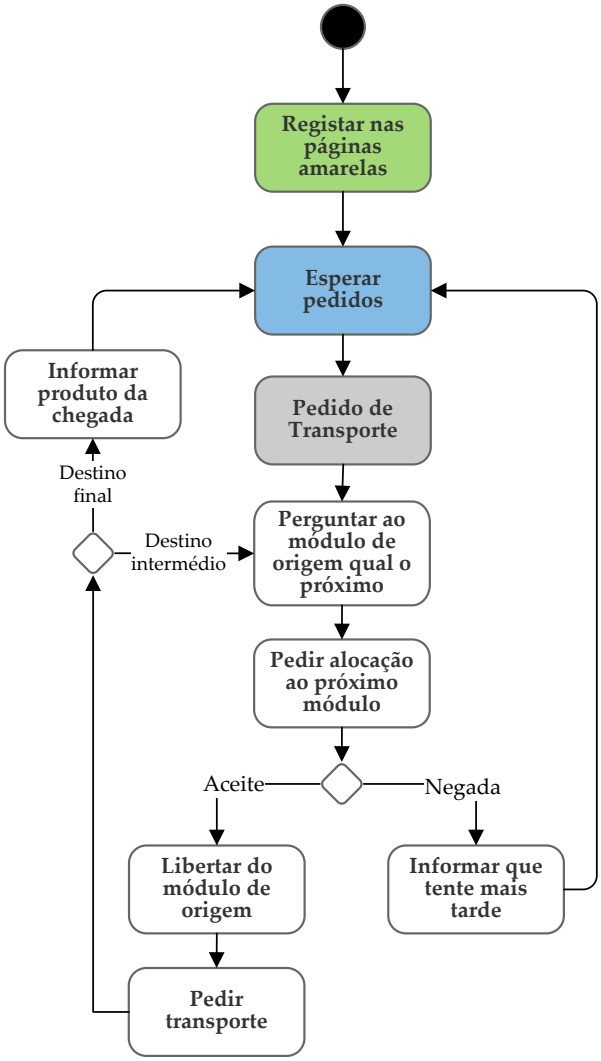

Figura 3.9 Diagrama de atividade do transporte

Sem conhecimento sobre as ligações do sistema, o transport agent pergunta ao módulo de origem qual o próximo módulo para chegar ao destino pretendido. Aquando da receção da resposta o transport agent fica responsável por pedir ao agente desse próximo módulo a alocação para o produto, para que este possa avançar no seu percurso até ao destino, e assim sucessivamente até que o produto chegue ao módulo pretendido. Na Figura 3.9, encontra-se um esquema do funcionamento pretendido para o transport agent.

Uma das características que diferencia este agente dos até agora descritos é a sua natureza puramente abstrata pois este agente não representa nenhum módulo físico, mas sim uma entidade integralmente virtual cujo objetivo é de controlo e integração do sistema.

3.6. Agentes de abstração

O objetivo dos agentes de abstração é representar entidades físicas que não interagem diretamente com o seu agente, ou seja, são agentes que apesar de não trocarem informação ou qualquer tipo de dados com os objetos que representam, são responsáveis pela sua abstração no sistema multiagente, permitindo um controlo lógico dos mesmos e maior inteligência do sistema.

3.6.1. Product Agent

Na arquitetura que está a ser desenvolvida apenas temos um agente deste tipo, o product agent. Como já dito acima, o agente e o produto não interagem diretamente, pois não há qualquer necessidade de trocar informação ou dados entre o produto e o seu agente. Assim sendo as funcionalidades do agente estão integralmente relacionadas com comportamentos lógicos que permitem ao produto organizar-se e completar as suas tarefas.

Para conseguir ter vários tipos de produtos numa linha de montagem precisamos de ter em conta as configurações e lista de tarefas de cada tipo, desse modo é indispensável ter uma fase inicial onde o produto carrega a lista de skills que pretende realizar de acordo com o seu tipo.

Depois de ter conhecimento das tarefas pretendidas, o comportamento do product agent rege-se por esta lista, seguindo passo a passo a execução de cada tarefa, como se de uma receita se tratasse. Quando termina todas as tarefas o agente é responsável por pedir que transportem o produto para a saída de modo a que este possa ser recolhido e saia do sistema.

3.7. Módulos

Os módulos são as entidades virtuais que representam as entidades físicas a simular. Para que o simulador seja fiável e que funcione como pretendido, os módulos devem restringir o seu funcionamento às funcionalidades básicas das máquinas que abstraem.

Na elaboração desta arquitetura estão a ser tidos em conta dois principais tipos de módulos, os módulos de execução, mais propriamente os módulos de estação, que representam os robôs ou os postos de trabalho, e os módulos de transporte, como é o caso da passadeira e dos pontos de encaminhamento. É consequência das diferentes funcionalidades destes módulos, o surgimento destes dois grupos, onde a intenção dos módulos de transporte apenas requer comportamentos relativos aos transporte e alocação do produto, o que não acontece com os módulos

de execução, onde se pretende simular tarefas e executar procedimentos sobre os produtos do sistema.

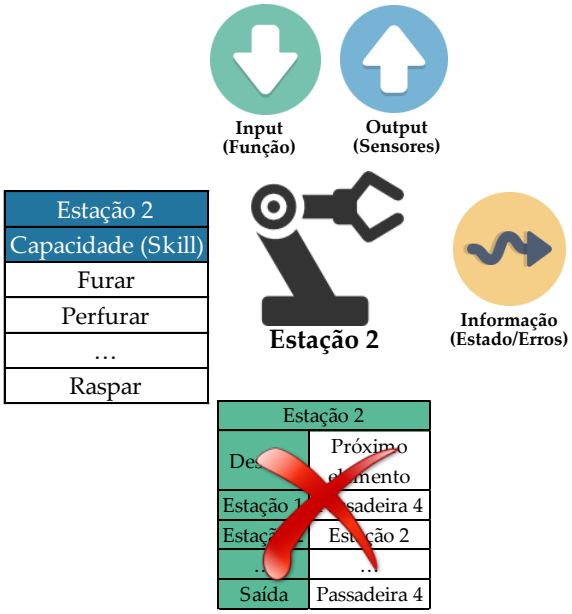

Figura 3.10 Esquema do módulo de estação

Neste sentido surgem dois diagramas ilustrativos, um para uma estação, Figura 3.10, e um para um módulo de transporte, Figura 3.11, neste caso uma passadeira, mas que se assemelha em muito aos pontos de encaminhamento.

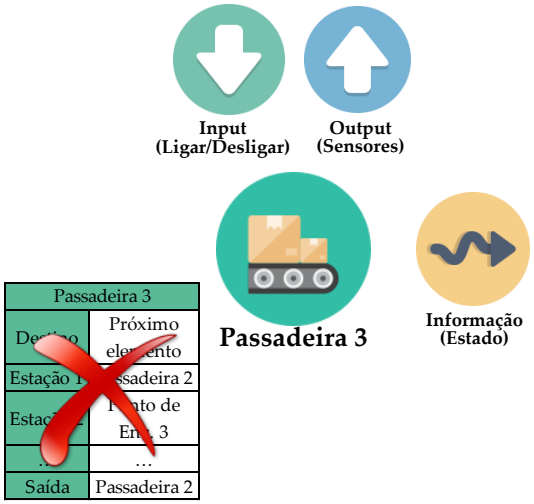

Figura 3.11 Esquema do módulo de passadeira

Como é percetível em ambos os diagramas, nenhum dos módulos deve ter acesso à sua tabela de encaminhamento, ou seja, todas as ligações entre módulos devem ser geridas pelo

agente correspondente, como explicado anteriormente. A maior diferença entre estes dois módulos é a tabela de skills existente nas estações. Esta lista de skills pretende simular um robô industrial, que consoante alguns parâmetros de entrada, consegue oferecer várias funções e assim disponibilizar ao sistema maior flexibilidade e dinâmica na distribuição de tarefas. Como os módulos de transporte não precisam de realizar várias funções diferentes, os seus comportamentos limitam-se a dois estados, ligado ou desligado. Outra das características que pode trazer benefícios às plataformas de simulação é a saída de informação durante a execução das funções, principalmente para fins estatísticos e de análise de performances é fundamental o acesso a informações paralelas.

É de notar a simplicidade que se pretende implementar com simulações modulares. Tal como no mundo real, uma linha de montagem é composta por várias secções, cada uma com as suas passadeiras, encaminhadores, máquinas de fabrico, entre outras peças importantes à indústria, mas todo este trabalho é feito sobre componentes isolados entre si e que funcionam de maneira independente. É sobre esta lógica modular que está elaborada esta arquitetura, permitindo que após uma simulação satisfatória seja de rápida e fácil transferência para o mundo físico, sendo precisos apenas pequenos ajustes devido a aproximações ou simplificação de processos utilizados.

Aprofundando agora a interação entre os módulos de transporte, é percetível na Figura 3.12, o modo de funcionamento convencional, representado pela seta a verde, onde os módulos comunicam diretamente através do envio de sinais, e o modo dinâmico que se pretende implementar, onde é necessária a interação entre vários agentes para o mesmo processo.

No modo convencional a passadeira onde o produto se encontra é responsável por enviar um sinal ao próximo módulo, neste caso um ponto de encaminhamento, para que este saiba que tem um produto pronto para o seu processamento. Este método é eficaz e rápido, mas não permite ao sistema dinâmica no transporte nem contorno de situações inesperadas, pois se os módulos estão ligados fisicamente e o sinal apenas pode sair para um ponto especifico, o sistema fica corrompido e preso à espera que esta via seja desimpedida.

No método que se pretende implementar, os módulos são abstraídos por um agente. Quando o produto chega ao final da passadeira é acionado o mesmo sensor que era no modo convencional, só que em vez de o sinal ser diretamente enviado para o próximo módulo, o agente da passadeira é informado que este está disponível para a próxima etapa. O agente da passadeira informa o transport agent que o produto está pronto, e logo este tende a procurar o próximo módulo no registo de páginas amarelas. Aquando encontrado são iniciadas negociações entre o transport agent e o diverter agent, de modo a cumprir a transferência do produto dum módulo para o outro. Após a conclusão de todos os protocolos é dada ordem para que o produto seja transferido, e termina assim o transporte entre dois módulos. Nem só de vantagens é feito este método, trazendo algum atraso ao sistema e alguma complexidade na troca de mensagens, no entanto este modo de operar trás grande flexibilidade no transporte, permitindo evitar falhas ou pontos obstruídos do sistema.

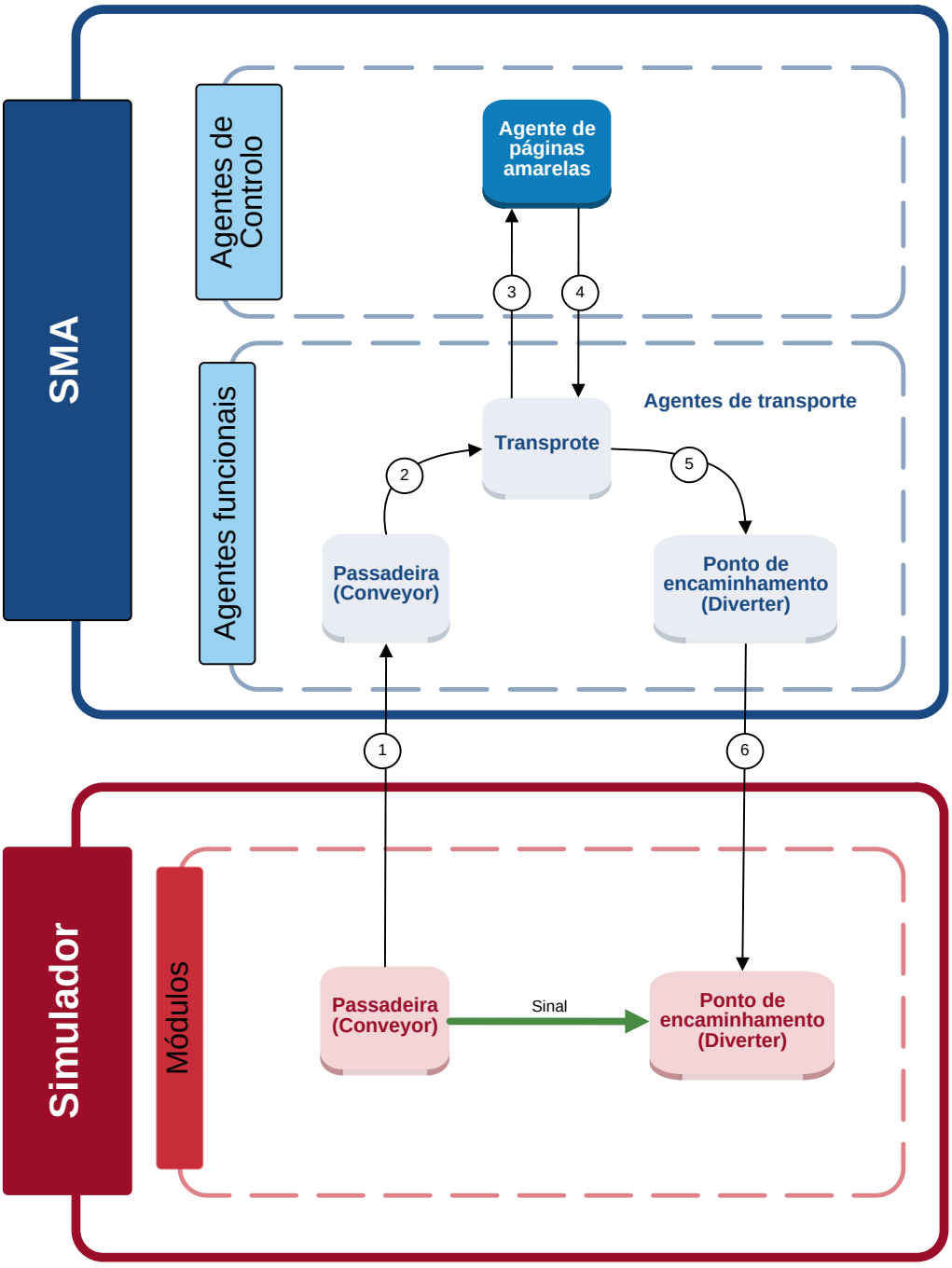

Figura 3.12 Comunicação entre módulos

3.8. Product Agent

Devido à importância deste elemento na arquitetura, segue-se uma descrição detalhada dos comportamentos que se pretendem do produto. O produto, neste caso, quer-se uma das

peças centrais da arquitetura, sendo a linha desenhada de acordo com as especificações e necessidades de produção. Em arquiteturas dinâmicas surge a possibilidade de atribuir ao elemento capacidades organizacionais e de comunicação para que este possa adaptar da melhor forma, tendo em conta o ambiente envolvente, a sua produção de maneira a aumentar a eficácia do sistema.

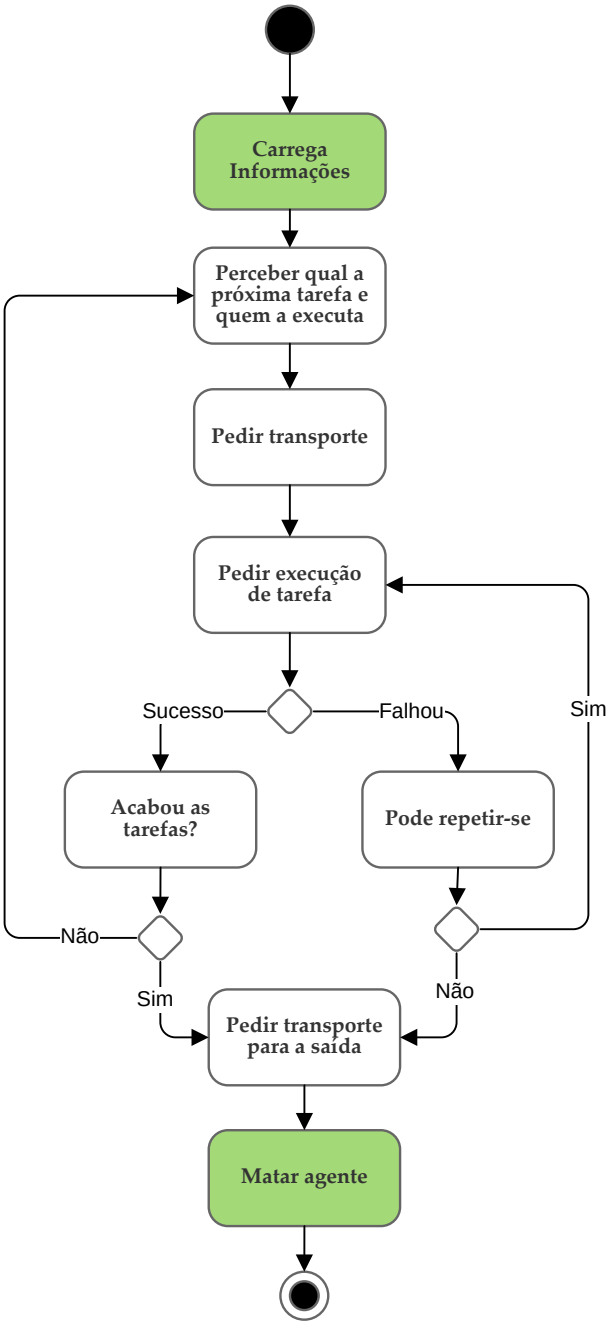

Figura 3.13 Diagrama de atividade do produto

O produto pretende-se representado no simulador como um elemento independente, que possa ser alterado e movido durante a simulação. Deste modo a maneira mais eficaz de o representar é limitar o seu contexto ao aspeto visual.

Por outro lado, o controlo das interações deve estar presente no agente que abstrai o produto. Com a finalidade de controlar a produção do produto que representa, o product agent é responsável por gerir e organizar todas as tarefas para a conclusão do produto.

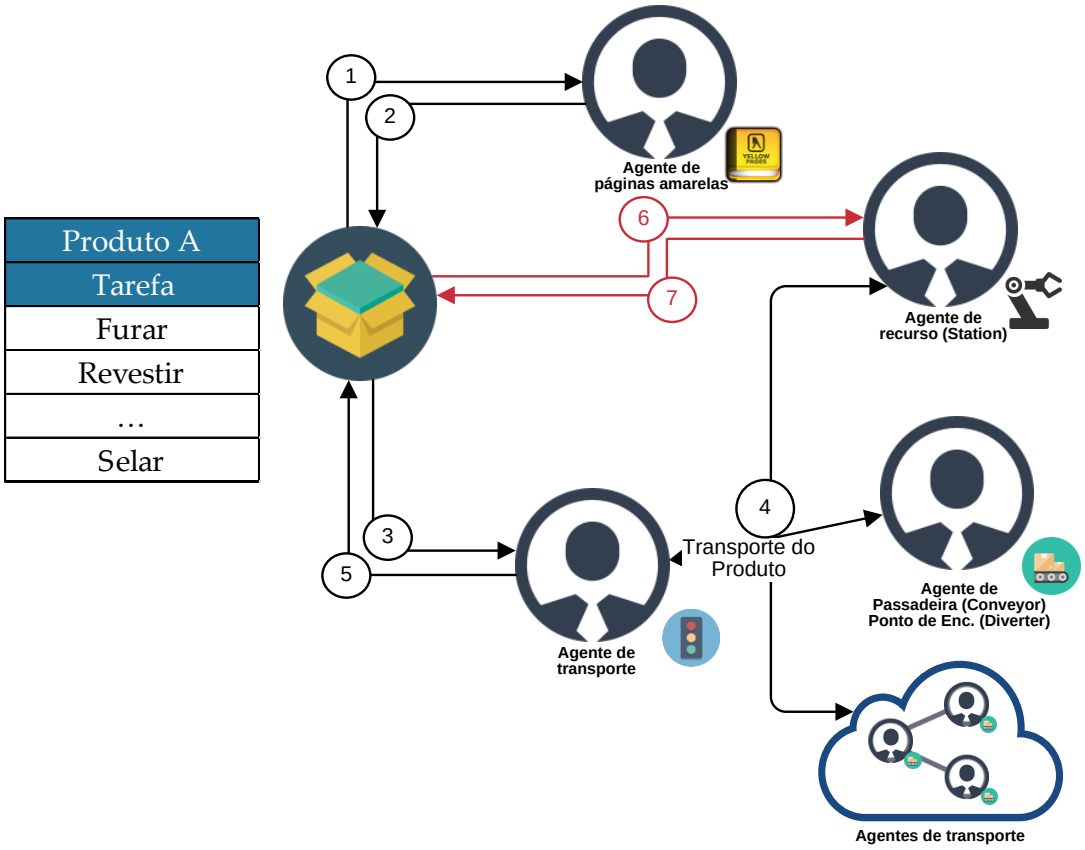

Figura 3.14 Esquema de comunicações do produto

Como é percetível através da Figura 3.13, o comportamento que se pretende para o produto é composto por várias fases. Aquando a criação ou lançamento de um novo produto no sistema, surge com ele um novo agente, um product agent. Este agente é responsável por saber que tipo de produto abstrai e quais as tarefas que este necessita para ficar completo. Assim sendo existe uma primeira fase onde o agente carrega as informações referentes ao tipo de produto e a organiza em listas. Estas informações relacionam-se com personalização e tarefas a completar para finalizar o produto. Após a perceção das tarefas envolvidas, o agente procura, com base na tarefa pretendida, quem pode disponibilizar a tarefa. Na Figura 3.14, onde estão esquematizadas as fases de comunicação do produto, pode observar-se a fase de pesquisa através das mensagens 1 e 2. Com base nas duas imagens já referidas, segue-se uma fase de transporte, onde o product agent pede ao transport agent que o desloque até ao módulo de execução.

Por último, quando o produto chega ao módulo de execução, segue-se a última etapa comportamental do product agent, o pedido de execução. Com a mensagem 6 e 7, são trocadas informações que culminam na execução da tarefa requerida.

Com o culminar da execução, o produto continua o seu caminho, sendo este em direção à próxima tarefa, pelo que tem de repetir todo o processo descrito, ou para a saída, caso já tenha terminado a sua lista de tarefas.

3.9. Transporte

De modo a garantir um transporte eficaz e sem falhas, será necessário criar um protocolo de comunicação, envolvendo as várias entidades relacionadas com o transporte do produto, para que tudo funcione adequadamente. O protocolo envolve interações entre o transport agent e o agente que representa o produto a ser transportado, o agente do módulo de origem e o agente do módulo de destino.

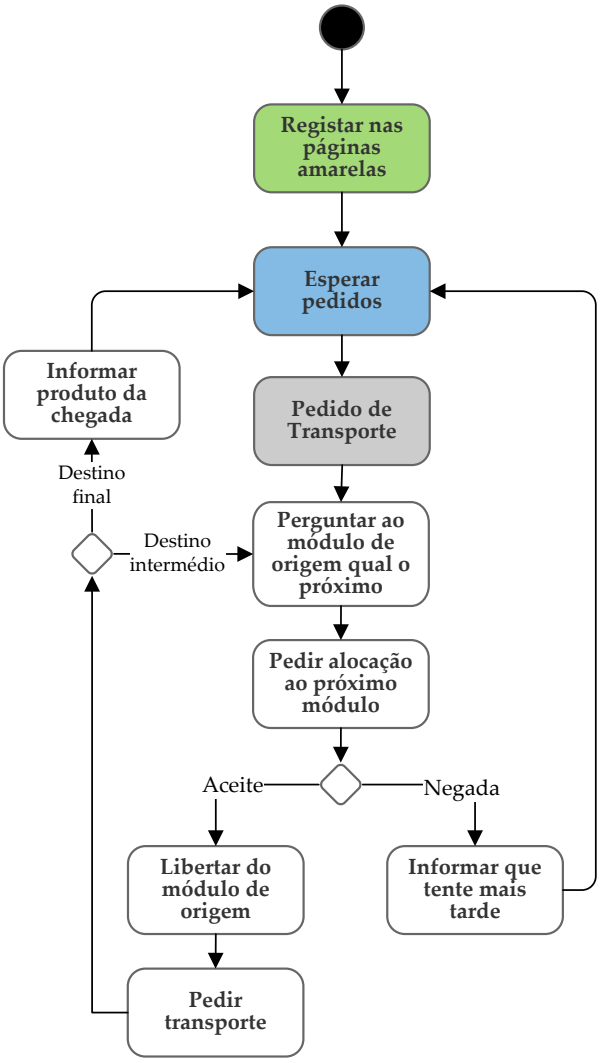

Figura 3.15 Diagrama de transporte

Na Figura 3.15, está representado um diagrama de atividades para esclarecer, de forma mais intuitiva, as várias fases do transporte do produto. Numa primeira fase, quando o produto

pede ao transport agent que inicie o seu deslocamento, surge a fase de pesquisa pelo próximo módulo. O produto deve ser responsável por informar o transport agent sobre o módulo de origem, o destino e o seu próprio identificador. O comportamento que se pretende do transport agent é que este pergunte ao módulo de origem, qual o próximo módulo para onde se deve deslocar o produto. Através da implementação das tabelas de encaminhamento, o módulo de origem responde com o próximo módulo do caminho mais curto para chegar ao destino.

Após obter a resposta sobre o próximo módulo que deve contatar, inicia-se a fase de alocação do produto no novo módulo. É nesta fase que se avalia a disponibilidade do próximo módulo e tenta guardar uma posição que será ocupada posteriormente pelo produto.

Se as duas fases anteriores forem concluídas com sucesso, o transport agent pede ao módulo de origem que adote as configurações de saída para interagir com o próximo módulo. Esta fase é fundamental para o bom funcionamento, devido às várias configurações entre módulos que podem existir, tanto a nível de orientação como de posição de transmissão.

Por último, surge a fase de transporte, onde o produto será efetivamente transportado para o módulo seguinte, atualizando a sua posição.

Para um correto funcionamento do protocolo é essencial que todas as etapas sejam cumpridas de maneira rigorosa, pois só assim se conseguem garantir que todas as interações e ligações entre os módulos são válidas e estão a ser utilizadas da maneira correta.

Um último requisito é a necessidade da utilização de uma lista onde estão presentes todos os produtos que se encontram no sistema. Nesta lista devem constar os identificadores dos produtos e as fases em que se encontram do transporte, de maneira a garantir que o processo de transporte é sequencial.

4

4. Implementação

Neste capítulo é descrita a implementação da arquitetura detalhada no capítulo anterior. Numa primeira fase surge uma breve descrição das tecnologias utilizadas na implementação, seguindo-se uma minuciosa descrição do trabalho efetuado e das decisões tomadas ao longo da implementação da arquitetura.

4.1. Tecnologias de suporte

Para que a implementação sugerida no capítulo anterior seja possível são necessárias várias tecnologias. Entre elas, a linguagem de programação Java, descrita em (Gosling & Mcgilton, 1996), que foi utilizada como linguagem base de toda a estrutura pois tanto a plataforma multiagente utilizada como o simulador, são baseados nesta linguagem. Seguem-se breves descrições relativas às tecnologias utilizadas, de modo a contextualizar a implementação da arquitetura.

4.1.1. Netbeans IDE

O ambiente de programação utilizado foi o NetBeans Integrated Development Environment, (Boudreau, Glick, Greene, Spurlin, & Woehr, 2002). Este programa foi escolhido pela sua robustez e versatilidade na programação utilizando a linguagem Java. Apesar do seu grande leque de funcionalidades e ferramentas, o software é gratuito e permite grande flexibilidade nas opções que se podem tomar. Por ser bastante utilizado, existe uma grande comunidade que interage partilhando os seus problemas e as suas soluções, o que torna fácil resolver quaisquer adversidades que se interponham durante o desenvolvimento.

4.1.2. JADE

Java Agent Development Framework (JADE) é uma plataforma *open source* cujo objetivo é suportar modelos que tenham por base a programação de agentes. Implementado com recurso à linguagem de programação Java, o JADE permite o desenvolvimento de aplicações de elevada versatilidade e interoperabilidade garantindo todas as vantagens que a programação distribuída possa trazer, (Bellifemine, Poggi, & Rimassa, 2001).

A plataforma garante a coordenação entre os vários agentes através das especificações impostas pela **FIPA**, Foundation for *Intelligent Physical Agents*, também conhecidos como **FIPA-ACL**, Linguagem de Comunicação de Agentes, do inglês *Agent Communication Language*, descritas em ("FIPA ACL Message Structure Specification," 2016).

A execução de múltiplas tarefas, sejam paralelas ou simultâneas, são suportadas pela plataforma, pois esta modela os comportamentos dos agentes segundo o modelo de *behaviours*, em português comportamentos, caracterizados pelo agendamento não preemptivo.

4.2. Implementação do trabalho proposto

Para o desenvolvimento do trabalho foram necessárias várias etapas. A primeira etapa prende-se com a adaptação à ferramenta que implementa o sistema multiagente. Esta aprendizagem deveu-se à consulta da vasta documentação online no site que disponibiliza a plataforma. Esta documentação é composta por documentos explicativos e vários exemplos, permitindo uma fácil aprendizagem e rápida adaptação ao ambiente multiagente. O presente curso também oferece uma introdução a esta plataforma, introduzindo os conceitos de agente e seus comportamentos, bem como o funcionamento da plataforma JADE, as suas capacidades e as vantagens da sua utilização.

A segunda etapa, a familiarização com o funcionamento do simulador e com os seus mecanismos de representação, caraterização e descrição dos módulos. Para facilitar este processo foram disponibilizados pela empresa fornecedora do simulador, a TTS, (Technology Transfer System S.r.l., 2016), alguns exemplos que retratam linhas de produção simples.

Por último e antes do desenvolvimento do projeto final, foi idealizado e implementado um pequeno teste, utilizado como prova de conceito, integrando o simulador e uma arquitetura composta por cinco agentes, três conveyor agent, um source agent e um sink agent, para provar que a interoperabilidade era alcançável e que era possível alcançar o objetivo a que este trabalho se propõe.

4.2.1. Funcionamento do simulador

O simulador utilizado é denominado de DDDSimulator e pertence à empresa TTS sediada em Milão. A TTS, do inglês Technology Transfer System S.r.l., fundada em 1993 caracteriza-se pela investigação e desenvolvimento de tecnologias de fabrico inovadoras e de sistemas de automação. Devido ao trabalho desenvolvido, tem vindo a tornar-se umas das empresas de referência no que diz respeito ao desenvolvimento de soluções tecnológicas e de simulação adaptadas às necessidades dos seus clientes. Outra das características desta empresa, prende-se com a constante atenção dada a atividades de investigação internacionais junto dos principais protagonistas europeus e mundiais de investigação e inovação (Technology Transfer System S.r.l., 2016).

Uma das soluções, como já referido, é o simulador DDDSimulator, um simulador com animações tridimensionais, 3D, baseada em eventos discretos e realidade virtual. Estas duas camadas encontram-se separadas, ou seja, o ambiente de visualização tem um funcionamento distinto da camada lógica, permitindo que os módulos lógicos não estejam acoplados a nenhum recurso 3D, mas consigam escrever e ler o estado de qualquer entidade da simulação, precisando apenas do seu identificador no sistema.

Por sua vez, o controlo lógico da simulação é baseado em eventos discretos. Estes eventos são registados numa escala temporal que utilizando um relógio global permite garantir a ordem dos eventos. É também com esta escala temporal e o seu relógio de sincronização que a camada lógica coordena o ambiente de visualização, onde se integram os modelos representativos.

Uma das vantagens da utilização deste simulador prende-se com a sua característica modular. Os controladores lógicos têm um caráter independente entre si, o que permite uma fácil configuração de sistemas, adicionando e removendo componentes conforme necessário.

Outra das principais vantagens deste simulador é a sua escala temporal e a programação de eventos. Com uma programação de eventos numa escala separada de todos os módulos, é possível registar, os mesmos, em tempo real, ou seja, não é preciso acabar o evento que está a decorrer para programar o próximo. Esta característica trás também a possibilidade de ter eventos a correr em paralelo, o que se revela um grande benefício para a integração nesta tese proposta. Deste modo a simulação termina quando não houverem eventos agendados e forem processados todos os anteriormente registados.

A discrição do ambiente de simulação é feita através de um ficheiro de XML. O simulador utiliza um ficheiro para descrever os modelos, as instâncias e as conexões entre os vários módulos. Na Figura 4.1 podemos encontrar a base de um ficheiro utilizado pelo simulador.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<model xmlns="http://heron.ttsnetwork.com/xml/v1" name="heron-model">
  <templates>
  </templates>

  <variables>
  </variables>

  <modules>
  </modules>

  <connections>
  </connections>
</model>
```

Figura 4.1 Modelo XML base para descrever o ambiente de simulação

Como podemos observar o ficheiro de descrição do ambiente de simulação é composto por quatro grupos. A etiqueta denominada *templates*, em português modelos, serve para descrever os modelos utilizados na simulação. Para uma melhor explicação, consideremos um sistema composto por três módulos, uma entrada de produtos e duas passadeiras. Neste caso iriam existir dois modelos diferentes, o modelo da passadeira e o modelo do módulo de entrada, cada um descreveria as variáveis de entrada, os portos, para receber e enviar informação e ainda outros parâmetros que podem diferenciar cada umas das instâncias, como é o caso de querer que as passadeiras funcionem a velocidades diferentes.

Na Figura 4.2, podemos observar um desses exemplos. É de salientar que é nesta fase que se relaciona o modelo com a classe em java que irá mapear o comportamento do módulo. É possível observar no cabeçalho do modelo vários campos, entre eles o *name*, responsável por atribuir um nome ao modelo, e um campo *type* onde é introduzido o nome da classe que implementa o código comportamental.

Um modelo é caracterizado por quatro campos, o *tag*, o *documentation*, o *ports* e o *params*, sendo estas duas últimas as mais importantes para a descrição do modelo. As duas primeiras etiquetas prendem-se com descrições informativas dos modelos, enquanto que as outras duas etiquetas descrevem os portos de entrada e saída, e as variáveis ou parâmetros de entrada, respetivamente.

Ainda na Figura 4.2, é mostrado um modelo de passadeira que tem um porto de entrada e outro de saída, os dois do tipo de *String*, descritos entre as etiquetas *ports*. Na etiqueta *params* são descritas as variáveis de entrada do modelo de passadeira. Neste caso temos três parâmetros de entrada, a velocidade de funcionamento da passadeira, e duas variáveis que vão representar a posição de entrada na passadeira e a posição de saída. A velocidade, do tipo double,

```
<template name="Passadeira" type="ClassePassadeira" version="1" icon="" author="PBarroca">
   <tag>passadeira, examplo</tag>
   <documentation>@author Pedro Grilo Barroca</documentation>
   <ports>
      <port name="in" type="java.lang.String" oneToMany="false" position="LEFT" kind="PORT" direction="IN"/>
      <port name="out" type="java.lang.String" oneToMany="false" position="RIGHT" kind="PORT" direction="OUT"/>
   </ports>
   <params>
      <param name="velocidade" type="double" level="BASIC">
         <message></message>
      </param>
      <param name="posicaoInicial" type="string" level="BASIC">
         <message></message>
      </param>
      <param name="posicaoFinal" type="string" level="BASIC">
         <message></message>
      </param>
   </params>
</template>
```

Figura 4.2 Modelo XML de exemplo duma passadeira

vai definir a que velocidade o produto se desloca na passadeira. As variáveis de entrada referentes aos referenciais são do tipo *String*, pois como já dito anteriormente, os módulos conseguem aceder a qualquer componente da simulação, e para isso apenas precisam do seu identificador. É o caso dos referenciais, no caso desta passadeira precisamos de dois referenciais, um para marcar a posição de entrada de produtos e outro para a saída dos mesmos da passadeira. O deslocamento do produto vai ser calculado utilizando a distância entre estes dois referenciais, e a velocidade correspondente da passadeira.

Definido o modelo a utilizar na passadeira, é necessário instanciar a quantidade pretendida. Na Figura 4.3 está representado um exemplo de instância utilizando o modelo descrito na Figura 4.2. Neste caso é descrita uma passadeira. A componente *visual* serve para definir a posição numa janela que representa o esquema total do sistema e os portos servem para localizar as entradas e saídas dos módulos, e correspondentes nomes, na figura representativa no esquema, exemplo na Figura 4.4. Como era de esperar, é na fase de instanciação que são atribuídos valores aos parâmetros de entrada dos modelos. No caso apresentado a velocidade de funcionamento da passadeira quere-se que seja 2 $m.s^{-1}$. No caso das posições iniciais e finais, são passados os nomes dos referenciais definidos no ambiente de simulação.

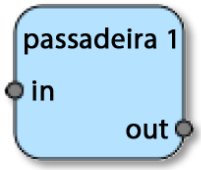

Figura 4.4 Representação de módulo

```
<module-instance name="Passadeira1" type="ClassePassadeira">
  <visual x="283" y="138">
    <port-position port="in" position="LEFT" index="0"/>
    <port-position port="out" position="RIGHT" index="1"/>
  </visual>
  <param name="velocidade">2</param>
  <param name="posicaoInicial">passadeira1.start</param>
  <param name="posicaoFinal">passadeira1.fim</param>
</module-instance>
```

Figura 4.3 Modelo XML de exemplo dum módulo

A próxima etapa relaciona-se com a classe em java que pretende implementar o comportamento do módulo. Esta vai estender as funcionalidades da classe *Module* do simulador. Na Figura 4.5 temos um exemplo de classe em java que implementa o exemplo da passadeira acima descrito. É de notar que o nome da classe tem de corresponder ao *type* descrito nos modelos e instâncias anteriores, de modo a que o simulador relacione as duas componentes. Para o bom funcionamento desta é necessário ter em conta também os parâmetros de entrada. É possível observar, como já se esperava, que o módulo tem três parâmetros de entrada, a velocidade, que caso não receba valor foi definida como 4, a posição inicial e a posição final, ambas sem valor predefinido. O último pormenor indispensável desta classe é a função *init()*, esta função é a função que é chamada quando o módulo é instanciado, assim sendo, é nesta função que será descrito o comportamento que o módulo deve adotar.

```
package my.company;

import … 5 lines

/**** @author Pedro Grilo Barroca*/
@LogicsModule(author = "PBarroca", version = 1, tags = "passadeira, examplo")
public class ClassePassadeira extends Module {

   @Parameter
   public double velocidade = 4;
   @Parameter
   public String posicaoInicial;
   @Parameter
   public String posicaoFinal;

   @Override
   protected void init() {
            //adicionar aqui o comportamento pretendido
   }

}
```

Figura 4.5 Classe comportamental de módulo

O último passo na definição de um módulo, relaciona-se com a ligação ao modelo tridimensional que irá representar o mesmo, e a definição dos referenciais necessários ao seu bom funcionamento.

O sistema de simulação, como já dito, é definido através de um ficheiro de **XML**, onde estão descritas todas as informações relativas aos módulos utilizados, ou seja, as orientações, as posições, as escalas entre outras características preponderantes. De modo a facilitar a utilização, o Netbeans IDE, disponibiliza uma ferramenta, o *navigator*, que permite editar o ficheiro mais intuitivamente.

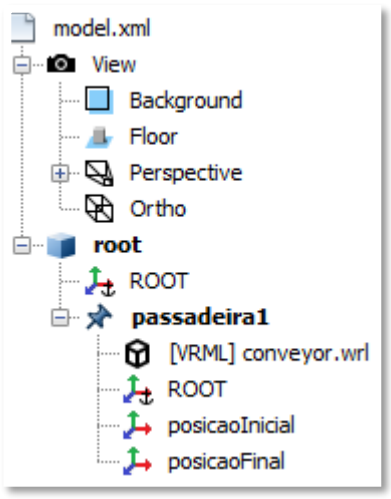

Figura 4.7 Representação do navigator do simulador

A Figura 4.7 representa uma vista do *navigator*. Como é possível observar, temos, na seção *view* que se destina ao aspeto geral do sistema, várias definições configuráveis, tais como vistas, perspetivas, a cor de fundo, entre outras. Por outro lado, na seção *root* é onde estão as referencias aos módulos do ambiente de simulação. É composto por um referencial principal, que serve de ancora para todos os objetos, o *ROOT*, seguido de todos os objetos presentes no sistema. No exemplo dado, é possível observar que o sistema é composto por uma passadeira, com o nome “passadeira1”, que por sua vez tem quatro elementos a si associados. Desses quatro elementos, três são referenciais e um outro que define o aspeto visual do módulo, neste caso foi utilizado um modelo tridimensional de uma passadeira fornecido pela empresa, como podemos observar na Figura 4.6. Dos três referenciais apresentados, um deles serve de referência à

Figura 4.6 Modelo de passadeira fornecido pela TTS

estrutura gráfica, ou seja, tal como o ambiente de simulação tem um referencial principal, o modelo tem um ponto que marca o zero absoluto para todos os outros referenciais, que se chama de *ROOT*. Os outros dois referenciais presentes na imagem, são definidos para integrarem o comportamento do módulo, pois como já explicado anteriormente o simulador funciona através do deslocamento entre referenciais.

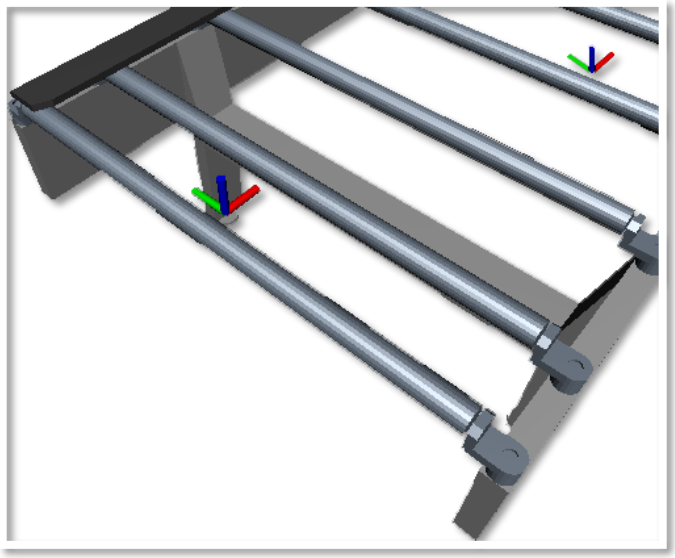

Figura 4.8 Exemplo de referenciais aplicados a módulo

Na Figura 4.8 podemos observar o exemplo implementado, onde é possível ver o referencial “posicaoInicial” que marca o inicio da passadeira, e o referencial “ROOT” que indica o zero absoluto, neste caso no centro, da passadeira.

Por fim, o Netbeans IDE oferece mais uma ferramenta, que nos permite editar as opções dos objetos criados. Como é visível na Figura 4.9, o objeto tem várias definições configuráveis, tais como o nome, a descrição, a geometria, ou *geometry*, onde se identifica o modelo tridimensional que irá representar o objeto, e a unidade, ou *length unit*, a que devem ser aplicadas as

Properties	
Visible	☑
Name	Passadeira1 …
Description	…
Geometry	conveyor.wrl …
Length Unit	mm
Local position	
Location	0.00 0.00 0.00 …
Orientation	0.00 0.00 0.00 …
Absolute position	
Location	0.00 0.00 0.00 …
Orientation	0.00 0.00 0.00 …
Joint	
Location	0.00 0.00 0.00 …
Orientation	0.00 0.00 0.00 …
Type	RIGID
Custom Properties	
*	…

Figura 4.9 Visualizador de opções dos objetos

medidas retiradas desse modelo. Os restantes campos mostrados na imagem, referem-se à posição e orientação do objeto, mas como este se encontra assente no zero absoluto do sistema todos os valores estão a zero.

Para melhor explicar a posição global e a posição absoluta, surge o exemplo das opções do referencial "posicaoInicial". Na Figura 4.10 podemos observar que a posição de transformação, referente às coordenadas da ancora do modelo tridimensional, é igual à absoluta, que indica a posição de acordo com o referencial principal do sistema, pois ambas são coincidentes.

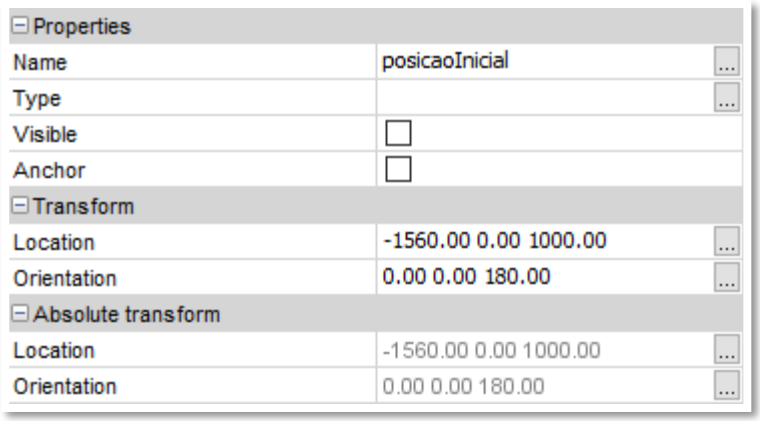

Figura 4.10 Exemplo de opções de referencial

Após explicadas todas as etapas de definição de um modelo, é possível rever o modo de ligação entre diferentes módulos. Na Figura 4.11 é possível observar a ligação entre dois módulos, a passadeira1 e a passadeira2. Estes dois módulos foram definidos tendo em conta todos os passos referidos acima.

```
<connection>
    <port-out module="passadeira1" port="out"/>
    <port-in module="passadeira2" port="in"/>
</connection>
```

Figura 4.11 Exemplo de conexão de dois módulos

As conexões entre os módulos são definidas entre as etiquetas *connections*, do ficheiro **XML** que descreve o sistema, tal como exemplificado na Figura 4.1. Esta conexão serve para os módulos comunicarem entre si como se do envio de um sinal elétrico se tratasse. Esta característica não foi utilizada neste trabalho, pelo que não será explicada aprofundadamente. No entanto, na Figura 4.12 está uma representação da ligação acima descrita.

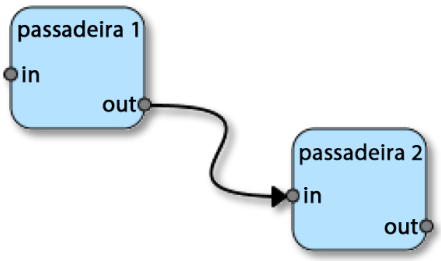

Figura 4.12 Representação de conexão entre módulos

4.2.1. Modo de interação

A interação entre os módulos do simulador e os agentes do MAS é feita através de referência direta, ou seja, quando o módulo é criado, este envia, através duma classe de controlo, um pedido ao agente de lançamento para criar um agente do seu tipo e com as suas características. O agente que lança a nova instância responde com uma referência à mesma, que é guardada no módulo e que fica acessível como de uma qualquer variável se tratasse. Por outro lado, o agente, quando é criado, recebe a instância do módulo, que guarda, ficando dessa maneira acessível. É dessa maneira que o agente e o módulo interagem, permitindo que a troca de informação seja feita através de funções implementadas dos dois lados.

Outra das características que foi necessária adotar nos módulos, visto que a simulação é baseada em programação de eventos, deve-se à maneira de saber quando o movimento é concluído, pois não era conveniente começar qualquer um dos protocolos de transporte sem ter certezas de que o produto estava disponível para o próximo movimento. Assim sendo, e explorando mais aprofundadamente as funcionalidades do simulador, é possível agendar uma função do tipo *runnable*, função esta que é programada em serie com a translação, o que garante que a mesma apenas irá correr quando a translação terminar.

4.2.2. Agentes

Como descrito no capítulo anterior, foram utilizados nove tipos de agente diferentes. O source agent, que tem como função representar a entrada de produtos na linha. O sink agent, que como o nome indica, abstrai o módulo de saída da linha de produção. O station agent, este pretende representar os postos de trabalho, sejam estes ocupados por robôs ou por humanos. O conveyor agent, que representa os módulos de passadeira, módulos estes de comprimento fixo, mas variável entre as várias instâncias, que permitem alocar e transportar vários produtos de uma só vez. O diverter agent, o qual é responsável pelas transições entre as passadeiras, simulando cruzamentos, que permitem escolher o caminho mais rápido até ao destino. O transport agent, responsável por encapsular todo o processo de transporte, permitindo que este seja feito com apenas uma mensagem por parte do produto com o destino final e a origem do movimento. O product agent, que abstrai o produto e o seu controlo lógico. E por fim os dois agentes que sobram, o de lançamento e o de páginas amarelas, que servem maioritariamente para ga-

rantir o bom funcionamento da plataforma. Permitindo, o primeiro, lançar todos os agentes funcionais na plataforma em causa. Como agente de páginas amarelas foi utilizado um serviço implementado pelo JADE, o DF, do inglês Directory Facilitator, especificado pela FIPA, cujo objetivo é guardar uma lista com todos os agentes presentes na plataforma e os serviços por estes disponibilizados de modo a que seja fácil a pesquisa e obtenção dos identificadores para futuras comunicações.

Seguem-se descrições mais pormenorizadas sobre os diferentes tipos de agentes, contextualizadas com diagramas de classe e alguns diagramas de sequência para representar as interações entre agentes.

4.2.2.1. Protocolo de Transporte

Para que o transporte fosse estável entre dois módulos, foi criado um protocolo de transporte composto por quatro fases, esquematizado na Figura 4.14.

Como é possível observar através da Figura 4.14 o transporte divide-se nas fases, de perceber qual o próximo módulo, de alocação do produto, de libertação da posição anterior e de transporte do produto. Todas as mensagens trocadas contêm o identificador do produto para que o transport agent possa ter vários processos de transporte em paralelo, caso contrário existiria confusão na sequência das mensagens.

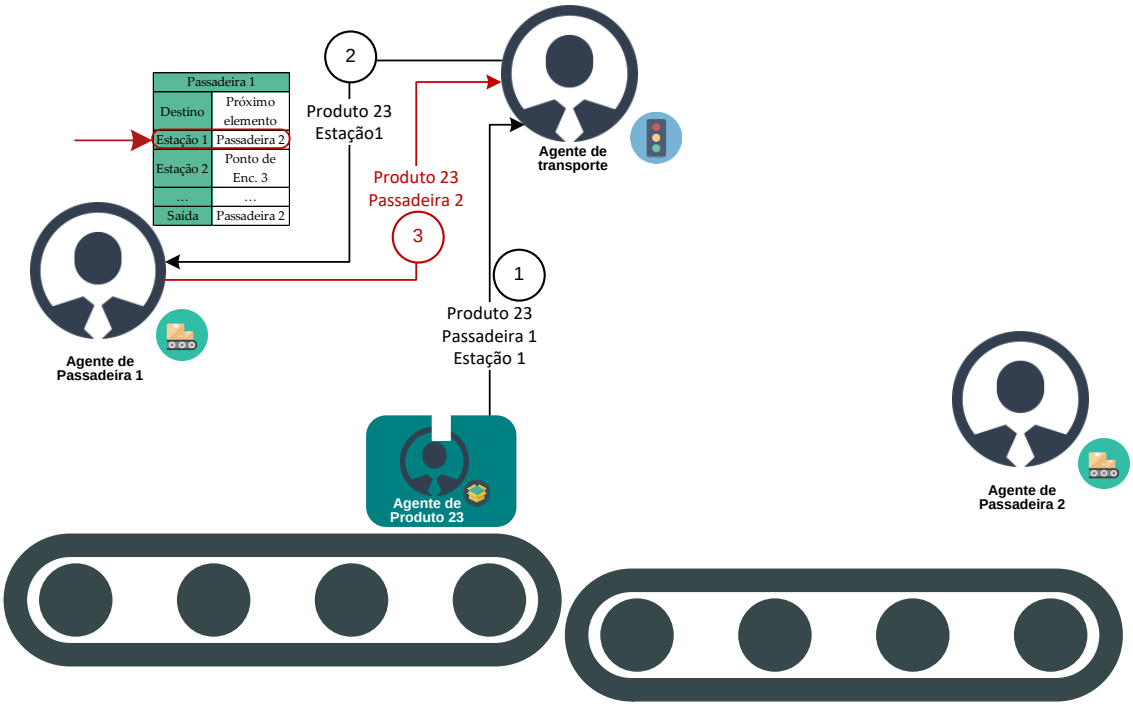

Figura 4.13 Representação da fase de próximo módulo

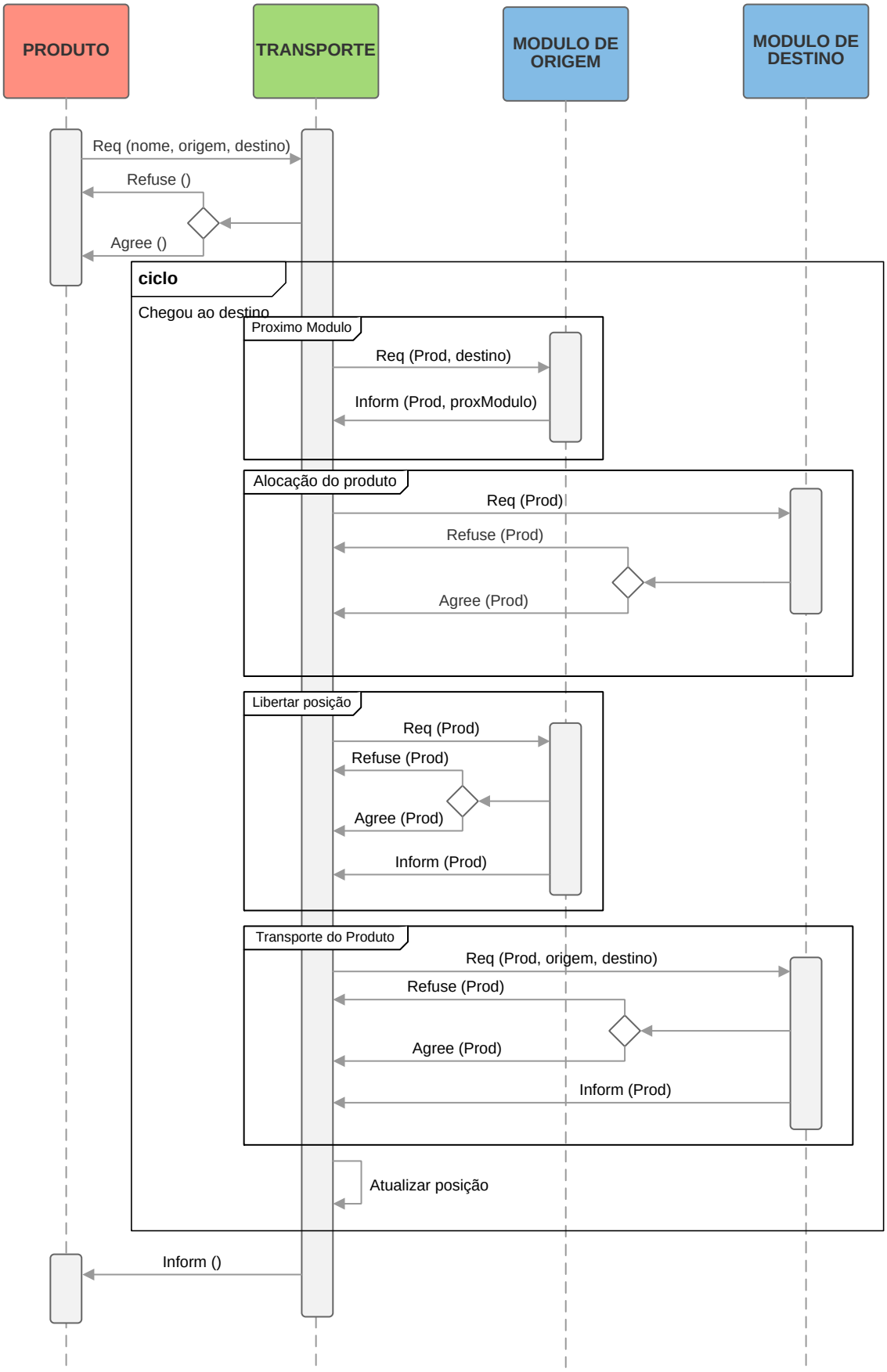

Figura 4.14 Diagrama de Sequência durante a execução do transporte

O protocolo inicia-se através de um pedido proveniente do product agent para transport agent. Neste pedido o produto envia, como informações, o seu identificador, o identificador do módulo onde se encontra e o identificador do módulo de destino. Após receber esta informação o transport agent verifica se é possível chegar ao módulo pretendido, se for o caso, este inicia a primeira das fases do protocolo de transporte, a perceção do próximo módulo, Figura 4.13. É enviada uma mensagem ao módulo de origem contendo o produto que pretende o transporte e o destino pretendido, de modo a que este consulte a sua tabela de encaminhamento e responda, através de uma mensagem do tipo *inform*, com o identificador do produto e o identificador do próximo módulo a ser contatado para chegar, da melhor maneira, ao destino. Caso o próximo módulo seja o mesmo que enviou a mensagem, ou seja, caso os identificadores do remetente e de próximo módulo sejam iguais, é enviado uma mensagem ao produto, do tipo *inform*, informando-o que o seu transporte foi concluído e que já se encontra no módulo pretendido. No caso de o próximo módulo não ser o mesmo do remetente da mensagem é iniciada a próxima fase do protocolo.

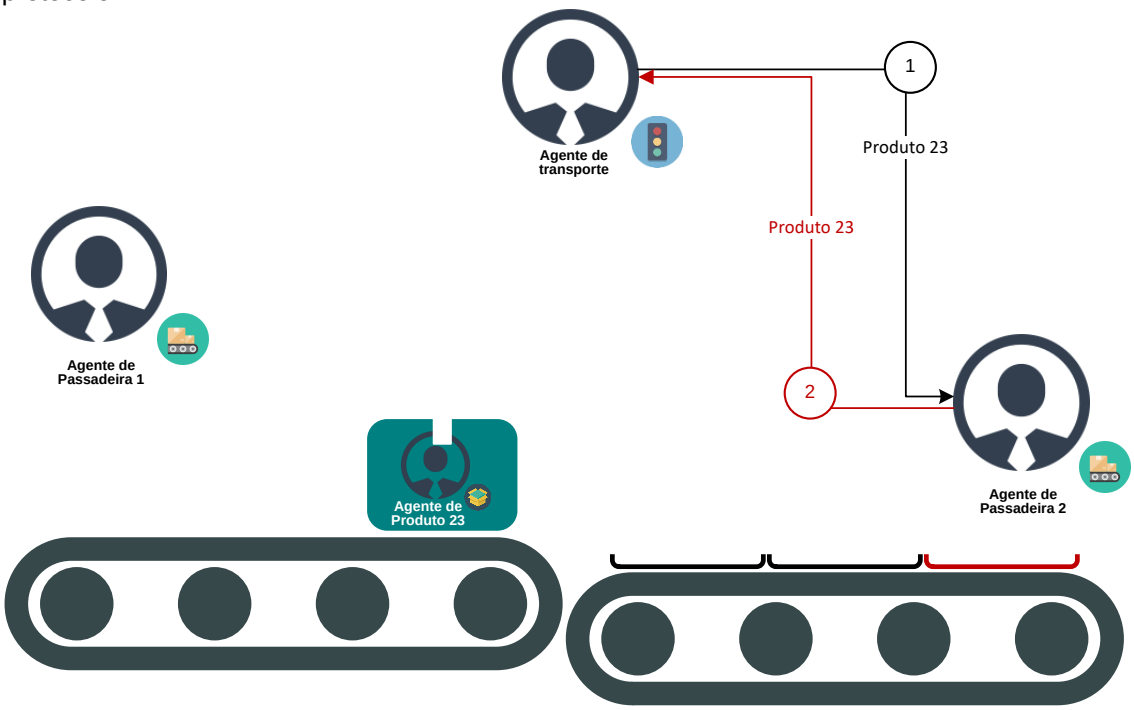

Figura 4.15 Representação da fase de alocação do produto

A segunda fase do transporte, a alocação do produto, tem como objetivo verificar se há espaço no próximo módulo para receber o novo produto, Figura 4.15. Para cumprir esta tarefa, o transport agent envia uma mensagem ao próximo módulo contendo o identificador do produto. Caso esta alocação seja possível o módulo recetor deve reservar a posição e responder de forma afirmativa ao pedido, utilizando a performativa *agree* caso contrário com *refuse*.

Segue-se a terceira fase, a libertação da posição anterior. Esta fase destina-se à libertação da posição anteriormente ocupada pelo produto no módulo de origem, Figura 4.16. Inicia-se com uma mensagem por parte do transport agent, com o identificador do produto, de modo a identificar o processo, para o módulo de origem. Aquando da receção da mensagem o módulo verifica

se está na posição certa de envio, caso não esteja, atualiza-a e em seguida envia uma mensagem do tipo *inform*, para indicar ao transport agent que está pronto para a próxima fase, o transporte propriamente dito.

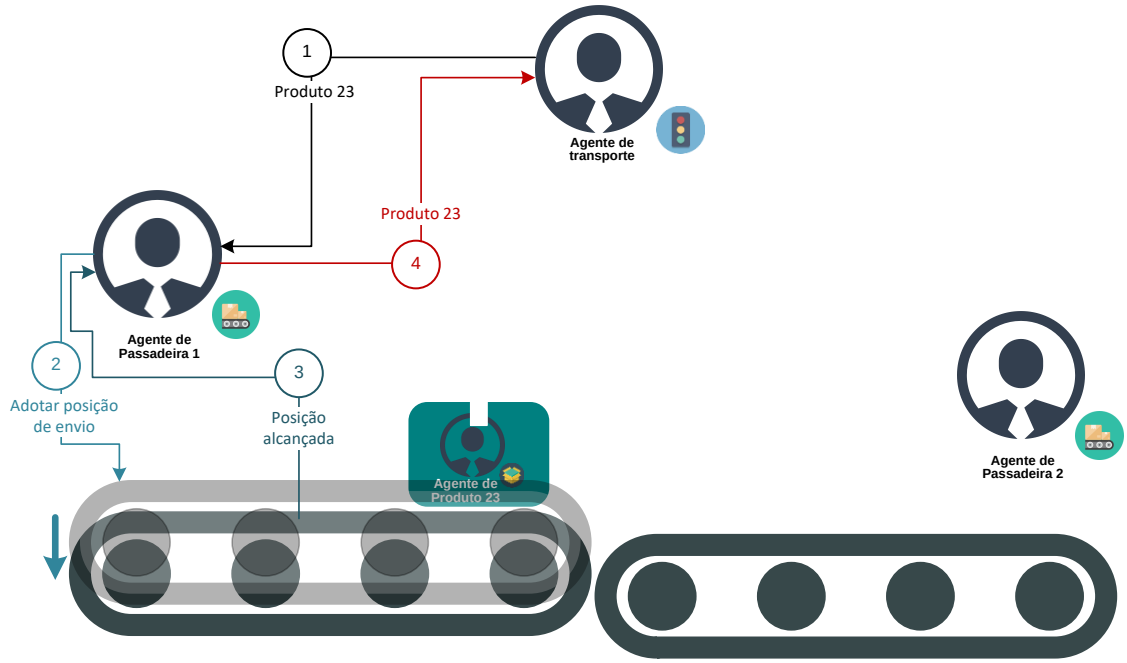

Figura 4.16 Representação da fase de libertação do produto

Na última fase, o transport agent envia uma mensagem ao módulo de destino informando-o que tudo está pronto para que o transporte ocorra, Figura 4.17. Após receber a mensagem do transport agent com os identificadores do produto, do módulo de origem e do destino, os quais guarda numa simples estrutura interna com estas três variáveis, o módulo adapta a sua posição à posição de transição com o módulo de origem. Em seguida ocorre a translação do produto até

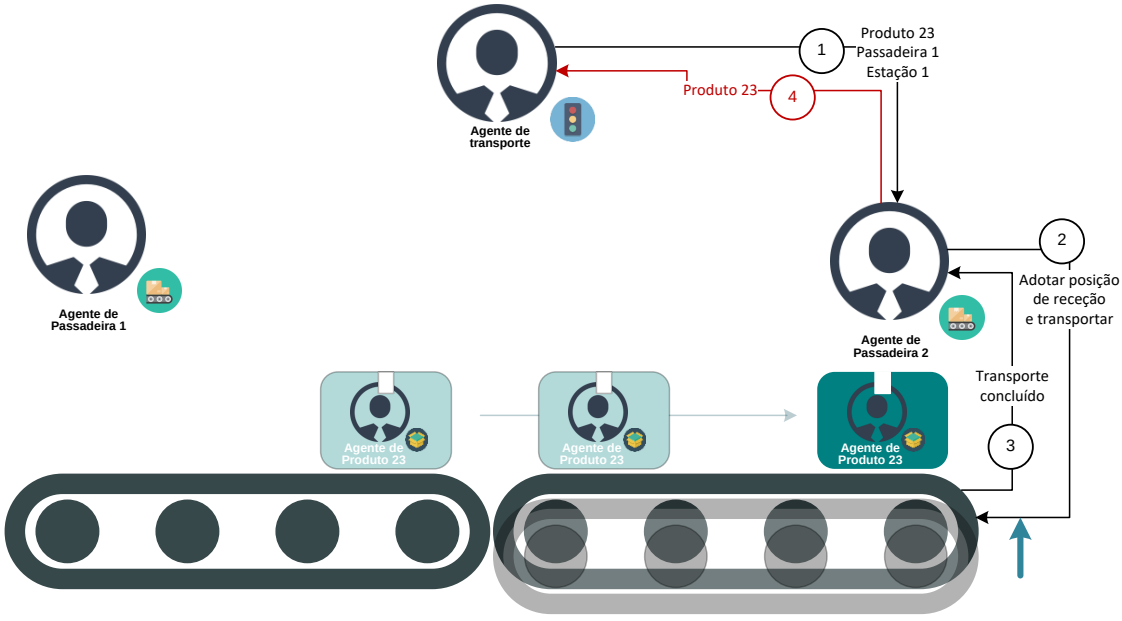

Figura 4.17 Representação da fase de transporte

à posição que este deve ocupar. No fim do movimento, ou seja, quando o produto chega ao local alocado, o módulo recetor envia uma mensagem do tipo *inform* ao transport agent de modo a que este saiba que o transporte foi concluído e possa atualizar a sua posição na lista de controlo.

4.2.2.2. Entrada

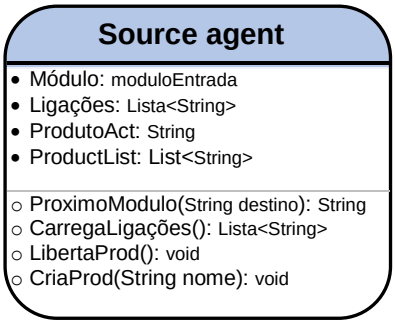

Figura 4.18 Diagrama de classe do source agent

O source agente, tal como o nome indica é o agente responsável pela entrada de produtos no sistema, e para tal precisa de ter funções que o auxiliem nessa tarefa. Para além de ser um agente de execução, visto que interage diretamente com o produto, o source agent é também responsável pelo transporte, assim sendo precisa de implementar alguns dos comportamentos indispensáveis para o mesmo. O source agent apenas precisa de implementar dois dos métodos de transporte, saber qual o próximo módulo e libertar o produto, depois de o enviar. De modo a suportar estas operações o agente guarda uma lista com as suas conexões, de modo a poder responder a qual é o próximo módulo, uma referência do seu módulo no simulador e também uma referência ao produto que atualmente está a alocar, para saber se pode criar um produto novo ou se ainda se encontra ocupado.

Relativamente a funções, o agente precisa de carregar as informações do ficheiro de configuração e de criar o produto, aplicando os modelos e as configurações certas à nova instância criada, e posteriormente pedir ao agente de lançamento que lance o produto na plataforma.

Para dinamizar a utilização da arquitetura foi criada uma classe, com uma simples interface de utilizador, Figura 4.20, que permite lançar produtos de dois tipos pré-definidos, adicionando-os a uma lista, e posteriormente vão sendo lançados à medida que o sistema tem disponibilidade.

4.2.2.3. Saída

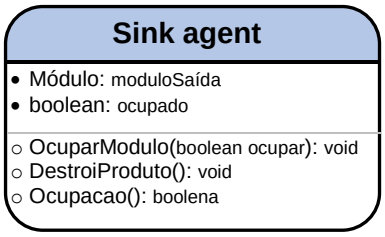

Figura 4.19 Diagrama de classe do sink agent

O próximo agente é o sink agent. Complementarmente ao agente do módulo de entrada, o de saída apenas tem de implementar as fases de transporte de alocar e movimentar, visto que nunca irá enviar o produto para nenhum outro, nem nunca precisará de libertar a posição, pois essa ação está implícita na destruição do produto.

Como variáveis o agente guarda uma referência ao seu módulo do simulador e uma variável de controlo, para saber se está em processo de destruição de algum produto ou não.

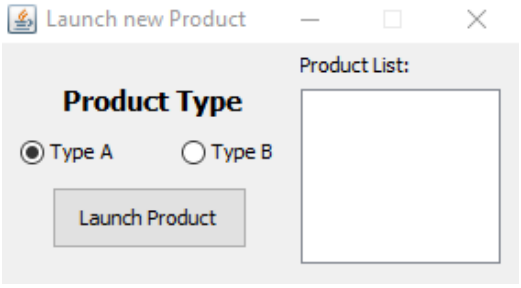

Figura 4.20 Interface de lançamento de produtos

Como função, é essencial a de destruir o produto, a sua principal rotina. Visto ser o módulo mais simples do sistema é também o que carece de um agente com menos comportamentos e funções.

4.2.2.4. Recurso

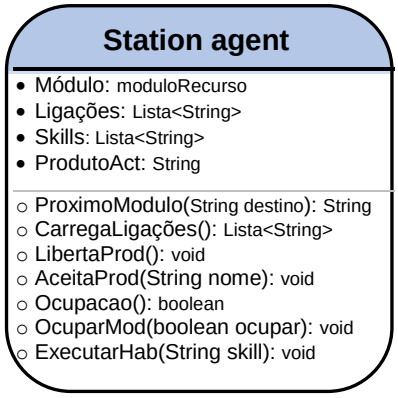

Figura 4.21 Diagrama de classe do station agent

O station agent é o agente responsável por abstrair as estações de trabalho, como por exemplo robôs. É um dos agentes mais completos da arquitetura, visto ter uma componente de transporte e uma componente que atua diretamente no produto, respondendo aos pedidos do mesmo e realizando as tarefas requeridas.

Com vista a garantir o bom funcionamento o station agent guarda, como todos os outros agentes de execução, uma referência ao seu módulo, para que possam interagir.

Nesta arquitetura o station agent também tem um caráter de transporte, assim sendo é necessário que implemente as quatro fases do transporte e seus comportamentos. Para que funcione como desejado, é necessário que o agente tenha presente uma lista com as suas ligações.

De modo a garantir que a estação pode ter mais que uma skill, o agente suporta uma lista onde estas são adicionadas aquando do seu lançamento. Tal como as ligações, as skills disponíveis são carregadas através de um ficheiro.

No que diz respeito às funções são necessárias as funções de transporte, e uma função de execução, que permite, consoante uma variável de entrada escolher entre as várias skills disponíveis e informar o módulo que procedimento deve ser iniciado.

4.2.2.5. Passadeira

A passadeira é um módulo que pode acomodar vários produtos de uma só vez, assim sendo o seu agente mantém uma lista com os produtos presentes no módulo, para garantir que nenhuma das informações é perdida. Outras duas variáveis indispensáveis são: a referência do módulo no simulador, de modo a estabelecer a comunicação e uma variável que indica o máximo

de produtos que a passadeira suporta, desta maneira o agente sabe quando deve parar de encher a lista de produtos.

Devido ao seu caráter de transporte, este agente possui, também, uma lista com as ligações, estas são carregadas quando o agente é lançado, baseando-se nas definições descritas num ficheiro de configuração do sistema.

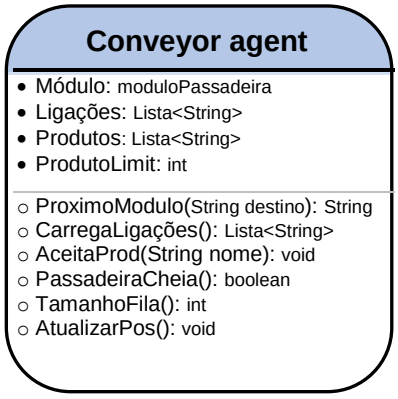

Figura 4.22 Diagrama de classe do conveyor agent

Em relação às funções que este agente implementa, estas relacionam-se com o transporte do produto ou produtos na passadeira. São precisas funções para aceitar produtos, funções para testar se a passadeira já se encontra lotada, funções para atualizar as posições quando sai um produto dos que estavam na passadeira e todas as funções inerentes aos protocolos de transporte descritos acima neste capítulo, nomeadamente para implementação das quatro fases do transporte.

4.2.2.6. Ponto de Encaminhamento

O diverter agent assemelha-se em muito ao conveyor agent, contudo existem diferenças que o fazem ter comportamentos diferenciados. Ambos os módulos têm como função principal o transporte, assim sendo, e tal como o conveyor agent, o ponto de encaminhamento implementa comportamentos que lhe permitem oferecer as rotinas necessárias às quatro etapas do protocolo de transporte.

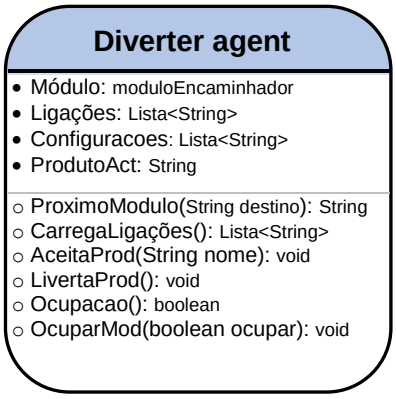

Figura 4.23 Diagrama de classe do diverter agent

Por outro lado, e é aqui que os agentes diferem, o ponto de encaminhamento foi desenvolvido para alocar apenas um produto de cada vez, sendo este utilizado como cruzamento entre passadeiras, e ter um comportamento adaptativo, assim sendo precisa de uma lista de configurações, pois consoante o módulo com que interage este tem definições ou características às quais tem de obedecer para que a transação seja realizada com sucesso. Dando o exemplo de passadeiras a alturas diferentes, ou até mesmo com ângulos de conexão com o ponto de encaminhamento diferentes, deste modo é carregado, tal como as ligações, uma lista de configurações dum ficheiro de configuração.

Em suma, o agente guarda uma lista de conexões, que mapeia as ligações do seu módulo, uma lista de configurações, uma referência ao módulo do simulador e uma referência ao produto que está de momento a transportar.

4.2.2.7. Transporte

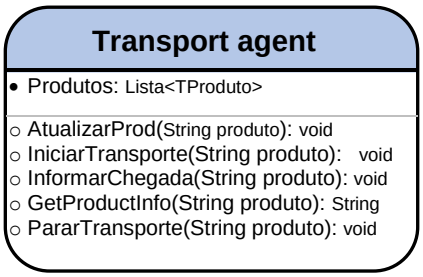

Figura 4.24 Diagrama de classe do transport agent

Surge agora o transport agent, este foi implementado de modo a simplificar o transporte do produto. Com a utilização deste agente, o produto apenas precisa de saber qual o destino para onde se precisa deslocar.

O transport agent é responsável pela implementação do protocolo de transporte descrito acima neste capítulo. Deste modo, para o produto se deslocar no sistema envia um pedido ao transport agent com o identificador do destino para o qual se deseja mover. Desse momento em diante o transport agent inicia o protocolo de transporte, gerindo as trocas de mensagens entre os agentes e informando o produto aquando da sua chegada ao destino.

Sendo este um agente que não representa qualquer módulo no simulador, apenas comporta, como variáveis, uma lista de todos os produtos presentes no sistema. Esta lista é composta por objetos do tipo "TProduto", descrita Figura 4.25, onde este permite identificar os produtos pelos seus identificadores e guardar todos os comportamentos associados, a sua origem, o seu destino e a próxima paragem. Permitindo assim garantir que podemos lidar com mais do que um processo de transporte de produto ao mesmo tempo, pois não corremos o risco de retirar o comportamento ao agente errado, ou baralhar os destinos entre os agentes. As funções presentes nesta classe são apenas para afetar e consultar cada um dos campos nela presentes.

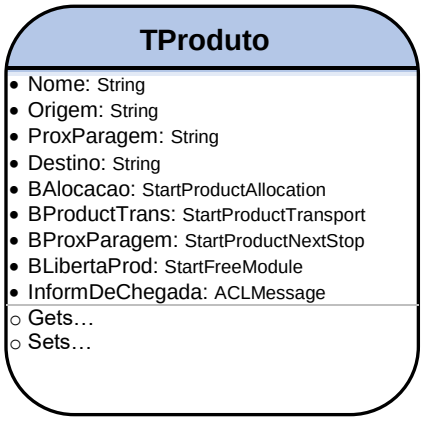

Figura 4.25 Diagrama de classe do produto no transporte

Concluindo, o product agent é responsável pela coerência do transporte, encapsulando o protocolo e simplificando a tarefa de programação do product agent.

4.2.2.8. Produto

Por último, o product agent. Este é um agente apenas de controlo, e assim sendo não precisa de se ligar diretamente ao produto do simulador. A intenção deste agente é organizar e mediar as tarefas necessárias à conclusão do produto em si. Este é responsável pela comunicação, tanto com o transport agent, para pedir transporte para a próxima estação, como na perceção de que estação oferece a tarefa a realizar.

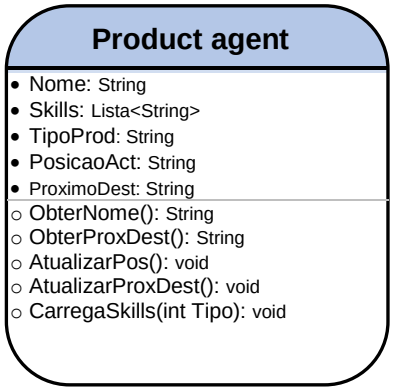

Figura 4.26 Diagrama de classe do product agent

É lançada uma nova instância, quando é criado um produto novo na entrada do sistema, que carrega as informações dum ficheiro de configuração dependendo do tipo de produto que abstrai. Foram implementados dois tipos de produtos, o A e o B, Figura 4.27, de modo a testar a utilização de dois produtos diferentes na arquitetura. Devido ao carregamento das informações no lançamento no produto, é possível alterar as tarefas do produto a qualquer momento, e assim, o próximo produto terá as novas especificações presentes no ficheiro.

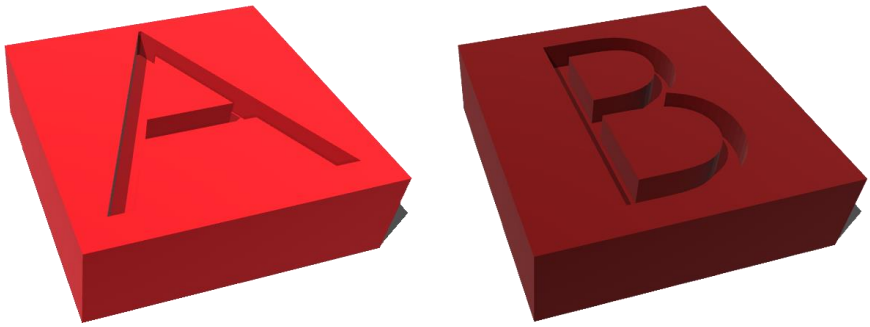

Figura 4.27 Modelos tridimensionais de produtos

Em seguida é apresentado um diagrama de sequência que mapeia o funcionamento do product agent. Na Figura 4.28 podemos observar que o seu comportamento se divide em várias etapas, de onde se destacam duas fases, o transporte e a execução da tarefa. Primeiramente há uma fase de controlo onde as informações são carregadas e o produto se regista no serviço de páginas amarelas, na imagem descrito como DF.

Após procurar quem pode satisfazer a primeira tarefa o agente inicia a fase de transporte. Nesta fase o product agent envia uma mensagem ao transport agent com o seu identificador, o módulo onde se encontra e o módulo de destino, o fornecedor de serviço. Quando recebe o pedido, tenta perceber se o destino é alcançável e responde ao produto afirma ou negativamente. Caso a resposta seja afirmativa, o transport agent inicia o protocolo de transporte descrito anteriormente, controlando todas as interações necessárias para deslocar o produto até ao destino. Quando o produto completa o movimento, o transport agent informa o agente do produto que este já se encontra no destino de modo a que este possa continuar a sua rotina.

A próxima fase, a execução, inicia-se com um pedido por parte do product agent para que a estação comece a execução da tarefa. Após uma análise do seu estado, a estação responde afirma ou negativamente, informando o produto no final do processo.

As etapas descritas acima repetem-se até que todas as tarefas do produto estejam completas e este se possa encaminhar para a saída. Mais uma vez o produto envia um pedido ao transport agent, com o identificador do destino, neste caso a saída.

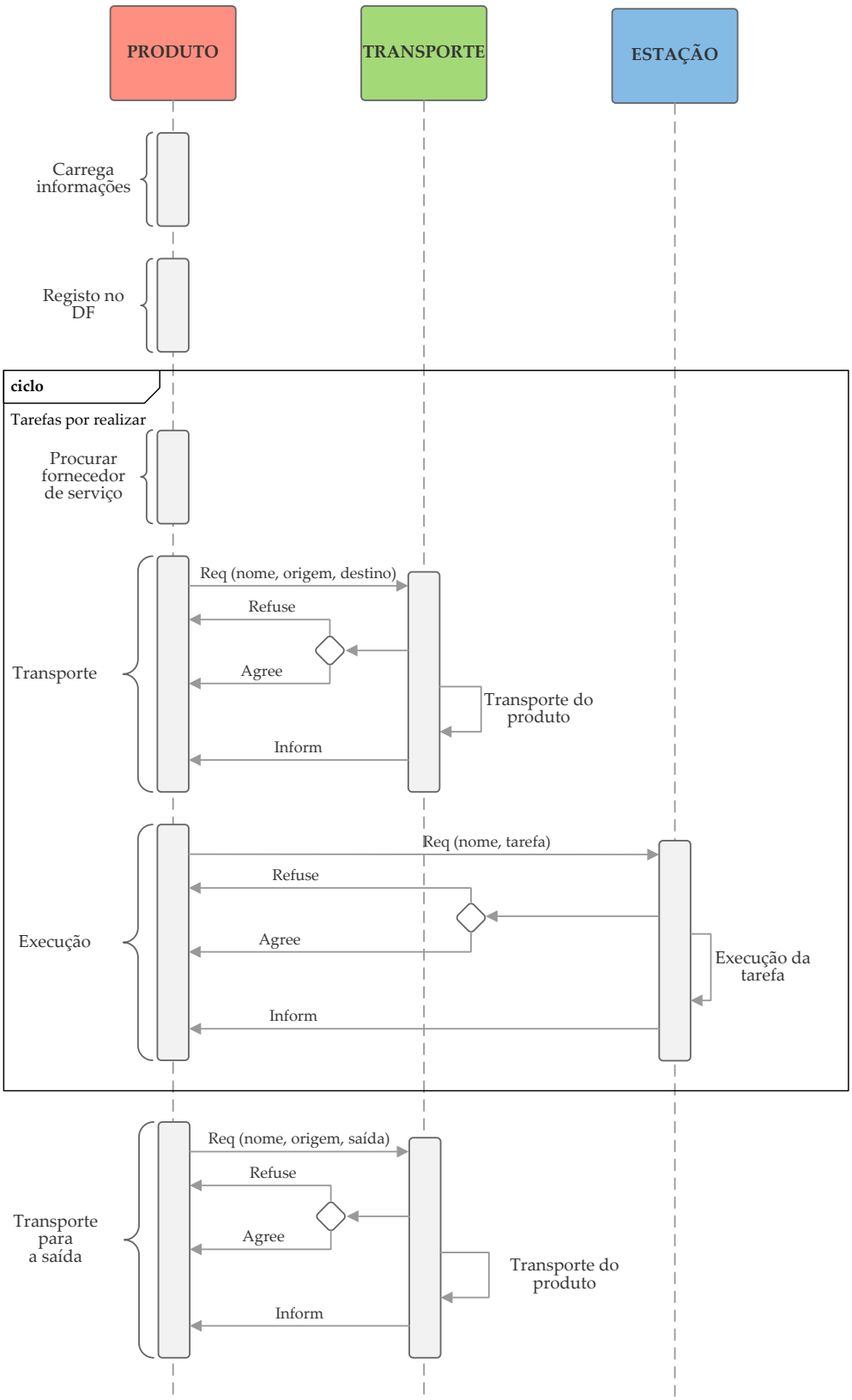

Figura 4.28 Diagrama de sequência do product agent

4.2.3. Primeira abordagem

De modo a garantir que a ideia proposta tem fundamento prático e que a sua implementação é possível, foi construído um pequeno sistema de transporte controlado por agentes, testando as suas capacidades e robustez. O sistema é composto por cinco agentes, três passadeiras, uma entrada e uma saída. Na Figura 4.29 podemos ter uma perceção do sistema em funcionamento.

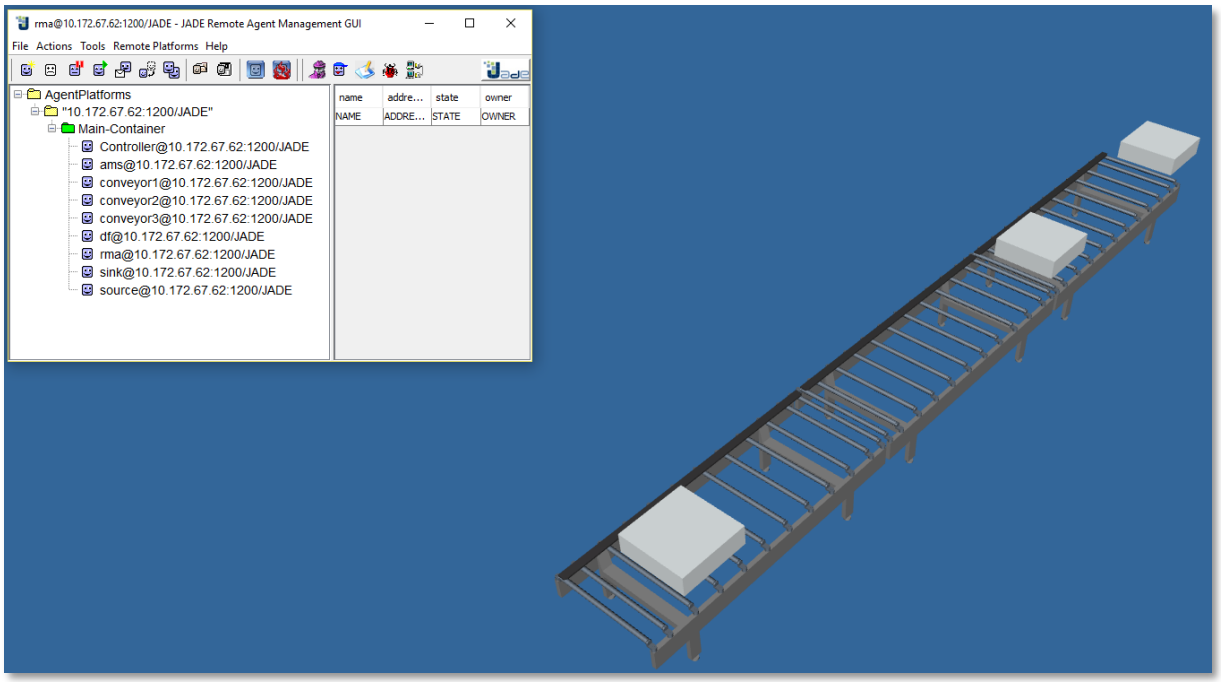

Figura 4.29 Sistema da primeira abordagem

Foram utilizados como modelos as passadeiras e caixas fornecidas pela empresa num exemplo de teste. No entanto a entrada e a saída não têm qualquer representação no simulador, sendo a entrada do lado direito em cima e a saída em baixo à esquerda.

Como teste, foi parado um dos agentes, e como era de esperar houve uma acumulação de caixas à entrada da passadeira que foi retirada. Na Figura 4.30 é mostrado o teste feito, em a passadeira 3 foi excluída, o que levou ao preenchimento da passadeira 2, e à permanência dum produto em cima da passadeira 3, pois como esta não estava funcional, não informou o produto de que este havia chegado ao fim do seu percurso.

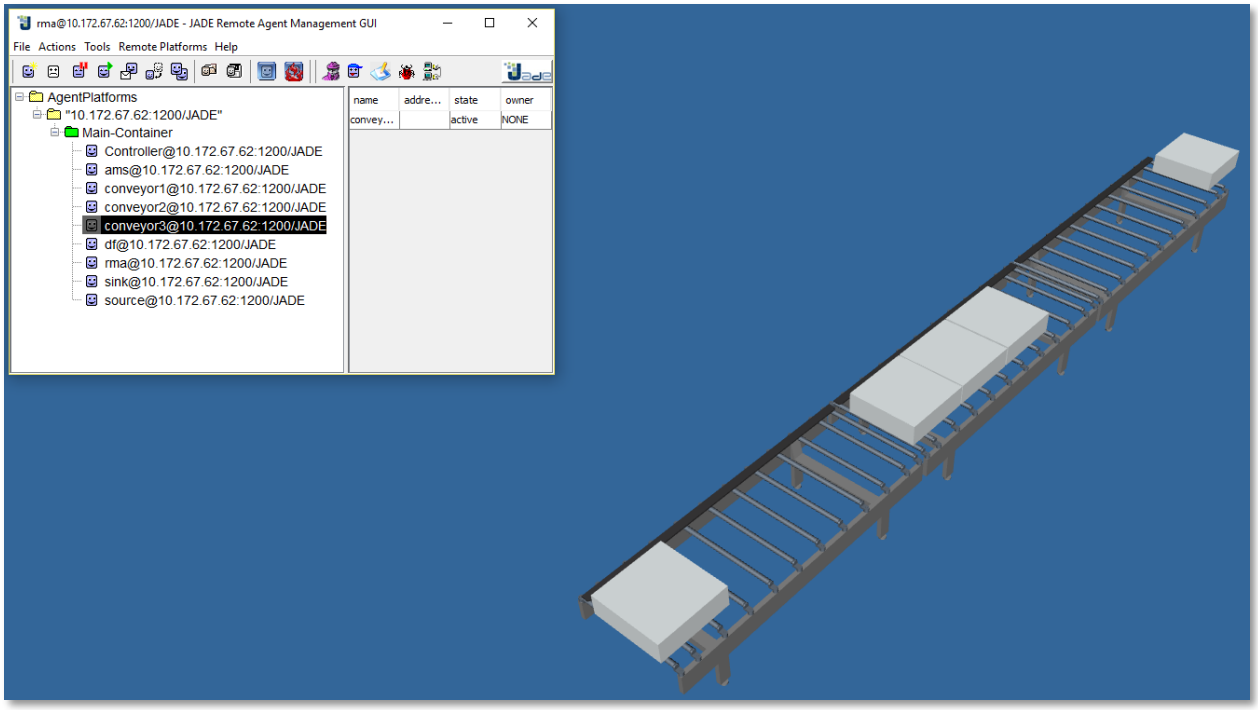

Figura 4.30 Teste ao sistema da primeira abordagem

5

5. Validação

Com o intuito de validar o trabalho apresentado no capítulo anterior, o mesmo foi aplicado a uma linha de montagem presente na instituição de ensino. Este teste permitiu garantir que a arquitetura de integração funciona com uma linha de produção complexa que dispõe de vários tipos de interceções e relacionamentos entre os módulos.

Seguem-se neste capítulo as descrições das várias componentes utilizadas na elaboração do ambiente de simulação, dos agentes utilizados para representar os módulos e dos resultados obtidos através deste teste de validação.

5.1. NOVAFLEX

De modo a validar a arquitetura deste trabalho, esta foi implementada, replicando e simulando um sistema real. A célula de produção foi escolhida devido à sua disponibilidade para estudo, visto estar montada na instituição de ensino, e dispor de um relevante grau de complexidade.

Várias interceções entre componentes e diferentes configurações na transição entre módulos permitem testar vários dos desafios presentes em sistemas de produção industrial.

Na Figura 5.1 está esquematizada a célula, identificando todos os seus componentes e o fluxo de trabalho.

A célula em questão denomina-se NOVAFLEX. É composta por catorze passadeiras, nove pontos de encaminhamento, uma entrada, uma saída e dois robôs ou estações de trabalho. A entrada e a saída do sistema, têm a particularidade de partilharem o mesmo módulo, o que implicou a implementação de um agente especifico para suportar esta caraterística.

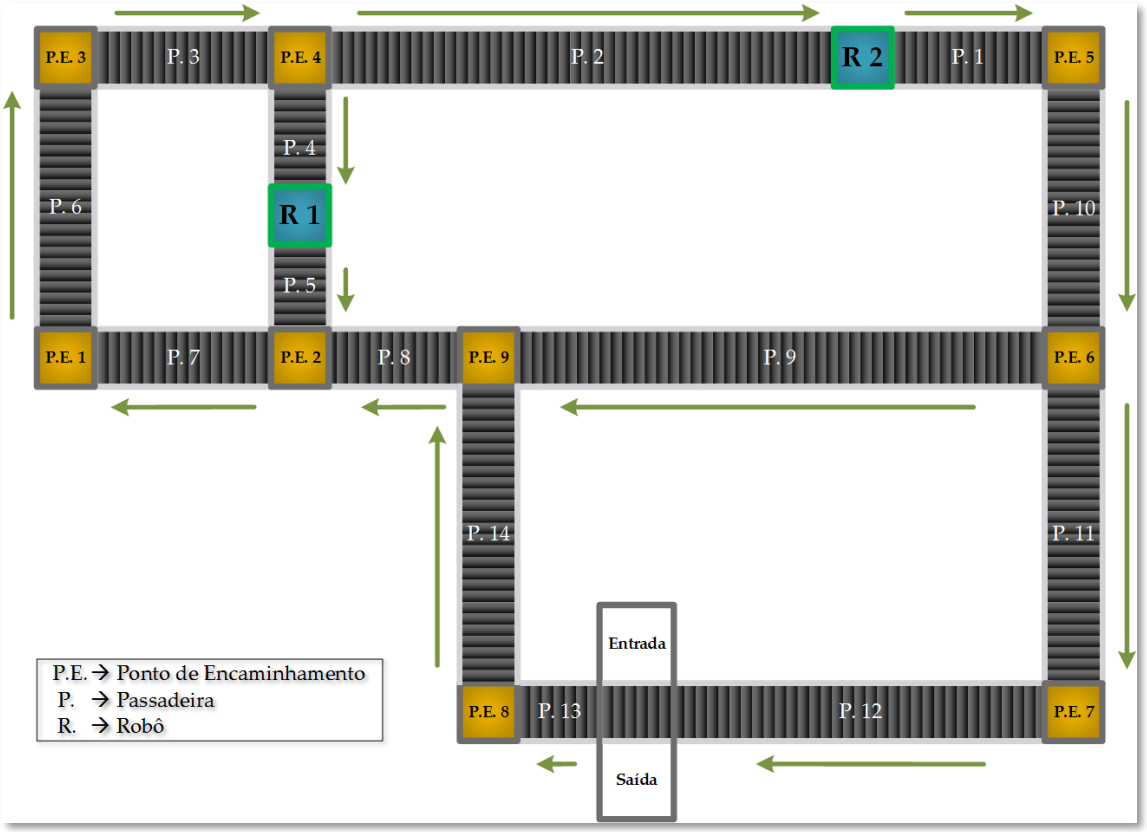

Figura 5.1 Esquema de funcionamento da linha NovaFlex

5.2. Modelos

De maneira a aproximar o sistema simulado do sistema real foi necessário construir três tipos de modelos diferentes. Passadeiras, pontos de encaminhamento e um módulo que conjuga a entrada e a saída de produtos.

O primeiro módulo, a passadeira, foi construída em vários tamanhos, visto que nem todas as componentes do sistema têm a mesma dimensão. Assim sendo, na Figura 5.2 temos a representação de uma passadeira de seis posições. Composto por pés, pernas, armação e esteiras, o modelo apresentado é a conjugação de várias pequenas partes, que permitiram, mais facilmente, construir os restantes tamanhos necessários. Sendo

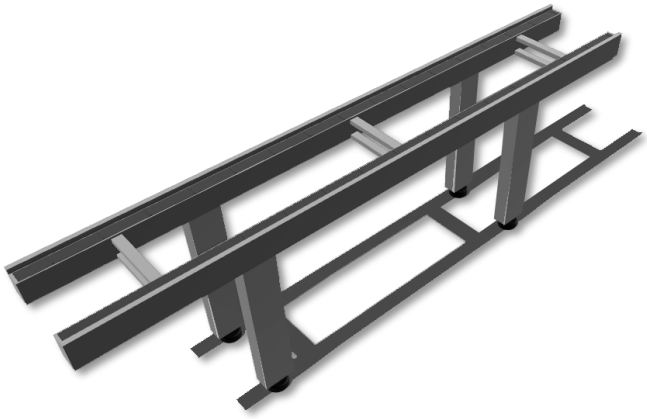

Figura 5.2 Modelo de passadeira

os tamanhos das passadeiras medidos pela quantidade de produtos que estas conseguem acomodar simultaneamente, foram construídos modelos de uma, quatro, seis, sete e vinte e duas unidades.

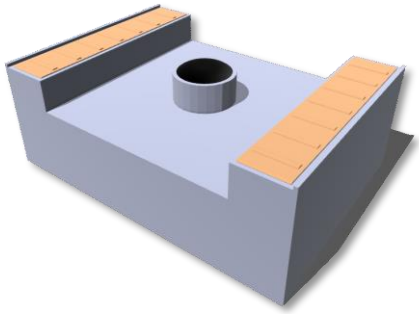

Figura 5.3 Modelo de ponto de encaminhamento

Outro dos modelos implementados foi do ponto de encaminhamento. Sendo um módulo pequeno que se pretende que complete um movimento de translação dentro das passadeiras, foi criado com base no espaço interior à estrutura das mesmas. Na Figura 5.3 temos uma representação dos pontos de encaminhamento. Visto serem todos iguais, variando apenas a sua disposição e orientação ao longo da linha. Este modelo é composto pela estrutura base, duas esteiras castanhas e um botão central que visa a representação de um sensor de pressão.

O último dos modelos desenvolvidos é o módulo que conjuga a entrada e a saída de produtos do sistema. Este módulo é o único que representa mais de um agente, sendo que irá alocar o source agent, o de saída e ainda um agente que se comporta como um ponto de encaminhamento, permitindo a ligação com o resto do sistema. Na Figura 5.4 pode visualizar-se o aspeto deste módulo, com a entrada de um lado, a saída de outro e o ponto de comunicação com a linha na zona central.

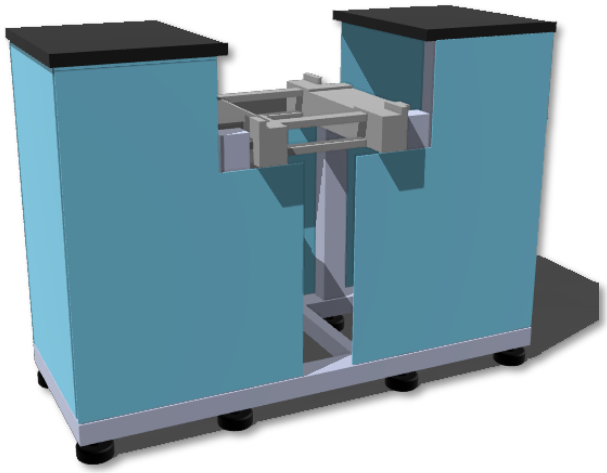

Figura 5.4 Modelo do módulo de entrada e saída

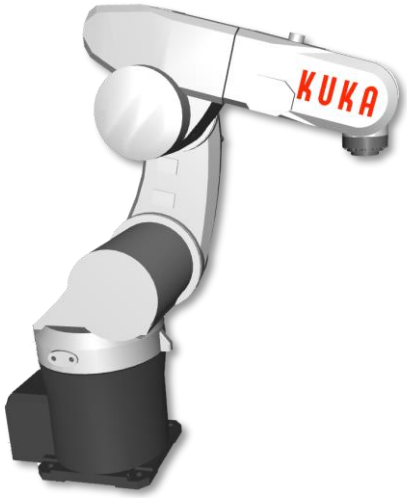

Figura 5.5 Modelo de robô KUKA

Os módulos acima referidos foram desenvolvidos utilizando um programa de CAD, neste caso o SolidWorks, visto ser o único que suporta a exportação no formato ".wrl", exigido pelo simulador.

Foi utilizado mais uma peça na construção do ambiente de simulação. O modelo do robô KUKA foi disponibilizado pela empresa do simulador. Este tem o papel de simular um robô industrial, alojando os agentes de recurso e executando duas tarefas diferentes. Estas diferem no tempo de execução e até na animação apresentada. O modelo representa um braço robótico, o KUKA kr 5 sixx R650, um robô com seis graus de liberdade, o que permite grande flexibilidade e diversidade nas operações a realizar.

5.3. Módulos

Na especificação dos módulos foram tidas em conta variáveis entre cada instância. No caso das passadeiras a configuração é feita através da velocidade a que o produto deve andar e quantos produtos esta consegue alocar em simultâneo. Os pontos de encaminhamento apenas diferem na velocidade a que o seu movimento de translação se realiza. Visto alocarem apenas um produto, não é precisa informação adicional.

O módulo de saída e o de entrada têm algumas características em comum. Sendo estes dois componentes incorporados no mesmo módulo, é necessária alguma informação extra para os especificar. Cada um dos componentes tem um ponto de receção/envio e um ponto de destruição/criação, de modo a que haja uma posição de espera antes do módulo atuar no produto, como demonstrado na Figura 5.6. Tal como nos outros módulos é possível definir a velocidade a que o produto se desloca. A entrada precisa de informações relativas aos modelos de produto a utilizar, ou seja, é preciso dizer à componente de entrada quais os nomes dos ficheiros que descrevem o aspeto visual dos produtos.

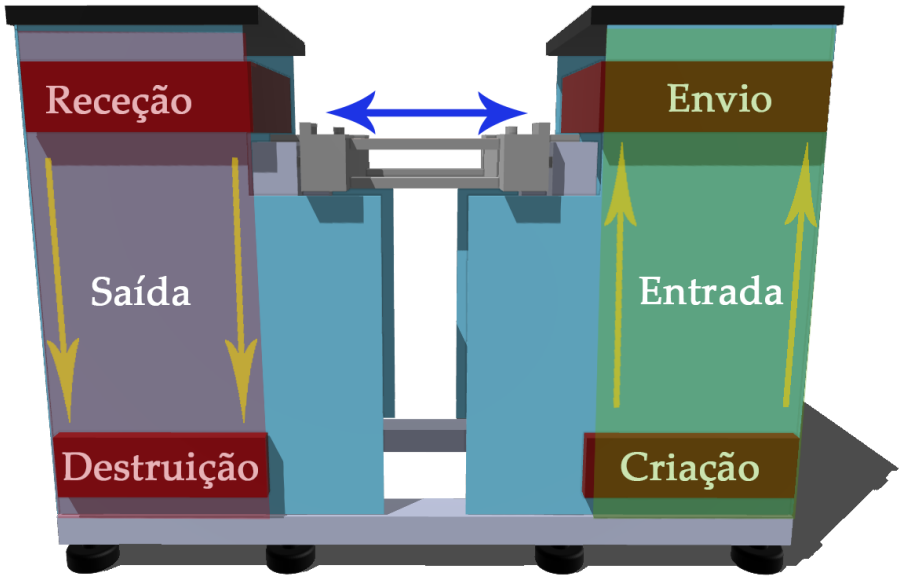

Figura 5.6 Esquema do módulo de entrada e saída

Por fim o módulo do robô, é necessário ter em atenção várias informações para especificar este módulo. Desde os referenciais das posições especificas, como exemplo a posição de descanso e a posição de trabalho, a velocidade dos movimentos, as referências aos eixos de rotação e a tolerância de rotação e translação, são informações essenciais ao bom funcionamento do módulo cedido pela TTS.

Todas as características acima descritas têm em conta a configuração do sistema num ficheiro do tipo XML, tal como descrito no capítulo 4.

5.4. Agentes

Para implementar a célula referida acima foram necessários implementar nove agentes diferentes. Na

Tabela 5.1 surgem os agentes utilizados e a quantidade de cada tipo.

Tipo de agente	Número
Agente de páginas amarelas	1
Agente de lançamento	1
Source agent	1
Sink agent	1
Agente de Execução	2
Conveyor agent	14
Diverter agent	9
Transport agent	1
Source and sink agent	1
Total	31

Tabela 5.1 Agentes utilizados

Como agente de páginas amarelas foi utilizado o serviço de DF, da plataforma multiagente, o JADE.

Todos os agentes incluídos na tabela estão descritos pormenorizadamente no capítulo da implementação. Aparece nesta tabela um agente que até agora não foi falado pois é um caso especifico e foi desenvolvido apenas para esta solução. O seu comportamento é em tudo semelhante a um ponto de encaminhamento, visto que a sua funcionalidade é ligar o módulo de entrada e saída ao resto do sistema.

Na tabela apresentada não foram referidos agentes de produto pois estes são criados em tempo real, e dependendo do teste que se pretende implementar, o seu número varia.

5.5. Análise de resultados

Na Figura 5.7 é apresentada uma vista superior do sistema modelado. Como é percetível, e comparando com a Figura 5.1 a linha de montagem tem alguns módulos, mais propriamente algumas passadeiras, que foram divididos em partes para que a sua implementação com agentes fosse mais clara e de fácil compreensão.

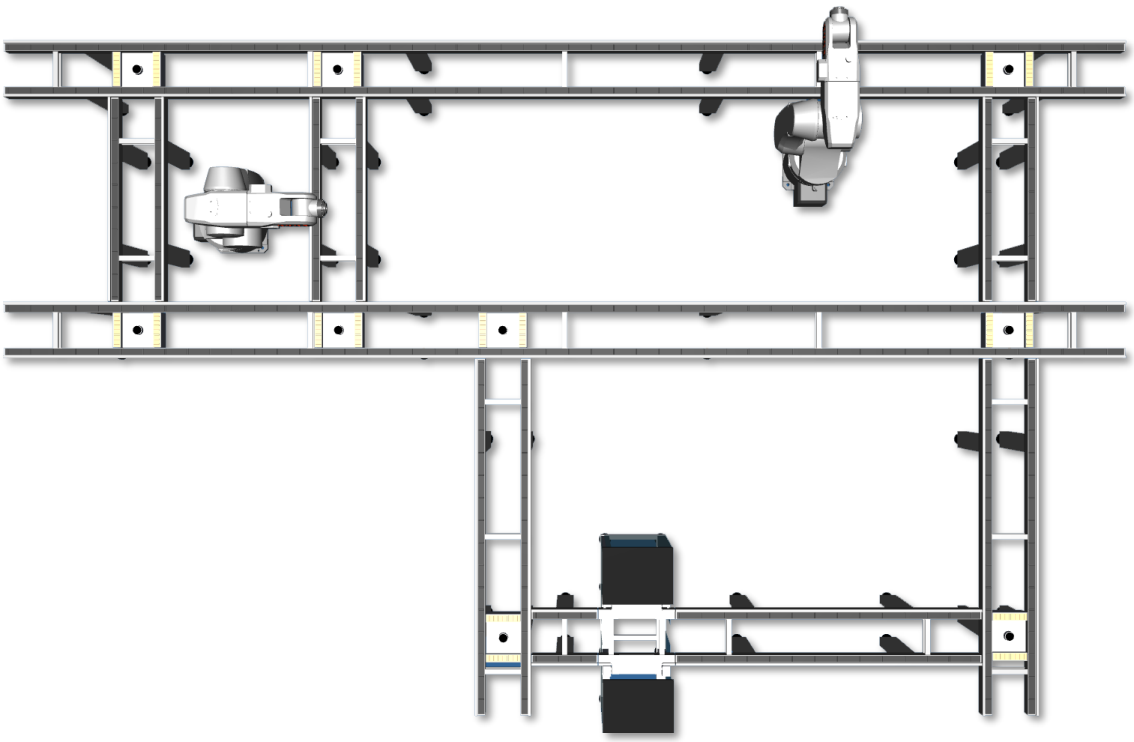

Figura 5.7 Vista superior do ambiente de simulação

Em seguida são apresentadas uma figura representativa do sistema virtualizado, retirada do simulador, Figura 5.8, e uma fotografia do sistema real, Figura 5.9. É possível observar que vários foram os esforços, no que diz respeito ao acerto de detalhes, de modo a que a solução alcançada fosse o mais parecida possível com a célula real. No entanto, devido a este ser o primeiro contacto com programas de CAD e simuladores de máquinas industriais, o resultado não é perfeito. É de notar também que os robôs utilizados no simulador não são iguais aos robôs presentes na linha, pelo que o intuito do trabalho não é o de recriar fidedignamente a linha num ambiente simulado, mas sim, utilizar a célula de fabrico como ponto de partida e integrar uma arquitetura multiagente com um simulador presente no mercado.

Seguem-se algumas imagens que pretendem mostrar o bom funcionamento da linha no simulador. Para isso foram realizados vários testes, e introduzidos vários produtos. Foi também testada a capacidade da linha se adaptar a novas especificações nos produtos, sendo esta capaz de lançar produtos com as alterações após estas serem feitas nos ficheiros que descrevem os mesmos.

Figura 5.9 Fotografia do sistema real NOVAFLEX

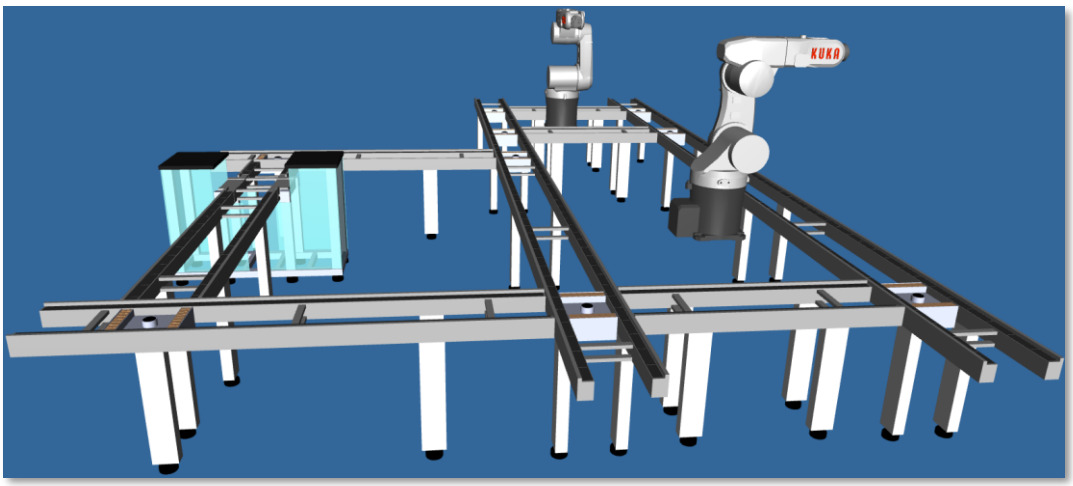

Figura 5.8 Virtualização do sistema NOVAFLEX

Foram criados dois tipos de produtos, o produto A, que inicialmente requeria a tarefa 1, e o produto B, que para ficar completo necessita da tarefa 2. Cada uma das tarefas é disponibilizada pela máquina 1 e 2 respetivamente. Assim sendo, quando os produtos são lançados no sistema, estes carregam a informação dum ficheiro de configuração e em seguida seguem os protocolos já explicados acima.

Na Figura 5.11 é exibida uma vista do sistema em funcionamento à espera de produtos. Este lançamento é feito através da interface gráfica visível no canto inferior esquerdo, a branco.

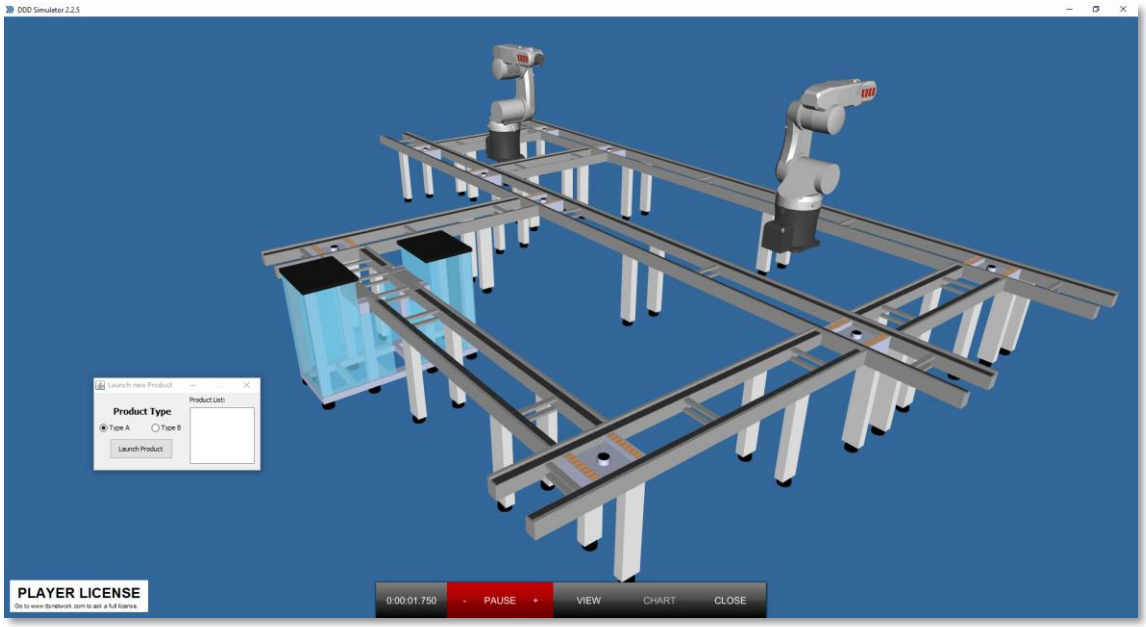

Figura 5.11 Sistema à espera do lançamento de produtos

A Figura 5.10 mostra que foram lançados no sistema um produto de cada tipo, e como era de esperar o produto A dirigiu-se para a estação 1, a estação que oferece a tarefa pretendida, e o produto B foi encaminhado para a estação 2.

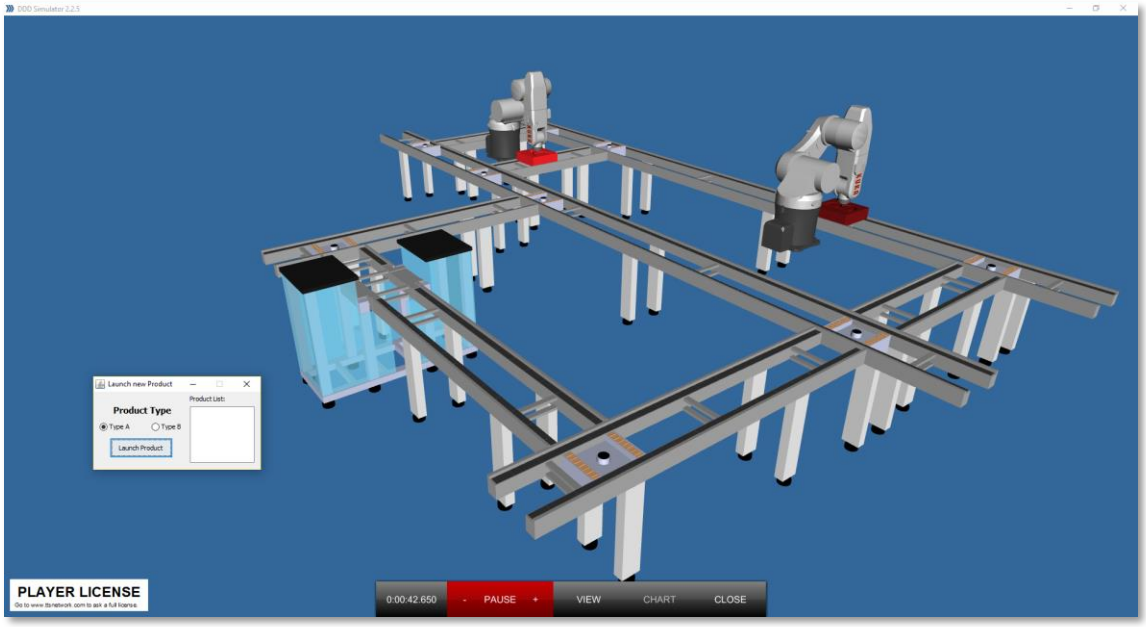

Figura 5.10 Sistema com um produto do tipo A e um do tipo B

O comportamento das máquinas difere na animação apresentada e na duração dependendo da tarefa que o produto pede que seja implementada. Após a conclusão das tarefas os produtos dirigem-se para o módulo de saída, onde são tratados por forma a serem apagados do sistema, pois o seu percurso está finalizado. A próxima figura, a Figura 5.12, mostra os produtos A e B a saírem das respetivas máquinas e a dirigirem-se para a saída do sistema.

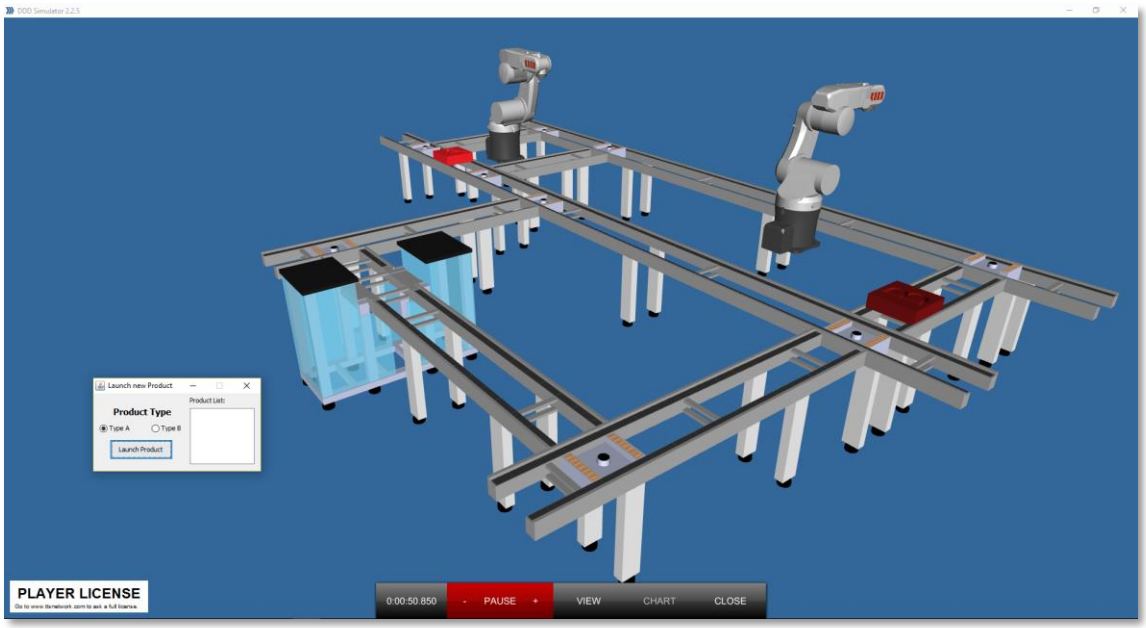

Figura 5.12 Produtos completos e a sair do sistema

Como próximo teste procedeu-se à alteração da configuração do produto B, sendo este agora completo depois de realizar a tarefa 2 e 1, por esta ordem. Na Figura 5.13 vemos o lançamento de vários produtos do tipo A e do tipo B. É possível observar na interface de lançamento a lista dos produtos em espera para serem lançados.

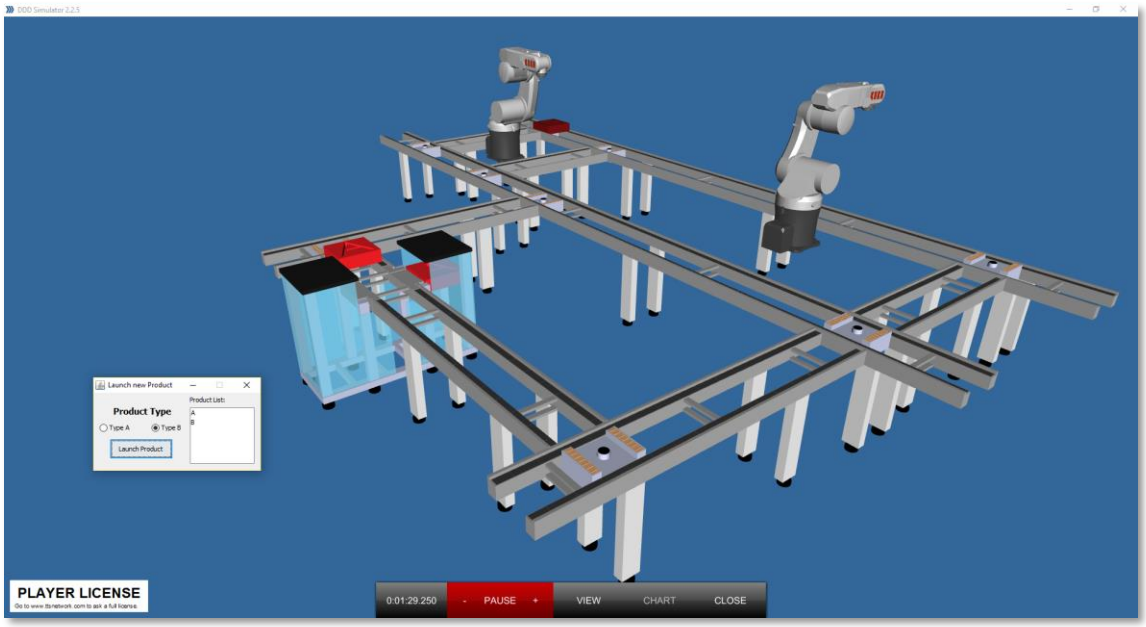

Figura 5.13 Lançamento de vários produtos do tipo A e B

A Figura 5.14 é mostrado o encaminhador a enviar o produto A para a estação 1 enquanto que o produto B, que requer uma tarefa diferente, foi enviado para a estação 2.

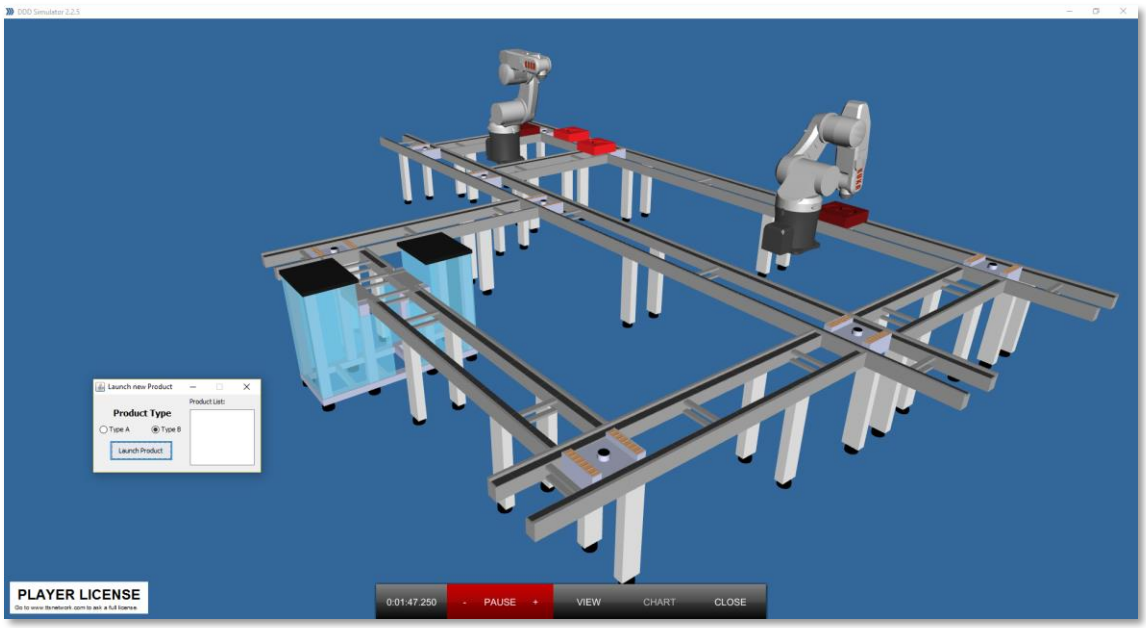

Figura 5.14 Encaminhador a filtrar produtos A e B

Após a conclusão da sua primeira tarefa, o produto B sai da estação 2 e requer ao sistema uma estação que protagonize a sua próxima tarefa. Em reação a esse pedido o sistema responde com o identificador da estação 1, e em seguida o produto pede ao transport agent que o leve até ao seu novo destino. Na Figura 5.15 é possível ver que o produto B alterou o comportamento em relação ao teste anterior. Este segue agora em direção à estação 1 de modo a completar a sua segunda e última tarefa.

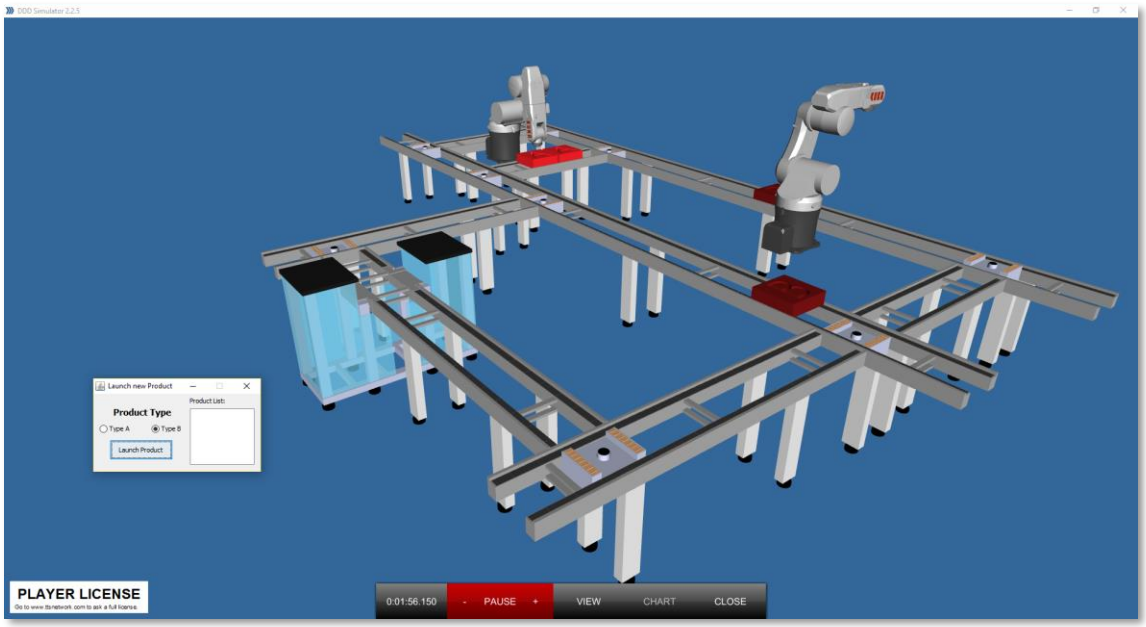

Figura 5.15 Produto B encaminhado para a próxima tarefa

Por fim, na Figura 5.16 é possível observar o produto B a executar a sua segunda tarefa, na estação 1 e o produto A a dirigir-se para a saída do sistema.

Os produtos podem ser compostos por um número indefinidos comportamentos, pelo que se o produto tem dois comportamentos iguais seguidos, ou dois comportamentos diferentes que sejam disponibilizados pela mesma estação, este efetua a sua execução de forma sequencial na máquina antes de sair do módulo. Isto acontece devido à implementação do protocolo de transporte, pois como o produto já se encontra no módulo que dispõe da função seguinte, apenas precisa de pedir a execução e assim satisfazer a sua tarefa.

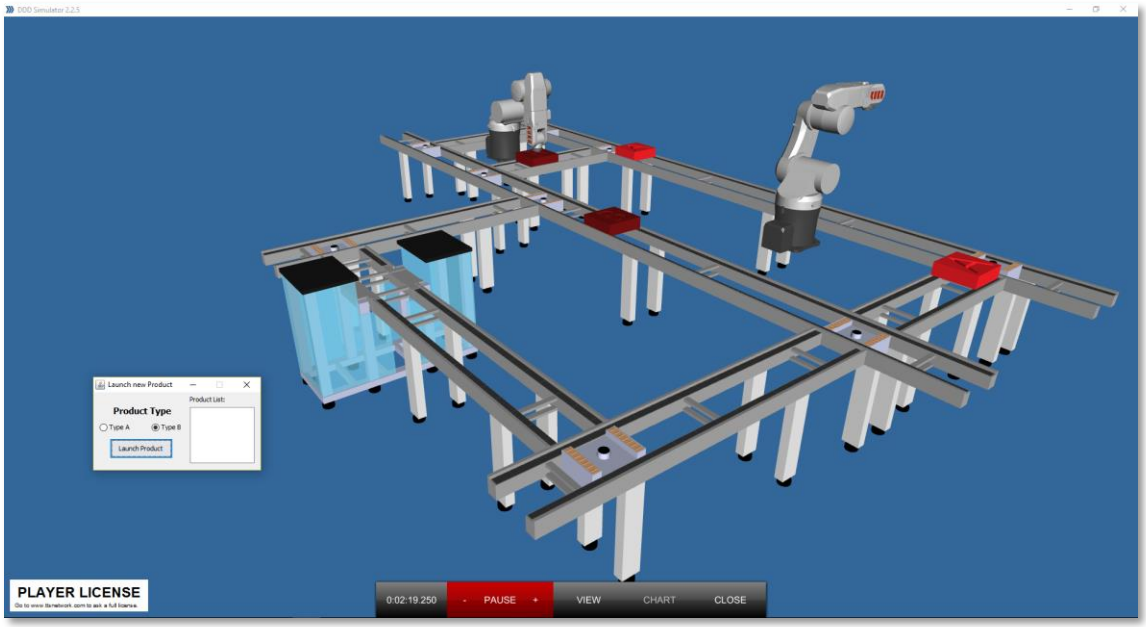

Figura 5.16 Produto B a realizar a segunda tarefa e A a sair do sistema

Com a sequência de imagens mostrada é possível comprovar-se que o sistema se adapta às configurações dos produtos, satisfazendo os seus pedidos.

Um último pormenor prende-se com o caso de o sistema não suportar alguma tarefa do produto. Se assim for, este é encaminhado para a saída e termina o seu percurso no sistema para que não atrapalhe o funcionamento do resto da linha.

6

6. Conclusões e Trabalho Futuro

6.1. Conclusões

Com o aumento da complexidade dos sistemas de manufatura, a programação de sistemas distribuídos é cada vez mais uma necessidade na indústria. À medida que novos paradigmas surgem no mercado as ferramentas de células de produção atuais ficam desatualizadas e carecem de uma total restruturação para saciarem os mais recentes problemas. De modo a facilitar ou até evitar essa restruturação, são necessárias ter em conta arquiteturas adaptativas que se adaptem às necessidades e as colmatem em tempo real.

No entanto, testar novas soluções pode implicar uma paragem de produção, o que pode por em causa a subsistência das empresas e a sua saúde financeira. Com o evoluir do tempo a simulação foi tomando grande importância na modelação de sistemas, pois um bom modelo é capaz de prever possíveis problemas e até servir de ferramenta de educação a operários especializados.

Devido às capacidades referidas torna-se indispensável a utilização de simuladores na indústria. Apesar de existirem grandes avanços na área da simulação, nomeadamente nos aspetos estéticos e até funcionais, estes têm em conta, na sua maior parte, sistemas convencionais de manufatura. Assim sendo é preciso dar um salto para os novos paradigmas e para a simulação distribuída.

O trabalho desenvolvido caminha neste sentido, pelo que integra um simulador de excelência, presente no mercado e com provas dadas na sua eficácia e fiabilidade, com uma arquitetura que tem por base os novos paradigmas da manufatura. A distribuição de responsabilidade pelos vários agentes e a abstração de cada peça da linha de produção por esses agentes, garante uma aproximação aos paradigmas atuais e permite ter as características de flexibilidade e adaptabilidade desejadas.

Outra das mais valias abordadas neste projeto é a alteração das especificações dos produtos, pelo que o agente pode consultar um ficheiro de configuração do produto e com a simples

alteração do ficheiro, a produção do produto altera-se e assim são cumpridas alterações nas especificações em tempo real.

A utilização do simulador DDDSimulator garantiu uma boa base de partida para o sistema implementado, pois a sua organização e implementação é clara e sucinta, tendo por base descrições padrão que permitem uma fácil compreensão do funcionamento dos módulos e implementação de novos designs. A integração de tecnologias como o XML, são fundamentais para a criação de standards e para garantir uma fácil compreensão do seu funcionamento.

Por outro lado, a plataforma utilizada para implementação do sistema multiagente, o JADE, permitiu uma grande flexibilidade na implementação dos agentes, mostrando ser uma grande ferramenta para implementação de arquiteturas distribuídas. As ferramentas por esta plataforma disponibilizadas foram essenciais para o desenvolvimento do projeto, permitindo encontrar erros de forma rápida, solucionar problemas de implementação e até um rastrear se as interações eram as desejadas.

Relacionado com a construção dos modelos está a criação das representações tridimensionais dos objetos a simular. Neste caso foi utilizado o programa de desenho assistido por computador SolidWorks, este bastante utilizado no campo da simulação e na construção de modelos virtuais nas mais variadíssimas áreas. O programa foi utilizado devido às suas funcionalidades serem as mais completas e o formato de modelo desejado ser suportado pela ferramenta. Contudo surge a necessidade de um tempo de aprendizagem algo prolongado. Devido à vasta gama de possibilidades oferecidas pela ferramenta e uma interface repleta de campos opcionais, torna-se difícil uma utilização rápida e sem grandes preocupações. Seria uma sugestão criar um modo de utilização simples, onde fossem apresentados apenas controlos principais na construção de peças tridimensionais.

Concluindo, o trabalho realizado propõe uma integração de duas tecnologias distintas com vista à evolução, tanto no ramo de investigação como industrial, e na adaptação de meios já desenvolvidos e com grande qualidade, às necessidades mais atuais, como é o caso dos paradigmas de manufatura. Ficou provado que esta integração é possível sem alterar qualquer das tecnologias envolvidas, tendo apenas em conta as interfaces de interação entre os dois sistemas.

No âmbito da contribuição cientifica, foi escrito um artigo referente ao trabalho realizado (Andre Dionisio Rocha, Barroca, Maso, & Barata, 2016). Este pretende descrever o trabalho desenvolvido e assim abordar a temática da simulação de sistemas distribuídos. Foi apresentado na conferência SOHOMA'16 (http://www.sohoma16.cimr.pub.ro/index.html). Nesta conferência foram apresentadas outras abordagens, com simuladores diferentes. De modo a contribuir ainda mais para o avanço nesta área, está a ser desenvolvido um artigo para posterior publicação numa revista internacional de modo a comparar as arquiteturas e as soluções propostas.

6.2. Trabalho Futuro

Algum trabalho futuro pode passar pela integração de outras arquiteturas de transporte ou de produção com o simulador utilizado. A forma como este trabalho foi desenvolvido abre portas

a novas formas de interagir com arquiteturas multiagente, pois foram tidas em conta interações genéricas.

Elaborar uma forma automática de definir os módulos e os seus pontos de referência pode ser uma mais valia na remoção de erros e no aumento da rapidez com que as soluções são implementadas no simulador. Por exemplo definir uma passadeira com quatro posições carece de quatro referenciais para alocação de produtos, será interessante que dando o tamanho do produto e a orientação da passadeira, o simulador se organize de forma automática e crie os referenciais necessários ao seu bom funcionamento.

Outro aspeto que seria interessante abordar é a integração deste software com uma aplicação de estatística, sendo possível recolher dados essenciais e que podem ser indicadores de produção fundamentais para uma melhoria nos métodos de produção.

Por fim, ter em conta a possibilidade de adicionar e remover módulos em tempo real ao simulador seria o ideal. Simular em tempo real linhas adaptativas será o caminho a seguir e certamente um ponto a alcançar. A interface do simulador pode ser melhorada, permitindo ao utilizador obter mais informação sobre a corrente simulação e proceder em tempo útil às alterações necessárias à melhoria da performance do sistema. Controlos mais interativos seriam uma forte mais valia a integrar no presente sistema.

7

7. Referências

AutoDesk, I. (2016). AutoCAD. Retrieved July 28, 2016, from http://www.autodesk.pt/products/autocad/overview

Babiceanu, R. F., & Chen, F. F. (2006). Development and Applications of Holonic Manufacturing Systems: A Survey. *Journal of Intelligent Manufacturing*, *17*(1), 111–131. http://doi.org/10.1007/s10845-005-5516-y

Barata, J., Camarinha-Matos, L., & Onori, M. (2005). A Multiagent Based Control Approach for Evolvable Assembly Systems, 478–483. http://doi.org/10.1109/INDIN.2005.1560423

Barbosa, J., Leitão, P., Adam, E., & Trentesaux, D. (2015). Dynamic self-organization in holonic multi-agent manufacturing systems: The ADACOR evolution. *Computers in Industry*, *66*, 99–111. http://doi.org/10.1016/j.compind.2014.10.011

Bellifemine, F., Poggi, A., & Rimassa, G. (2001). JADE. In *Proceedings of the fifth international conference on Autonomous agents - AGENTS '01* (pp. 216–217). New York, New York, USA: ACM Press. http://doi.org/10.1145/375735.376120

Blender Foundation. (2016). Blender. Retrieved July 28, 2016, from https://www.blender.org/

Blum, C. (2005). Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*. http://doi.org/10.1016/j.plrev.2005.10.001

Boudreau, B. T., Glick, J., Greene, S., Spurlin, V., & Woehr, J. J. (2002). NetBeans: The Definitive Guide,. *Building*, (October). Retrieved from https://books.google.pt/books?hl=pt-PT&lr=&id=JCMBWozfcJUC&oi=fnd&pg=PR5&ots=Wy6V0e1ef0&sig=sY_St9t0xzqa6WPmHPMZwKtP0cs&redir_esc=y#v=onepage&q&f=

false

Buccioli, A. A. B., Zorzal, E. R., & Kirner, C. (2006). Usando Realidade Virtual e Aumentada na Visualização da Simulação de Sistemas de Automação Industrial, (19), 3124–1560. Retrieved from http://www.lbd.dcc.ufmg.br/colecoes/svr/2006/033.pdf

Cavalcante, A. L. D., Pereira, C. E., & Barata, J. (2010). Component-Based Approach to the Development of Self-X Automation Systems. *IFAC Proceedings Volumes*, *43*(4), 222–227. http://doi.org/10.3182/20100701-2-PT-4011.00039

Clark, J., & Daigle, G. (1997). The importance of simulation techniques in ITS research and analysis. In *Proceedings of the 29th conference on Winter simulation - WSC '97* (pp. 1236–1243). New York, New York, USA: ACM Press. http://doi.org/10.1145/268437.268766

Coppela Robotics Gmbh. (2016). V-rep, Virtual Robot Experimentation Platform. Retrieved July 28, 2016, from http://www.coppeliarobotics.com/index.html

Dassault Systemes SolidWorks Corporation. (2016). SolidWorks. Retrieved July 28, 2016, from http://www.solidworks.com/

Di Orio, G. (2013). Adapter module for self-learning production systems. Retrieved from http://hdl.handle.net/10362/10402

Di Orio, G., Rocha, A., Ribeiro, L., & Barata, J. (2015). The PRIME Semantic Language : Plug and Produce in Standard - based Manufacturing Production Systems. Retrieved from https://www.researchgate.net/profile/Giovanni_Di_Orio/publication/278410738_The_PRIME_Semantic_Language_Plug_and_Produce_in_Standard-based_Manufacturing_Production_Systems/links/55808e3b08aea3d7096e4bc5.pdf

Drogoul, A., & Ferber, J. (1994). Multi-agent simulation as a tool for modeling societies: Application to social differentiation in ant colonies (pp. 2–23). Springer Berlin Heidelberg. http://doi.org/10.1007/3-540-58266-5_1

ElMaraghy, H. A. (2005). Flexible and reconfigurable manufacturing systems paradigms. *International Journal of Flexible Manufacturing Systems*, *17*(4), 261–276. http://doi.org/10.1007/s10696-006-9028-7

FIPA ACL Message Structure Specification. (2016). Retrieved from http://www.fipa.org/specs/fipa00061/SC00061G.html

Freese, M., Singh, S., Ozaki, F., & Matsuhira, N. (2010). Virtual Robot Experimentation Platform V-REP: A Versatile 3D Robot Simulator (pp. 51–62). Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-

17319-6_8

Frei, R., Ribeiro, L., Barata, J., & Semere, D. (2007). Evolvable Assembly Systems: Towards User Friendly Manufacturing. In *2007 IEEE International Symposium on Assembly and Manufacturing* (pp. 288–293). IEEE. http://doi.org/10.1109/ISAM.2007.4288487

Gosling, J., & Mcgilton, H. (1996). The Java™ Language Environment. Retrieved from http://www.hs-augsburg.de/informatik/projekte/mebib/emiel/entw_inf/lernprogramme/java/Tools/Java/Doc/Papers/langenviron.pdf

Hoeger, H. R., & Jones, D. W. (1996). Integrating Concurrent and Conservative Distributed Discrete-Event Simulators. *SIMULATION*, *67*(5), 303–314. http://doi.org/10.1177/003754979606700502

Jacintho, L. F. O., Batista, A. F. M., Ruas, T. L., Marietto, M. G. B., & Silva, F. A. (2010). An agent-based model for the spread of the Dengue fever. In *Proceedings of the 2010 Spring Simulation Multiconference on - SpringSim '10* (p. 1). New York, New York, USA: ACM Press. http://doi.org/10.1145/1878537.1878540

Jahromi, M. H. M. A., & Tavakkoli-Moghaddam, R. (2012). A novel 0-1 linear integer programming model for dynamic machine-tool selection and operation allocation in a flexible manufacturing system. *Journal of Manufacturing Systems*, *31*(2), 224–231. http://doi.org/10.1016/j.jmsy.2011.07.008

Kanaga, E. G. M., & Valarmathi, M. L. (2012). Multi-agent based patient scheduling using particle swarm optimization. In *Procedia Engineering* (Vol. 30, pp. 386–393). http://doi.org/10.1016/j.proeng.2012.01.876

Klingstam, P., & Gullander, P. (1999). Overview of simulation tools for computer-aided production engineering. *Computers in Industry*, *38*(2), 173–186. http://doi.org/10.1016/S0166-3615(98)00117-1

KOESTLER, A. (1968). THE GHOST IN THE MACHINE. Macmillan.

Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G., & Van Brussel, H. (1999). Reconfigurable Manufacturing Systems. *CIRP Annals - Manufacturing Technology*, *48*(2), 527–540. http://doi.org/10.1016/S0007-8506(07)63232-6

Law, A. M., & Mccomas, M. G. (1998). SIMULATION OF MANUFACTURING SYSTEMS. *Proceedings of the 30th Conference on Winter Simulation*. Retrieved from http://delivery.acm.org/10.1145/300000/293186/p49-law.pdf?ip=193.137.127.2&id=293186&acc=ACTIVE SERVICE&key=2E5699D25B4FE09E.DE7F6CA86C6E574F.4D4702B0C3E38B35.4D4702B0C3E38B35&CFID=888317604&CFTOKEN=

60073046&__acm__=1484306431_4aac4789f04c7479b590b6071710

Law, a M., & Kelton, W. D. (2000). Simulation Modeling and Analysis. *ACM Transactions on Modeling and Computer Simulation*. http://doi.org/10.1145/1667072.1667074

Leitão, P., & Restivo, F. (2006). ADACOR: A holonic architecture for agile and adaptive manufacturing control. *Computers in Industry*, *57*(2), 121–130. http://doi.org/10.1016/j.compind.2005.05.005

Leung, C. W., Wong, T. N., Mak, K. L., & Fung, R. Y. K. (2010). Integrated process planning and scheduling by an agent-based ant colony optimization. *Computers and Industrial Engineering*, *59*(1), 166–180. http://doi.org/10.1016/j.cie.2009.09.003

Li, J., Dai, X., Meng, Z., Dou, J., & Guan, X. (2009). Rapid design and reconfiguration of Petri net models for reconfigurable manufacturing cells with improved net rewriting systems and activity diagrams. *Computers & Industrial Engineering*, *57*(4), 1431–1451. http://doi.org/10.1016/j.cie.2009.07.013

Li, X., Zhang, C., Gao, L., Li, W., & Shao, X. (2010). An agent-based approach for integrated process planning and scheduling. *Expert Systems with Applications*, *37*(2), 1256–1264. http://doi.org/10.1016/j.eswa.2009.06.014

Mařík, V., & McFarlane, D. (2005). Industrial adoption of agent-based technologies. *IEEE Intelligent Systems*. http://doi.org/10.1109/MIS.2005.11

Mendes, J. M., Leitão, P., Colombo, A. W., & Restivo, F. (2008). Service-oriented control architecture for reconfigurable production systems. *IEEE International Conference on Industrial Informatics (INDIN)*, 744–749. http://doi.org/10.1109/INDIN.2008.4618201

Monostori, L., Váncza, J., & Kumara, S. R. T. (2006). Agent-Based Systems for Manufacturing. *CIRP Annals - Manufacturing Technology*, *55*(2), 697–720. http://doi.org/10.1016/j.cirp.2006.10.004

Oliveira, J. A. B. de. (2003). Coalition based approach for shop floor agility – a multiagent approach. Retrieved from http://hdl.handle.net/10362/2483

Onori, M., & Barata, J. (2009). Evolvable Production Systems : Mechatronic Production Equipment With Process-Based Distributed Control. *IFAC Proceedings Volumes*, *42*(16), 80–85. http://doi.org/10.3182/20090909-4-JP-2010.00016

Onori, M., Barata, J., & Frei, R. (2006). Evolvable Assembly Systems Basic Principles. In *Information Technology For Balanced Manufacturing*

Systems (pp. 317–328). Boston, MA: Springer US. http://doi.org/10.1007/978-0-387-36594-7_34

Pegden, C. D. (2007). Simio: A new simulation system based on intelligent objects. *2007 Winter Simulation Conference*, 2293–2300. http://doi.org/10.1109/WSC.2007.4419867

Ribeiro, L., & Barata, J. (2011). Re-thinking diagnosis for future automation systems: An analysis of current diagnostic practices and their applicability in emerging IT based production paradigms. *Computers in Industry*, *62*(7), 639–659. http://doi.org/10.1016/j.compind.2011.03.001

Rocha, A. (2013). *An agent based architecture for material handling systems*. Retrieved from http://run.unl.pt/handle/10362/10504

Rocha, A. D., Barata, D., Di Orio, G., Santos, T., & Barata, J. (2015). PRIME as a Generic Agent Based Framework to Support Pluggability and Reconfigurability Using Different Technologies (pp. 101–110). Springer International Publishing. http://doi.org/10.1007/978-3-319-16766-4_11

Rocha, A. D., Barroca, P., Maso, G. D., & Barata, J. (2016). Environment to Simulate Distributed Agent Based Manufacturing Systems.

Rockwell Automation. (2016). Arena, Simulation Software. Retrieved July 28, 2016, from https://www.arenasimulation.com/

Sabar, M., Montreuil, B., & Frayret, J.-M. (2009). A multi-agent-based approach for personnel scheduling in assembly centers. *Engineering Applications of Artificial Intelligence*, *22*(7), 1080–1088. http://doi.org/10.1016/j.engappai.2009.02.009

Shapiro, V., & Farin, G. (2002). Handbook of computer aided geometric design. *Handbook of Computer Aided Geometric Design*. http://doi.org/10.1016/B978-044451104-1/50021-6

SIMIO LLC. (2016). SIMIO, SImulation Modeling framework based on Intelligent Objects. Retrieved July 28, 2016, from http://www.simio.com/index.php

Technology Transfer System S.r.l. (2016). Technology Transfer System S.r.l. Retrieved July 28, 2016, from http://www.ttsnetwork.net/en/

Ueda, K. (1992). A concept for bionic manufacturing systems based on DNA-type information. In *Human Aspects in Computer Integrated Manufacturing* (pp. 853–863). http://doi.org/10.1016/B978-0-444-89465-6.50078-8

Ueda, K., Hatono, I., Fujii, N., & Vaario, J. (2000). Reinforcement Learning Approaches to Biological Manufacturing Systems. *CIRP Annals - Manufacturing Technology*, *49*(1), 343–346.

http://doi.org/10.1016/S0007-8506(07)62960-6

Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., & Peeters, P. (1998). Reference architecture for holonic manufacturing systems: PROSA. *Computers in Industry*, *37*(3), 255–274. http://doi.org/10.1016/S0166-3615(98)00102-X

Veiga, Â. M. H. (2013). A multiagent based shop floor transportation system simulator. Retrieved from http://hdl.handle.net/10362/10415

Vrba, P., Tichy, P., & Mařík, V. (2011). Rockwell Automation's Holonic and Multiagent Control Systems Compendium. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, *41*(1), 14–30. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5545420